UNIVERSIDAD DE LA CORUÑA
Departamento de Computación

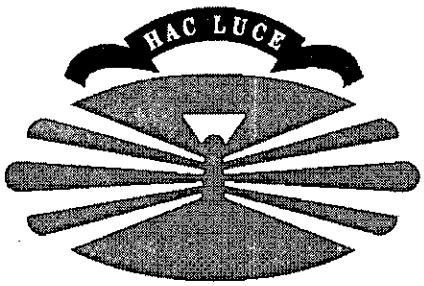
TESIS DOCTORAL

Técnicas de Análisis Sintáctico Robusto para la Etiquetación del Lenguaje Natural

Autor:
JORGE GRAÑA GIL

Directores:
MANUEL VILARES FERRO
MARTIN RAJMAN

Septiembre de 2000



UNIVERSIDAD DE LA CORUÑA
Departamento de Computación

TESIS DOCTORAL

Técnicas de Análisis Sintáctico Robusto para la Etiquetación del Lenguaje Natural

Autor:
JORGE GRAÑA GIL

Directores:
MANUEL VILARES FERRO
MARTIN RAJMAN

Septiembre de 2000

Dr. Manuel Vilares Ferro
Profesor Titular de Universidad
Departamento de Computación
Universidad de La Coruña

Dr. Martin Rajman
Maître d'Enseignement et de Recherche
Département d'Informatique
École Polytechnique Fédérale de Lausanne

CERTIFICAN:

Que la memoria titulada "*Técnicas de Análisis Sintáctico Robusto para la Etiquetación del Lenguaje Natural*" ha sido realizada por D. Jorge Graña Gil bajo nuestra dirección en el Departamento de Computación de la Universidad de La Coruña, y concluye la Tesis que presenta para optar al grado de Doctor en Informática.

La Coruña, 11 de Septiembre de 2.000

Fdo: Dr. Manuel Vilares Ferro
Codirector de la Tesis Doctoral

Fdo: Dr. Martin Rajman
Codirector de la Tesis Doctoral

A Cristina

Resumen

Técnicas de Análisis Sintáctico Robusto para la Etiquetación del Lenguaje Natural

El objetivo último que persigue el Procesamiento del Lenguaje Natural es el perfecto análisis y entendimiento de los lenguajes humanos. Actualmente, estamos todavía lejos de conseguir este objetivo. Por esta razón, la mayoría de los esfuerzos de investigación de la lingüística computacional han sido dirigidos hacia tareas intermedias que dan sentido a alguna de las múltiples características estructurales inherentes a los lenguajes, sin requerir un entendimiento completo. Una de esas tareas es la asignación de categorías gramaticales a cada una de las palabras del texto. Este proceso se denomina también etiquetación.

La eliminación de ambigüedades es una tarea crucial durante el proceso de etiquetación de un texto en lenguaje natural. Si tomamos aisladamente, por ejemplo, la palabra `sobre`, vemos que puede tener varias categorías posibles en español: sustantivo, preposición o verbo. Sin embargo, si examinamos el contexto en el que aparece dicha palabra, seguramente sólo una de ellas es posible. Por otra parte, el interés se centra también en asignar una etiqueta a todas aquellas palabras que aparecen en los textos, pero que no están presentes en nuestro diccionario, y garantizar de alguna manera que ésa es la etiqueta correcta. Un buen rendimiento en esta fase asegura la viabilidad de procesamientos posteriores tales como los análisis sintáctico y semántico.

Tradicionalmente, el problema de la etiquetación se aborda a partir de recursos lingüísticos bajo la forma de diccionarios y textos escritos, previamente etiquetados o no. Esta línea de desarrollo se denomina lingüística basada en corpus. Dichos textos se utilizan para ajustar los parámetros de funcionamiento de los etiquetadores. Este proceso de ajuste se denomina entrenamiento. Las técnicas tradicionales engloban métodos estocásticos, tales como los modelos de Markov ocultos, los árboles de decisión o los modelos de máxima entropía, y también aproximaciones basadas en reglas, tales como el aprendizaje de etiquetas basado en transformaciones y dirigido por el error.

La mayoría de las herramientas basadas en estos paradigmas de etiquetación resultan ser de propósito general, en el sentido de que pueden ser aplicadas a textos en cualquier idioma. Ésta es una idea muy atractiva, pero surge la duda de si un etiquetador diseñado especialmente para una lengua dada puede ofrecer mejores rendimientos o no. Por tanto, el primer objetivo del presente trabajo consiste en implementar una nueva herramienta de etiquetación que permita integrar información específica para el español, y posteriormente realizar una evaluación exhaustiva de todos estos modelos. Este estudio es de gran interés ya en sí mismo, dado que los recursos lingüísticos disponibles para el español no abundan, y por tanto existen todavía muy pocas cifras concretas que proporcionen una idea clara del comportamiento de los etiquetadores sobre nuestro idioma.

Aún con todo esto, un pequeño porcentaje de palabras etiquetadas erróneamente (2-3%) es una característica que está siempre presente en los sistemas de etiquetación puramente estocásticos. Por esta razón, apoyamos la idea del uso de estos sistemas en combinación con información sintáctica, esto es, con técnicas de análisis sintáctico robusto, y éste es precisamente

el segundo de los objetivos del presente trabajo.

Cuando una frase es correcta, pero la gramática no es capaz de analizarla, todavía es posible considerar los subárboles correspondientes a los análisis parciales de fragmentos válidos de la frase. El posterior estudio de estos subárboles puede ser utilizado, por ejemplo, para completar la gramática, generando automáticamente las reglas sintácticas necesarias para analizar la frase. Éste es precisamente el objetivo más ambicioso del análisis sintáctico robusto. En nuestro caso particular, resulta de especial interés la consideración de las etiquetas de las palabras de dichos subárboles como información adicional de apoyo para las técnicas tradicionales de etiquetación. La estrategia consiste en combinar esas subsecuencias de etiquetas para generar varias etiquetaciones completas posibles de la frase en cuestión, y posteriormente aplicar un filtro estadístico para elegir la secuencia global más probable.

Es bienvenido cualquier comentario, sugerencia o petición de información adicional. Para contactar con el autor:

Jorge Graña Gil
Universidad de La Coruña
Facultad de Informática
Departamento de Computación
Campus de Elviña, s/n
15071 - La Coruña
España

grana@dc.fi.udc.es
http://www.dc.fi.udc.es/~grana

Abstract

Robust Parsing Techniques for Natural Language Tagging

The ultimate goal of research on Natural Language Processing is to parse and understand human languages. Currently, we are still far from achieving this goal. For this reason, much research in computational linguistics has focussed on intermediate tasks that make sense of some of the structure inherent in language without requiring complete understanding. One such task is part-of-speech tagging, or simply tagging.

Elimination of lexical ambiguities is a crucial task during the process of tagging a text in natural language. If we take in isolation, for instance, the word `time`, we can see that it has several possible tags in English: substantive, adjective or verb. However, if we examine the context in which the word appears, only one of the tags is possible. In addition, we are also interested in being able to give a tag to all the words that appear in a text, but are not present in our dictionary, and to guarantee somehow that this tag is the correct one. A good performance at this stage will improve the viability of syntactic and semantic analysis.

Traditionally, the starting point for tagging is linguistic resources like dictionaries and written texts, previously tagged or not. This research line is called corpus-based linguistics. These corpora are used to tune the running parameters of the taggers. This tuning process is called training. Traditional techniques involve stochastic methods, such as hidden Markov models, decision trees or maximum entropy models, and also rule-based approaches, such as transformation-based error-driven learning of tags.

Most tools based on these tagging paradigms are general purpose, to the effect that they can be applied to texts in any language. This is a very attractive idea, but begs the question of whether a tagger specifically designed for a particular language is able to provide better performance. Therefore, the first goal of the present work is to implement a new tagger able to integrate specific information on Spanish, and then to perform an exhaustive evaluation of all the above-mentioned models. This study is in itself very interesting, because there are very few available linguistic resources for Spanish and very few concrete data about the behaviour of taggers on our language.

However, a small percentage of wrongly tagged words (2-3%) is a feature that is always present in pure stochastic taggers. For this reason we support the idea of using these in combination with syntactic information, that is, with robust parsing techniques, and this is the second goal of the present work.

When a sentence is correct and the grammar is not able to parse it, it is still possible to consider all subtrees corresponding to all partial analyses of valid fragments of the sentence. A later study of these subtrees can be used, for instance, to complete the grammar by automatically generating all the syntactic rules we need to parse the sentence. This is in fact the most ambitious goal in robust parsing. In our particular case, it is important to consider all the word tags of the subtrees in question as additional information that can be useful for traditional techniques of tagging. Our strategy combines these subsequences of tags in order to generate several complete

taggings for a given sentence, and then applies a probabilistic filter to choose the most probable one.

We value and appreciate your comments, suggestions and requests for further information. You may contact the author at:

Jorge Graña Gil
Universidad de La Coruña
Facultad de Informática
Departamento de Computación
Campus de Elviña, s/n
15071 - La Coruña
Spain

grana@dc.fi.udc.es
http://www.dc.fi.udc.es/~grana

Agradecimientos

Quiero expresar mi más sincero agradecimiento a las personas que me han ayudado a llegar hasta aquí, y a todos los que durante estos últimos años han colaborado de muy diversas maneras en este trabajo.

A mis directores, Manuel Vilares y Martin Rajman, por ser como la línea de investigación en la que me han brindado la oportunidad de trabajar, el Procesamiento del Lenguaje Natural, que combina a la perfección grandes dosis de técnica y de humanidad.

A los miembros del grupo *Compiladores y Lenguajes* de la Universidad de La Coruña, especialmente a Miguel Alonso y a David Cabrero, que desde la perspectiva de la sintaxis, siempre han sabido aportar ese grano de visión global que necesita toda persona que se enreda demasiado en un tema, como ha sido mi caso con el léxico.

A los miembros del *Grupo de Sintaxis del Español* de la Universidad de Santiago de Compostela, y del proyecto *Etiquetador-lematizador para el gallego actual* del Centro Ramón Piñeiro para Investigación en Humanidades, especialmente a Guillermo Rojo, Paula Santalla, Susana Sotelo y Eva Domínguez, por tantas y tan enriquecedoras reuniones de trabajo.

A los miembros del *Grupo de Procesamiento de Lenguaje Natural* de la Escuela Politécnica Federal de Lausanne, y muy especialmente a Jean-Cedric Chappelier, cuyos comentarios han sido el origen de muchas de las ideas plasmadas en este trabajo.

A Teresa Romero, Óscar Sacristán y Mario Barcala, que con su juventud e ímpetu me han recordado a mí mismo en mis comienzos, y me han hecho recuperar la ilusión por el trabajo en los momentos de menos motivación. Espero que el futuro nos depare algún otro proyecto juntos.

A Ricardo Cao, Margarita Alonso y Alberto Valderruten, por sus siempre apasionados y alentadores comentarios y revisiones, que han ayudado en gran medida a elevar el rigor científico de este documento.

A Antonio Blanco, al que solemos atormentar en los malos ratos en busca del afecto y confianza que sólo él sabe inculcar, porque no en vano nos ha iniciado a casi todos en este fascinante mundo.

A mis padres porque nunca escatimaron ni lo más mínimo para darnos a mis hermanos y a mí la mejor de las educaciones, y a Cristina por ser el antídoto perfecto de todos mis temores e inseguridades.

A todos y por todo, muchísimas gracias.

Índice General

Índice de Figuras

Índice de Tablas

Parte I

Etiquetación del lenguaje natural: conceptos previos

Capítulo 1

Introducción

El objetivo último que persigue el Procesamiento del Lenguaje Natural o NLP[1] es el perfecto análisis y entendimiento de los lenguajes humanos. Actualmente, estamos todavía lejos de conseguir este objetivo. Por esta razón, la mayoría de los esfuerzos de investigación en lo que al NLP se refiere han sido dirigidos hacia tareas intermedias que dan sentido a alguna de las múltiples características estructurales inherentes a los lenguajes, sin requerir un entendimiento completo. Una de esas tareas es la asignación de categorías gramaticales a cada una de las palabras del texto. Este proceso se denomina también *etiquetación* de las partes del discurso o POST[2]. En definitiva, se trata de decidir si cada palabra es un sustantivo, un adjetivo, un verbo, etc. Por ejemplo, si consideramos aisladamente la palabra `sobre`, vemos que puede ser:

- Un sustantivo, como por ejemplo en la frase: `mételo en ese sobre`.
- Una preposición, como por ejemplo en la frase: `déjalo sobre la mesa`.
- O un verbo en primera o tercera persona del presente de subjuntivo del verbo `sobrar`, como por ejemplo en la frase: `dame lo que te sobre`.

Es decir, si echamos un vistazo al contexto en el que dicha palabra aparece, es muy probable que sólo una de esas tres etiquetas sea la correcta. El proceso de etiquetación debe eliminar por tanto este tipo de ambigüedades y encontrar cuál es el papel más probable que juega cada palabra dentro de una frase. Es más, dicho proceso debe ser capaz también de asignar una etiqueta a cada una de las palabras que aparecen en un texto y que no están presentes en nuestro diccionario, y garantizar de alguna manera que ésa es la etiqueta correcta.

Como ya hemos visto, la etiquetación es un problema de ámbito limitado. En lugar de construir un análisis completo, simplemente se establecen las categorías de las palabras, y se dejan de lado problemas tales como por ejemplo el de encontrar la correcta ligadura de las frases preposicionales. Debido a esto, la etiquetación es más sencilla de resolver que los análisis sintáctico o semántico, y el rendimiento es bastante elevado. Las aproximaciones más exitosas son capaces de etiquetar correctamente alrededor del 95-97% de las palabras. Sin embargo, es importante señalar que estas cifras tan impactantes no son tan buenas como parecen, ya que normalmente son el resultado de una evaluación palabra por palabra. Por ejemplo, en muchos géneros de escritura tales como los artículos periodísticos, donde el número medio de palabras

[1] A lo largo de la presente memoria intentaremos hacer referencia a todos los conceptos propios del lenguaje especializado de la materia a tratar con su correspondiente término en español; no obstante, la mayoría de los acrónimos utilizados constituirán una excepción a este principio, ya que muchas veces su traducción resulta excesivamente artificial; en este caso, utilizaremos la sigla NLP, que responde al término inglés *Natural Language Processing*.

[2] *Part-Of-Speech Tagging*.

por frase puede ser de unas veinte o veinticinco, un rendimiento del 95% todavía implica que pueden aparecer entre una y dos palabras mal etiquetadas en cada frase. Además, estos errores no siempre se localizan en las categorías más pobladas, tales como sustantivos, adjetivos o verbos, donde en principio parece más probable el encontrarse con palabras desconocidas. Muchas veces, los errores aparecen asociados a las partículas que conectan los sintagmas entre sí, tales como preposiciones, conjunciones o relativos, y pueden hacer que una frase tome un significado muy distinto del original.

Aún con todo esto, y a pesar de sus limitaciones, la información que se obtiene mediante la etiquetación es muy útil. Las potenciales aplicaciones de los textos en lenguaje natural aumentan cuando dichos textos están anotados, y el primer nivel lógico de anotación es normalmente la asignación de este tipo de etiquetas gramaticales a cada una de las palabras. Por tanto, los textos ya no serán vistos como una mera secuencia de caracteres, sino como una secuencia de entidades lingüísticas con algún tipo de significado natural. El texto etiquetado puede utilizarse entonces para introducir nuevos tipos de anotaciones, normalmente mediante posteriores análisis sintácticos o semánticos, o puede utilizarse también para recoger datos estadísticos sobre el uso de un idioma que pueden servir para múltiples aplicaciones. El trabajo a partir de las palabras etiquetadas hace mucho más viables tareas tales como el proceso y síntesis del diálogo, la lexicografía computacional o, como veremos más adelante, el más reciente y novedoso tema de la recuperación de información.

1.1 Fuentes de información relevantes para la etiquetación

¿Cómo se puede decidir cuál es la etiqueta correcta de una palabra? Existen esencialmente dos fuentes de información:

1. La primera de ellas consiste en mirar las etiquetas de las otras palabras que pertenecen al contexto en el que aparece la que nos interesa. Esas palabras podrían ser también ambiguas, pero el hecho de observar secuencias de varias etiquetas nos puede dar una idea de cuáles son comunes y cuáles no lo son. Por ejemplo, en inglés, una secuencia como artículo-adjetivo-sustantivo es muy común, mientras que otras secuencias como artículo-adjetivo-verbo resultan muy poco frecuentes o prácticamente imposibles. Por tanto, si hubiera que elegir entre sustantivo o verbo para etiquetar la palabra `play` en la frase `a new play`, obviamente optaríamos por la primera de las etiquetas.

   Este tipo de estructuras constituyen la fuente de información más directa para el proceso de etiquetación, pero por sí misma no resulta demasiado exitosa: uno de los primeros etiquetadores basado en reglas deterministas que utilizaba este tipo de patrones sintagmáticos etiquetaba correctamente sólo el 77% de las palabras [Greene y Rubin 1971]. Una de las razones de este rendimiento tan bajo es que en inglés las palabras que pueden tener varias etiquetas son efectivamente muy numerosas, debido sobre todo a procesos productivos como el que permite a casi todos los sustantivos que podamos tener en nuestro diccionario transformarse y funcionar como verbos, con la consiguiente pérdida de la información restrictiva que es necesaria para el proceso de etiquetación.

2. Sin embargo, existen palabras que, aunque puedan ser usadas como verbos, su aparición es mucho más probable cuando funcionan como sustantivos. Este tipo de consideraciones sugiere la segunda fuente de información: el simple conocimiento de la palabra concreta puede proporcionarnos datos muy valiosos acerca de la etiqueta correcta. La utilidad de esta información fue demostrada de manera concluyente por Charniak, quien puso de manifiesto que un etiquetador que simplemente asigne la etiqueta más común a cada

palabra puede alcanzar un índice de acierto del 90% [Charniak *et al.* 1993]. De hecho, el rendimiento de este etiquetador fue utilizado como referencia en todos los trabajos posteriores de esa época.

La información léxica de las palabras resulta tan útil porque la distribución de uso de una palabra a lo largo de todas sus posibles etiquetas suele ser rara. Incluso las palabras con un gran número de etiquetas aparecen típicamente con un único uso o etiqueta particular.

Efectivamente, esta distribución es tan peculiar que casi siempre existe esa etiqueta denominada *básica*, mientras que las otras representan usos derivados de ésta, y como resultado se han producido algunos conflictos en relación con la manera en la que el término *etiqueta*, *categoría* o *parte del discurso* debe ser utilizado. En las gramáticas tradicionales, se pueden encontrar palabras clasificadas como *un sustantivo que está siendo utilizado como un adjetivo*, lo cual confunde la etiqueta básica del lexema de la palabra, con la función real que dicha palabra está desempeñando dentro del contexto. Aquí, al igual que en la lingüística moderna en general, nos centraremos siempre en el segundo concepto, es decir, en el uso real de las palabras dentro de cada frase concreta.

En cualquier caso, la distribución de uso de las palabras proporciona una información adicional de gran valor, y es por ello por lo que parece lógico esperar que las aproximaciones estadísticas al proceso de etiquetación den mejores resultados que las aproximaciones basadas en reglas deterministas. En éstas últimas, uno sólo puede decir que una palabra puede o no puede ser un verbo, y existe la tentación de dejar fuera la posibilidad de ser verbo cuando ésta es muy rara, bajo la creencia de que esto aumentará el rendimiento global, mientras que en una aproximación estadística se puede decir *a priori* que una palabra tiene una gran probabilidad de ser un sustantivo, pero también que existe una posibilidad, por remota que sea, de ser un verbo o incluso cualquier otra etiqueta. Por tanto, se podría argumentar que la eliminación de ambigüedades léxicas es una tarea en la cual un marco que permita especificar información cuantitativa para representar el conocimiento lingüístico es más adecuado que uno puramente simbólico. A lo largo de este trabajo veremos si esto es cierto o no, aunque seguramente llegaremos a la conclusión de que una solución de compromiso entre las dos aproximaciones es la más adecuada. De hecho, hoy en día, los etiquetadores modernos utilizan de alguna manera una combinación de la información sintagmática proporcionada por las secuencias de etiquetas y de la información léxica proporcionada por las palabras.

1.2 Los primeros etiquetadores

Los trabajos que inicialmente se desarrollaron en relación con el proceso de etiquetación eran simplemente programas que buscaban la categoría de las palabras en un diccionario. El primer sistema conocido que realmente intentaba asignar etiquetas en función del contexto sintagmático es el programa basado en reglas presentado en [Klein y Simmons 1963], aunque aproximadamente la misma idea fue presentada en [Salton y Thorpe 1962]. Klein y Simmons utilizan los términos *etiqueta* y *etiquetación*, aunque son aparentemente de uso intercambiable con *código* y *codificación*. El primer etiquetador probabilístico conocido es [Stolz *et al.* 1965]. Este sistema asignaba inicialmente etiquetas a algunas palabras mediante el uso de un diccionario, de reglas morfológicas y de otras reglas confeccionadas a medida. El resto de las palabras se etiquetaban entonces usando probabilidades condicionadas calculadas a partir de secuencias de etiquetas. Ni que decir tiene que no se trataba de un modelo probabilístico bien definido.

Dos grandes grupos de trabajo, uno en la Universidad Brown y otro en la Universidad de Lancaster, emplearon considerables recursos para etiquetar dos grandes *corpora* de texto: el

corpus BROWN y el corpus LOB (Lancaster-Oslo-Bergen). Ambos grupos coincidieron en que la existencia de un corpus anotado sería de incalculable valor para la investigación en el campo de la etiquetación, y es cierto que sin estos dos *corpora* etiquetados el progreso de esta línea de trabajo hubiera sido extremadamente duro, si no imposible. La disponibilidad de grandes cantidades de texto etiquetado es sin lugar a dudas una importante razón que explica el hecho de que la etiquetación haya sido un área de investigación tan activa.

El corpus BROWN fue preetiquetado automáticamente con el etiquetador basado en reglas TAGGIT [Greene y Rubin 1971]. Esta herramienta utilizaba información léxica sólo para limitar las etiquetas de las palabras y sólo aplicaba reglas de etiquetación cuando las palabras del contexto no presentaban ambigüedades. La salida de este etiquetador se corrigió entonces manualmente. Este esfuerzo duró años, pero finalmente proporcionó los datos de entrenamiento que constituyeron la base de numerosos trabajos posteriores.

Uno de los primeros etiquetadores basado en modelos de Markov ocultos o HMM,s[3] fue creado en la Universidad de Lancaster como parte del proyecto de etiquetación del corpus LOB [Garside *et al.* 1987, Marshall 1987]. El punto central de este etiquetador era el manejo de probabilidades para las secuencias de bigramas de etiquetas, con uso limitado de un contexto de mayor orden, y las probabilidades de asignación de una palabra a sus diferentes etiquetas eran gestionadas mediante factores de descuento diseñados a medida. Los etiquetadores basados en modelos de Markov que etiquetan utilizando ambos tipos de información, las probabilidades de las palabras y las probabilidades de transición entre etiquetas, fueron introducidos en [Church 1988] y [DeRose 1988].

A pesar de que los trabajos de Church y DeRose fueron la clave del resurgir de los métodos estadísticos en lingüística computacional, la aplicación de los HMM,s al proceso de etiquetación había comenzado realmente mucho antes en los centros de investigación de IBM en Nueva York y París [Jelinek 1985, Derouault y Merialdo 1986]. Otras referencias de los primeros trabajos sobre etiquetación probabilística incluyen [Bahl y Mercer 1976, Baker 1975, Foster 1991].

1.3 Rendimiento y precisión de los etiquetadores

Las cifras de rendimiento conocidas para los etiquetadores se encuentran casi siempre dentro del rango del 95 al 97% de acierto, cuando se calculan sobre el conjunto de todas las palabras de un texto. Algunos autores proporcionan la precisión sólo para los términos ambiguos, en cuyo caso las cifras son por supuesto menores. Sin embargo, el rendimiento depende considerablemente de una serie de factores, tales como los siguientes:

- La cantidad de texto de entrenamiento disponible. En general, cuanto más texto se tenga, mejor.

- El juego de etiquetas[4]. Normalmente, cuanto más grande es el conjunto de etiquetas considerado, existe más ambigüedad potencial, y la tarea de etiquetación se vuelve más compleja. Por ejemplo, en inglés, algunos juegos de etiquetas hacen una distinción entre `to` como preposición y `to` como marca de infinitivo, y otros no. En el primero de los casos, la palabra `to` podría etiquetarse incorrectamente.

- La diferencia entre, por un lado el corpus de entrenamiento y el diccionario, y por otro el corpus de aplicación. Si los textos de entrenamiento y los textos que posteriormente se van a etiquetar proceden de la misma fuente, por ejemplo, textos de la misma época, o extraídos de un mismo periódico particular, entonces la precisión será alta. Normalmente,

[3] *Hidden Markov Models.*
[4] También denominado *tag set.*

los resultados que los investigadores proporcionan sobre sus etiquetadores provienen de situaciones como ésta. Pero si los textos de aplicación pertenecen a un periodo de tiempo distinto, a una fuente distinta, o a un género o estilo distinto, por ejemplo, textos científicos contra textos periodísticos, entonces el rendimiento será bajo.

- Las palabras desconocidas. Un caso especial del punto anterior es la cobertura del diccionario. La aparición de palabras desconocidas puede degradar el rendimiento. Una situación típica en la cual el porcentaje de palabras fuera de vocabulario puede ser alto es cuando se intenta etiquetar material procedente de algún dominio técnico.

Un cambio en cualquiera de estas cuatro condiciones puede producir un impacto muy fuerte en la precisión de los etiquetadores, pudiendo provocar que el rendimiento se reduzca de manera dramática. Si el conjunto de entrenamiento es pequeño, el juego de etiquetas grande, y el corpus a etiquetar significativamente diferente del corpus de entrenamiento, o si nos enfrentamos a un gran número de palabras desconocidas, el rendimiento puede caer muy por debajo del rango de cifras citado anteriormente.

Es importante también señalar que efectivamente estos factores son externos al proceso de etiquetación y al método elegido para realizar dicho proceso. Es por ello que el efecto que producen es a menudo mucho mayor que la influencia ejercida por el propio método en sí.

1.4 Aplicaciones de la etiquetación

El amplio interés que ha surgido alrededor de la etiquetación está fundamentado en la creencia de que muchas aplicaciones de NLP se pueden beneficiar de los textos etiquetados. Sin embargo, aún conociendo esta motivación, resulta sorprendente el hecho de que existan más trabajos y referencias bibliográficas sobre el proceso de etiquetación aisladamente, que sobre la aplicación de los etiquetadores a tareas de interés inmediato. A pesar de esto, intentaremos enumerar aquí las aplicaciones más importantes en las que la etiquetación ha jugado y está jugando un papel relevante. Además de las citadas anteriormente, estas aplicaciones podrían ser las siguientes:

- La mayoría de las aplicaciones requieren un paso de procesamiento adicional posterior a la etiquetación: el análisis sintáctico parcial[5], que puede reflejar varios niveles de detalle dentro del análisis sintáctico. Los analizadores parciales más simples se limitan a buscar las frases nominales de una oración. Otras aproximaciones más sofisticadas asignan funciones gramaticales a esas frases nominales (sujeto, objeto directo, objeto indirecto, etc.) y al mismo tiempo proporcionan información parcial sobre las ligaduras[6]. Una presentación conjunta del análisis sintáctico parcial y de la etiquetación puede verse en [Abney 1996].

- Un uso importante de la combinación de la etiquetación y el análisis sintáctico parcial es el proceso de adquisición automática de información léxica[7]. El objetivo general de este proceso es desarrollar algoritmos y técnicas estadísticas para rellenar los huecos de información sintáctica y semántica que existen en los diccionarios electrónicos, mediante el estudio de los patrones de palabras que se pueden observar en grandes *corpora* de textos. Existen multitud de problemas relacionados con la adquisición de información léxica: la colocación de las palabras, las preferencias de selección[8], los marcos de subcategorización[9],

[5] También denominado *partial parsing*.

[6] Por ejemplo, *esta frase nominal está ligada a otra frase de la derecha*, la cual puede aparecer especificada o no.

[7] También denominado *lexical acquisition*.

[8] Por ejemplo, el verbo `comer` normalmente lleva como objeto directo elementos relacionados con la comida.

[9] Por ejemplo, el beneficiario del verbo `contribuir` se expresa mediante una frase preposicional encabezada por la preposición `a`.

o la categorización semántica[10]. La mayoría de este tipo de propiedades de las palabras no se suele cubrir completamente en los diccionarios. Esto es debido principalmente a la productividad de los lenguajes naturales: constantemente inventamos nuevas palabras o nuevos usos de las palabras que ya conocemos. Incluso aunque fuera posible crear un diccionario que reflejara todas las características del lenguaje actual, inevitablemente estaría incompleto en cuestión de pocos meses. Ésta es la razón por la cual la adquisición automática de información léxica es tan importante en el NLP estadístico.

- Otra aplicación importante es la extracción de información[11]. El objetivo principal de la extracción de información es encontrar valores para un conjunto predeterminado de ranuras de información de una plantilla. Por ejemplo, una plantilla de información metereológica podría tener ranuras para el tipo de fenómeno (tornado, tormenta de nieve, huracán), la localización del evento (la bahía coruñesa, Europa Central, Venezuela), la fecha (hoy, el próximo domingo, el 27 de diciembre de 1999), y el efecto causado (apagón general, inundaciones, accidentes de tráfico, etc.). La etiquetación y el análisis sintáctico parcial ayudan a identificar las entidades que sirven para rellenar las ranuras de información y las relaciones entre ellas. En cierta manera, podría verse la extracción de información como un proceso de etiquetación donde las categorías son semánticas, en lugar de gramaticales. Sin embargo, en la práctica se tiende a emplear métodos bastante diferentes [Cardie 1997], ya que las secuencias locales proporcionan mucha menos información sobre categorías semánticas que sobre categorías gramaticales.

- La etiquetación y el análisis sintáctico parcial se pueden utilizar también para encontrar los términos de indexación adecuados en los sistemas de recuperación de información[12]. La mejor unidad de tratamiento para decidir qué documentos están relacionados con las consultas de los usuarios a menudo no es la palabra individual. Frases como `Estados Unidos de América` o `educación secundaria` pierden gran parte de su significado si se rompen en palabras individuales. El rendimiento de la recuperación de información se puede incrementar si la etiquetación y el análisis sintáctico parcial se dirigen hacia el reconocimiento de la frase nominal y si las asociaciones consulta-documento se realizan apoyándose en este tipo de unidad de información de más alto rango de significado [Fagan 1987, Smeaton 1992, Strzalkowski 1995]. Otra línea de investigación relacionada se ocupa de la normalización de frases, es decir, del estudio de las múltiples variantes de términos que representan realmente la misma unidad de información básica[13] [Jacquemin *et al.* 1997].

- Por último, existen también trabajos relacionados con los llamados sistemas de respuesta de preguntas[14], los cuales intentan responder a una cuestión del usuario devolviendo una frase nominal concreta, como por ejemplo un lugar, una persona o una fecha [Kupiec 1993, Burke *et al.* 1997]. Es decir, ante una pregunta como `¿quién mató a John Lennon?` obtendríamos como respuesta `Chapman`, en lugar de una lista de documentos tal y como ocurre en la mayoría de los sistemas de recuperación de información. Una vez más, el análisis de la consulta para determinar qué tipo de entidad está buscando el usuario y cómo está relacionada con las frases nominales que aparecen en la pregunta requiere etiquetación y análisis sintáctico parcial.

[10] Por ejemplo, cuál es la categoría semántica de una nueva palabra no presente en nuestro diccionario.

[11] O *information extraction*, y a veces también referenciada como *message understanding* (entendimiento del mensaje) o *data mining* (minería de datos).

[12] O también *information retrieval*.

[13] Por ejemplo, `book publishing` y `publishing of books`.

[14] También denominados *question answering systems*.

Terminamos, sin embargo, con un resultado no del todo positivo: los analizadores sintácticos probabilísticos mejor lexicalizados son hoy en día suficientemente buenos como para trabajar con texto no etiquetado y realizar la etiquetación por sí mismos, en lugar de utilizar un etiquetador como preprocesador [Charniak 1997]. Por tanto, el papel de los etiquetadores parece ser más bien el de un componente que aligera rápidamente la complejidad de los textos y proporciona suficiente información para multitud de interesantes tareas de NLP, y no el de una fase de preprocesado deseable e imprescindible para todas y cada unas de las aplicaciones.

1.5 Motivación y objetivos de la tesis

Tomando como punto de partida la tesis doctoral [Vilares 1992], un sistema de análisis sintáctico incremental para gramáticas independientes del contexto arbitrarias, se creó en el seno del Departamento de Computación de la Universidad de La Coruña un grupo especializado en el diseño de aplicaciones para el procesamiento automático de lenguajes tanto formales como naturales. Este grupo se denomina actualmente COLE[15]. Poco a poco fue surgiendo la necesidad de dotar a esta herramienta de módulos especializados en el tratamiento del léxico, con el objetivo de extenderla y convertirla en un entorno eficaz de NLP.

Desde ese momento, han sido numerosos los proyectos y los contactos con otros grupos de investigación que se han planteado para la realización de ésta y de otras tareas. De todos ellos, nos gustaría destacar los siguientes:

- *Análisis automático del régimen verbal español*, en colaboración con el Grupo de Sintaxis del Español del Departamento de Lengua Española de la Universidad de Santiago de Compostela. Proyecto financiado por la *Xunta de Galicia* durante el periodo 1995-1996 (XUGA20403B95) y 1997-1999 (XUGA20402B97).

- *Etiquetador-lematizador para el gallego actual*, en colaboración con el Centro Ramón Piñeiro para Investigación en Humanidades. Convenio de colaboración entre la Universidad de La Coruña y la *Xunta de Galicia* durante el periodo 1995-2000.

- *Diseño de analizadores tabulares para lenguajes naturales*, en colaboración con el proyecto ATOLL[16] del INRIA[17]. Proyecto financiado por los gobiernos español y francés durante el periodo 1997-1998, a través de las acciones integradas HF96-36 y HF97-223.

- *Extensiones del formalismo HMM para la etiquetación de textos en lenguaje natural*, en colaboración con el Laboratorio de Inteligencia Artificial del Departamento de Informática de la EPFL[18]. Estancia de investigación financiada por la *Xunta de Galicia* durante 1998, y realizada por el autor de esta tesis.

- *Extracción y recuperación de información aplicando conocimiento lingüístico*, en colaboración con el Grupo de Sintaxis del Español del Departamento de Lengua Española de la Universidad de Santiago de Compostela, el Departamento de Traducción, Lingüística y Teoría de la Literatura de la Universidad de Vigo, el Centro Ramón Piñeiro para Investigación en Humanidades, la empresa *Digital Equipment Corporation* España, y la Editorial Compostela. Proyecto financiado por la Unión Europea durante el periodo 1998-2000 (1FD97-0047-C04-02).

[15] COmpiladores y LEnguajes.
[16] *ATelier d'Outils Logiciels pour le Langage naturel.*
[17] *Institut National de Recherche en Informatique et en Automatique*, Francia.
[18] *Ecole Polytechnique Fédérale de Lausanne*, Suiza.

- *Interrogación estructurada de bases de datos textuales*, en colaboración con el Departamento de Lógica de la Universidad de Santiago de Compostela. Proyecto financiado por la *Xunta de Galicia* durante el periodo 1999-2001 (PGIDT99XI10502B).

Todas estas colaboraciones resumen la trayectoria en progresión que ha seguido nuestro grupo de trabajo: desde los primeros contactos con los investigadores lingüistas, destinados a perfeccionar y formalizar nuestro conocimiento y tratamiento de las lenguas, en particular de las geográficamente más cercanas, español y gallego, hasta las últimas relaciones establecidas con equipos de trabajo fuertemente consolidados en el extranjero, que nos están permitiendo abordar proyectos de especial relevancia incluso para el mundo empresarial e industrial.

Así pues, los objetivos del presente trabajo, que se inscribe tanto en el terreno de la lingüística computacional, como en el de la lingüística basada en corpus, responden a las necesidades que han surgido a nivel léxico durante el desarrollo de dichos proyectos. Básicamente, dichos objetivos son los siguientes:

1. Como hemos visto, el problema de la etiquetación se aborda tradicionalmente a partir de recursos lingüísticos bajo la forma de diccionarios y textos escritos, previamente etiquetados o no. Esta línea de desarrollo es la que se denomina *lingüística basada en corpus*. Dichos textos se utilizan para ajustar los parámetros de funcionamiento de los etiquetadores. Este proceso de ajuste se denomina *entrenamiento*. Las técnicas tradicionales engloban métodos estocásticos, tales como los modelos de Markov ocultos [Church 1988], los árboles de decisión [Schmid 1994] o los modelos de máxima entropía [Ratnaparkhi 1996], y también aproximaciones basadas en reglas, tales como el aprendizaje de etiquetas basado en transformaciones y dirigido por el error [Brill 1993b]. La mayoría de las herramientas basadas en estos acercamientos al problema de la etiquetación resultan ser de propósito general, en el sentido de que pueden ser aplicadas a textos en cualquier idioma. Ésta es una idea muy atractiva, pero surge la duda de si un etiquetador diseñado especialmente para una lengua dada puede ofrecer mejores rendimientos o no.

   Por tanto, el primer objetivo del presente trabajo consiste en implementar una nueva herramienta de etiquetación estocástica que permita integrar información específica para el español, y posteriormente realizar una evaluación exhaustiva de todos estos modelos. Este estudio es de gran interés ya en sí mismo, dado que los recursos lingüísticos disponibles para el español no abundan, y por tanto existen todavía muy pocas cifras concretas que proporcionen una idea clara del comportamiento de los etiquetadores sobre nuestro idioma.

   Actualmente, la situación típica, en lo que se refiere al procesamiento automático del español, pasa por utilizar textos de entrenamiento muy pequeños, ya que se trata de recursos poco frecuentes, pero diccionarios muy grandes, ya que la morfología del idioma se conoce bien y el trabajo realizado para formalizarla ha sido mucho mayor. Por tanto, bajo el término de *integración de información específica de una lengua* entendemos todos aquellos métodos que hacen posible que los etiquetadores manejen el conocimiento morfológico de esa lengua en cuestión. En particular, en el presente trabajo se ha realizado un importante esfuerzo destinado a estudiar los métodos de integración de diccionarios externos dentro de un marco de etiquetación estocástica, y a comprobar de qué manera el uso de estos diccionarios puede ayudar a incrementar el rendimiento del proceso de etiquetación, en especial en situaciones donde el corpus de entrenamiento es pequeño.

2. Aún con todo esto, un pequeño porcentaje de palabras etiquetadas erróneamente (2-3%) es una característica que está siempre presente en los sistemas de etiquetación puramente estocásticos. Por esta razón, apoyamos la idea del uso de estos sistemas en combinación

con información sintáctica, esto es, con técnicas de análisis sintáctico robusto, y éste es precisamente el segundo de los objetivos del presente trabajo.

Cuando una frase es correcta, pero la gramática no es capaz de analizarla, todavía es posible considerar los subárboles correspondientes a los análisis parciales de fragmentos válidos de la frase. El posterior estudio de estos subárboles puede ser utilizado, por ejemplo, para completar la gramática, generando automáticamente las reglas sintácticas necesarias para analizar la frase. Éste es precisamente el objetivo más ambicioso del análisis sintáctico robusto. En nuestro caso particular, resulta de especial interés la consideración de las etiquetas de las palabras de dichos subárboles como información adicional de apoyo para las técnicas tradicionales de etiquetación. La estrategia consiste en combinar esas subsecuencias de etiquetas para generar varias etiquetaciones completas posibles de la frase en cuestión, y posteriormente aplicar un filtro estadístico para elegir la secuencia global más probable.

Para abordar este objetivo, es imprescindible la disponibilidad de una gramática o de un banco de árboles[19] del que poder extraerla. No habiendo sido posible contar con este tipo de recursos para el español, la experimentación desarrollada en este punto se ha realizado únicamente para el idioma inglés.

Respecto al gallego, como puede deducirse de la trayectoria investigadora descrita anteriormente, es cierto que estamos desarrollando una línea de trabajo paralela especialmente dedicada a esta lengua, pero es cierto también que esta línea ha evolucionado más lentamente debido a la total inexistencia de trabajos previos y a que el idioma es intrínsecamente más complejo, sobre todo a nivel morfológico. Por esta razón, es muy posible que en la presente memoria no hagamos ninguna referencia más al idioma de la comunidad autónoma en la que nos encontramos. En este punto es importante señalar que comienzan a aparecer los primeros trabajos realizados sobre esta lengua [Vilares *et al.* 1998], por lo que esperamos que esta labor pionera fructifique pronto, de manera que en un futuro muy próximo estén disponibles los que quizás a la postre podrían ser los primeros recursos lingüísticos libremente disponibles para todas aquellas personas que estén interesadas en realizar labores de procesamiento automático del idioma gallego.

1.6 Estructura de la presente memoria

Los contenidos fundamentales de este trabajo se dividen en tres partes claramente diferenciadas. A continuación introducimos cada una de ellas, centrándonos en el contenido de los capítulos que las conforman:

- La parte I engloba los conceptos preliminares sobre NLP en general, y sobre etiquetación en particular, necesarios para abordar el resto de los contenidos del documento. Además de la presente introducción, esta parte consta también de los siguientes capítulos:
  - El capítulo 2 describe los recursos lingüísticos que han sido utilizados en este trabajo. Dichos recursos pueden presentarse bajo la forma de diccionarios, textos etiquetados o bancos de árboles, y constituyen el principio de funcionamiento de la lingüística basada en corpus, ya que a partir de ellos es posible ajustar el comportamiento de las herramientas de etiquetación. A través de este proceso de ajuste, también denominado entrenamiento, los etiquetadores adquieren el conocimiento relativo a los fenómenos lingüísticos que aparecen presentes en el lenguaje, y que posteriormente utilizarán durante el procesamiento de nuevos textos. Se trata de un capítulo

[19] También denominado *treebank*.

puramente descriptivo que enumera las características de los recursos considerados, aunque se detalla también la manera en la que se ha hecho uso de ellos.

- El capítulo 3 esboza las distintas técnicas existentes para la realización del análisis léxico de los textos. Este análisis transforma los caracteres de entrada en unidades de más alto nivel de significado, normalmente las palabras, y obtiene rápida y cómodamente todas las etiquetas candidatas de esas palabras. Por tanto, el análisis léxico constituye un paso de procesamiento previo que simplifica tareas posteriores tales como la etiquetación o el análisis sintáctico.

- La parte II describe en profundidad tres de las principales estrategias que definen el estado actual del arte en lo que al proceso de etiquetación se refiere. Se trata de modelos ampliamente utilizados y referenciados, pero las razones para haber elegido éstos y no otros se apoyan también en el hecho de que ha sido posible encontrar y disponer de sistemas ejecutables concretos basados en dichos modelos. Todos ellos han sido evaluados sobre el mismo conjunto de textos, lo cual ha permitido realizar un estudio comparativo muy exhaustivo. Los capítulos que conforman esta parte son los siguientes:

  - El capítulo 4 se ocupa de los etiquetadores basados en modelos de Markov ocultos o HMM,s[20]. Los HMM,s presentan un conjunto de estados que normalmente coincide con el conjunto de etiquetas que se está considerando. La dependencia probabilística del estado o etiqueta actual generalmente se trunca para considerar sólo uno o dos de los estados o etiquetas precedentes (propiedad del horizonte limitado) y esta dependencia no varía a lo largo del tiempo (propiedad del tiempo estacionario). Esto quiere decir que si, por ejemplo, se ha establecido en 0,2 la probabilidad de que un verbo venga después de un pronombre al principio de una frase, dicha probabilidad es la misma para el resto de la frase e incluso para otras frases posteriores. Como ocurre con la mayoría de los modelos probabilísticos, las dos propiedades de Markov sólo aproximan la realidad, pero no la formalizan a la perfección. Por ejemplo, la propiedad del horizonte limitado no modeliza bien estructuras recursivas, y la propiedad del tiempo estacionario no modeliza bien relaciones de larga distancia como las de los pronombres relativos. De hecho, ésta es la base del famoso argumento de Chomsky contra el uso de los modelos de Markov para el procesamiento del lenguaje natural [Chomsky 1957]. Esta crítica provocó, al menos parcialmente, que el uso de los modelos de Markov se abandonara a principios de los años sesenta, aunque quizás la falta de datos de entrenamiento y de los recursos computacionales necesarios para realizar aproximaciones empíricas a los lenguajes naturales jugó también su papel. El argumento de Chomsky está todavía vigente, pero lo que sí ha cambiado es el hecho de que los métodos que permiten obtener resultados técnicos de calidad para resolver tareas particulares se convierten en aceptables, incluso aunque no estén totalmente fundamentados en una teoría que explique completamente el lenguaje como un fenómeno cognitivo. Las herramientas representativas de este paradigma de etiquetación que han sido elegidas para nuestro estudio son el sistema TnT[21] [Brants 1996, Brants 2000] y el etiquetador del sistema Galena[22] [Vilares *et al.* 1995, Sacristán y Graña 1999].

  - El capítulo 5 describe el sistema de etiquetación presentado por Eric Brill, que utiliza reglas de transformación tanto léxicas como contextuales [Brill 1993b]. Las reglas no

[20] *Hidden Markov Models.*
[21] *Trigrams'and'Tags.*
[22] Generador de Analizadores para LEnguajes NAturales.

se escriben a mano, sino que se generan automáticamente a partir de un corpus de entrenamiento mediante un procedimiento iterativo que, en cada etapa, selecciona las transformaciones que minimizan el número de errores cometidos. El procedimiento se repite hasta que dicho número cae por debajo de un cierto umbral. Este método de entrenamiento se denomina *aprendizaje basado en transformaciones y dirigido por el error*[23]. El formato de las reglas es extremadamente sencillo, pero permite especificar relaciones muy complejas entre palabras y etiquetas, y representar así un conjunto de propiedades del lenguaje más rico que en el caso de los etiquetadores puramente estocásticos. Por ejemplo, se puede hacer referencia no sólo al contexto precedente de una palabra dada, sino también al contexto posterior. Sin embargo, el etiquetador de Brill no proporciona información sobre la distribución de probabilidad de las etiquetas y las palabras, y por tanto su uso como componente probabilístico de un sistema mayor podría ser cuestionable.

- El capítulo 6 presenta otro modelo estadístico para la etiquetación, que combina las ventajas de los métodos anteriores, y que se puede clasificar como un modelo de Máxima Entropía: el sistema JMX [Ratnaparkhi 1996]. La entropía es una función de estado que permite medir y comparar entre sí modelos basados en distribuciones de probabilidad. Así pues, este sistema utiliza un texto de entrenamiento para construir un modelo de referencia, y para medir la diferencia existente entre éste y cualquier otro modelo generado *a posteriori* por el proceso de etiquetación. Es decir, la base de esta aproximación es construir modelos a partir de los nuevos textos de entrada, evaluarlos mediante una función de entropía, y ajustarlos hasta obtener la proximidad máxima respecto al modelo de referencia.
- El capítulo 7 cierra esta parte discutiendo los aspectos fundamentales que es necesario tener en cuenta para realizar un estudio comparativo de los diferentes sistemas de etiquetación que hemos considerado. Se describen, por un lado, las estrategias seguidas a la hora de realizar los experimentos, y por otro, cuáles son exactamente los índices de rendimiento que han sido calculados. En definitiva, se presenta una metodología de evaluación completa [Graña y Rajman 1999], que permite decidir si un etiquetador es mejor o peor que otro.

- La parte III se centra en la generación automática de secuencias de etiquetas mediante la sintaxis, y en cómo se puede hacer uso de dichas secuencias para mejorar el rendimiento de los sistemas de etiquetación. Los capítulos que conforman esta parte son los siguientes:

  - El capítulo 8 presenta los conceptos fundamentales del *análisis sintáctico probabilístico*, también denominado *análisis sintáctico estocástico*. La única forma de sintaxis permitida por los métodos de etiquetación descritos en la parte anterior es la simple consideración del orden secuencial de aparición de las palabras y de las etiquetas dentro de la frase. En este capítulo intentamos huir de esta visión lineal, introducir otras nociones de gramática más complejas, y comenzar a explorar su aplicación al proceso de etiquetación. Después de introducir formalmente el concepto de *gramática independiente del contexto estocástica*, se estudia el problema del análisis sintáctico para este tipo de gramáticas. En el presente trabajo, no pretendemos realizar una cobertura exhaustiva de todas las técnicas de análisis sintáctico, tal y como hicimos con las técnicas de etiquetación. En lugar de esto, el capítulo presenta un sencillo e intuitivo mecanismo de análisis sintáctico, el algoritmo CYK [Kasami 1965, Younger 1967], que progresivamente

[23] *Transformation-based error-driven learning.*

se va mejorando hasta adecuarlo totalmente a nuestras necesidades. Las mejoras introducidas en este algoritmo básico serán su adaptación para el funcionamiento extendido sobre gramáticas independientes del contexto arbitrarias [Erbach 1994], su integración dentro del marco estocástico [Chappelier y Rajman 1998], y la discusión de consideraciones de paralelismo capaces de reducir su complejidad temporal [Barcala y Graña 1999]. Por último, al igual que los etiquetadores tienen que enfrentarse al problema de las palabras desconocidas, es decir, al problema de los diccionarios incompletos, los analizadores sintácticos deben saber enfrentarse al problema de las gramáticas incompletas. Por tanto, nuestro objetivo aquí es el de dejar preparado un marco de análisis sintáctico estocástico que permita experimentar cómodamente con técnicas de análisis sintáctico robusto orientadas específicamente al problema de la etiquetación. Dichas técnicas se abordan en el siguiente capítulo.

- El capítulo 9 estudia el uso combinado de los sistemas de etiquetación tradicionales con información sintáctica procedente de una gramática. La manera de enfrentarse con garantías al pequeño porcentaje de errores que de manera inherente presentan los sistemas de etiquetación puramente estocásticos pasa inevitablemente por incorporar alguna fuente de información de más alto nivel, como puede ser una gramática. Por supuesto, resulta muy difícil disponer de una gramática que genere todas y cada una de las estructuras que aparecen en las frases de un idioma dado. Es aquí donde adquieren relevancia las técnicas de análisis sintáctico robusto. Para cada frase no completamente analizable, se recuperan las etiquetas que los subárboles de análisis asignan a las palabras. Posteriormente, se diseñan estrategias similares a las utilizadas en el análisis sintáctico robusto para combinar dichas subsecuencias de etiquetación parcial, y construir así secuencias de cobertura total [Graña *et al.* 1999]. La etiquetación final se obtiene aplicando un filtro estadístico sobre dichas secuencias. Por último, se evalúa el rendimiento de estas nuevas técnicas de etiquetación y se compara con los de las técnicas tradicionales.
- El capítulo 10 finaliza este estudio presentando las principales conclusiones extraídas y las líneas de trabajo futuro.

Una última parte, la parte IV, engloba los apéndices que contienen los aspectos complementarios, así como la lista de las referencias bibliográficas utilizadas a lo largo de este trabajo y el índice de materias.

1.7 Otras aproximaciones a la etiquetación y otros idiomas

La etiquetación ha sido una de las áreas de mayor actividad dentro de la investigación en NLP durante los últimos diez años. En el presente trabajo sólo hemos cubierto las aproximaciones más importantes, pero existen muchas otras técnicas que han sido aplicadas también a la etiquetación: las redes neuronales [Benello *et al.* 1989, Marques y Pereira 1996], los árboles de decisión [Schmid 1994], y el aprendizaje basado en memoria, o método de los k vecinos más cercanos [Daelemans *et al.* 1996, Zavrel y Daelemans 1999], son algunas de ellas.

Otra de las técnicas de etiquetación de mejor rendimiento para el inglés es un formalismo reduccionista no cuantitativo denominado ENGCG[24] o gramática de restricciones para el inglés, desarrollado en la Universidad de Helsinki [Voutilainen y Heikkilä 1994, Tapanainen y Voutilainen 1994, Samuelsson y Voutilainen 1997]. En ENGCG, existe un conjunto de reglas escritas a mano que manejan contexto global o, en la mayoría de las veces,

[24] *English Constraint Grammar.*

contexto local. No existe por tanto una verdadera noción de gramática formal, sino más bien un conjunto de restricciones, casi siempre negativas, que van eliminando los análisis imposibles en el contexto [Karlsson *et al.* 1995, Samuelsson *et al.* 1996]. Dicho conjunto de reglas se compila bajo la forma de un autómata de estado finito para su rápida aplicación. La idea básica es similar al aprendizaje basado en transformaciones, excepto por el hecho de que es un humano, y no un algoritmo, el que modifica iterativamente el conjunto de reglas de etiquetación para minimizar el número de errores. En cada iteración, el conjunto de reglas se aplica al corpus y posteriormente se intentan modificar dichas reglas de manera que los errores más importantes queden manualmente corregidos. En definitiva, esta metodología propone la construcción de un pequeño sistema experto para la etiquetación, y parece ofrecer mejores rendimientos que los etiquetadores basados en modelos de Markov ocultos, especialmente cuando los *corpora* de entrenamiento y de aplicación no provienen de la misma fuente. Los resultados pueden llegar al 99% de precisión con un conjunto de unas 1.000 reglas de restricción. No obstante, la comparación de estos dos modelos es difícil de realizar, ya que cuando ENGCG no es capaz de resolver determinadas ambigüedades devuelve un conjunto de más de una etiqueta. Si alguien está ya familiarizado con la metodología, la construcción de este tipo de etiquetadores no requiere más esfuerzo que la construcción de un etiquetador basado en un HMM [Chanod y Tapanainen 1995], aunque quizás se puede argumentar que la metodología de los HMM,s es automática y más accesible.

Actualmente, la Real Academia Española está desarrollando también un formalismo de reglas de restricción denominado RTAG, para la anotación automática de los *corpora* CORDE[25] y CREA[26] [Porta 1996, Sánchez *et al.* 1999]. Este sistema aplica gramáticas de reglas de contexto ponderadas sobre textos anotados ambiguamente. Esto quiere decir que cuando un contexto satisface la descripción estructural de una regla recibe la puntuación que indica la regla. Esta puntuación puede ser positiva, para promover lecturas, o negativa, para penalizarlas. Finalizado el proceso, permanecen las lecturas con mayor puntuación siempre que aventajen a otra u otras por encima de un umbral definido previamente. El sistema intenta también eliminar lecturas imposibles en función del contexto, sin pérdida de lecturas posibles aunque improbables en ocasiones. Para esta poda de lecturas imposibles en función del contexto se utilizan, básicamente, tres tipos de información: información derivada del propio texto (es decir, de sus características estructurales, tipográficas o secuenciales), información gramatical local (fundamentalmente concordancia y restricciones de aparición conjunta) e información gramatical estructural (toma de decisiones con ayuda de la información estructural derivable de la secuencia lineal del texto).

Otro etiquetador basado en gramáticas de restricciones que ha sido utilizado también con éxito sobre el español es el sistema RELAX [Padró 1996]. En este sistema, las reglas de restricción pueden ser tanto escritas de forma manual, como generadas automáticamente mediante un algoritmo de adquisición basado en un árbol de decisión [Màrquez y Padró 1997, Màrquez y Rodríguez 1997].

Ha habido también trabajos orientados hacia cómo construir un corpus etiquetado con la mínima cantidad de esfuerzo humano [Brill *et al.* 1990]. Esto es en sí mismo un problema cuando se trabaja sobre un idioma para el cual no existe ningún corpus de entrenamiento ya etiquetado, o cuando aún disponiendo de un corpus etiquetado las aplicaciones trabajan con textos tan diferentes que esos datos de entrenamiento carecen casi totalmente de utilidad. Algunos autores han explorado también la manera de construir un conjunto de etiquetas automáticamente, con el fin de crear categorías sintácticas apropiadas para un idioma o para un estilo de texto particular [McMahon y Smith 1996].

En el presente trabajo hemos cubierto sólo la etiquetación de textos en español y en inglés. Este último se muestra como un idioma que parece ser particularmente adecuado para el uso de

[25] Corpus Diacrónico del Español.
[26] Corpus de Referencia del Español Actual.

métodos que intentan inferir las categorías gramaticales a partir de la posición de las palabras dentro de la frase. En otros idiomas, el orden de las palabras es mucho más libre, y por tanto el contexto contribuye al proceso de etiquetación con mucha menos información. Sin embargo, en la mayoría de esos lenguajes, incluido el español, los modelos de inflexión son mucho más ricos que en el caso del inglés y ésa sí es una información de gran utilidad.

Una evaluación completa de los etiquetadores en su papel de preprocesadores de alto nivel para aplicaciones de NLP multilingua será posible sólo cuando esté disponible una gran cantidad de resultados experimentales sobre un amplio conjunto de idiomas. A pesar de esto, existen actualmente bastantes estudios, al menos para las lenguas europeas. Dichos estudios sugieren que el rendimiento de los etiquetadores sobre esos idiomas es similar al rendimiento sobre el inglés [Dermatas y Kokkinakis 1995], aunque muchas veces es difícil realizar este tipo de comparaciones debido a la frecuente incompatibilidad de los conjuntos de etiquetas. A pesar de los esfuerzos realizados para la provisión de estándares de etiquetación, dichos conjuntos no son universales. Precisamente intentan codificar las categorías funcionales particulares de cada idioma individual. En cualquier caso, nuestro trabajo pretende ser una aportación más que proporcione cifras concretas del comportamiento de los etiquetadores para el caso del español.

Capítulo 2

Recursos lingüísticos

Este capítulo está orientado a detallar los recursos lingüísticos que se han utilizado en el presente trabajo, la forma en la que éstos se presentan, y la manera en la que hemos hecho uso de ellos. Cada uno de esos recursos está asociado a un lenguaje natural concreto, es decir, a un idioma concreto. Los idiomas con los que más extensamente hemos trabajado han sido principalmente dos:

- El español, por ser de una utilidad práctica palpable. No sólo es el idioma más hablado en todo el territorio de nuestro país, sino también uno de los más hablados en todo el mundo.
- El inglés, por ser quizás el idioma que más ampliamente ha sido considerado por toda la comunidad científica internacional y, por tanto, por ser el idioma al que pertenecen los recursos lingüísticos de mayor calidad que se pueden encontrar.

Básicamente, consideraremos tres tipos de recursos:

- Textos etiquetados. Un texto o corpus etiquetado es aquél en el que todas y cada una de las palabras aparecen acompañadas al menos de su etiqueta correcta, es decir, aquélla que define correctamente el papel que dicha palabra está jugando en la frase concreta en la que aparece, y quizás también de su lema.

  El lema, o forma canónica del concepto al que hace referencia la palabra, es normalmente la forma del masculino singular, en el caso de sustantivos, adjetivos, etc., y el infinitivo en el caso de los verbos, es decir, lo que en definitiva constituye la entrada que podemos encontrar en cualquier diccionario semántico a la hora de buscar las acepciones o significados de un término dado.

  La principal aplicación de los *corpora* etiquetados es la de servir como referencia del comportamiento de un idioma, o al menos de un estilo de uso del mismo, durante el proceso de ajuste o puesta a punto de las herramientas destinadas a realizar cualquier tipo de procesamiento de nuevos textos en ese idioma, en particular su etiquetación.

- Diccionarios. En nuestro contexto, un diccionario, a veces diremos también *lexicón*, es simplemente una lista de palabras acompañada de sus posibles etiquetas o papeles candidatos que puede desempeñar dentro de la frase. Uno de nuestros objetivos concretos es comprobar de qué manera el uso de un diccionario externo puede ayudar a incrementar el rendimiento del proceso de etiquetación de textos en lenguaje natural.

- Bancos de árboles. Los bancos de árboles son textos donde no sólo cada palabra está acompañada de su etiqueta correcta, sino que además cada frase aparece completamente analizada sintácticamente. Los árboles de análisis con todos sus nodos se presentan en

línea de manera parentizada. Veremos cómo a partir de recursos de este tipo es posible extraer las reglas de la gramática a la que obedece un determinado idioma, o al menos un subconjunto relevante de él, incluso con una probabilidad de uso asignada a cada una de esas reglas.

Es obvio que un recurso de estas características presenta un nivel de anotación muy superior al de los textos simplemente etiquetados. Es por esta razón que intuimos que el uso de esta nueva información que aparece en él podría constituir una gran ayuda para el proceso de etiquetación, siendo quizás éste el principal objetivo que persigue el presente trabajo. Es evidente también que estos recursos requieren un esfuerzo de diseño mucho mayor, pero también es cierto que cada vez será más frecuente el poder encontrarlos disponibles, lo que nos lleva a pensar que el futuro de las técnicas que vamos a proponer no se verá comprometido.

A continuación presentamos en detalle todos y cada uno de los recursos lingüísticos que serán utilizados como fuentes de información para la formalización y descripción del comportamiento y características de los idiomas tratados.

2.1 El corpus ITU

Los recursos lingüísticos de calidad para el idioma español no son todavía demasiado abundantes. Uno de los textos ya etiquetados y libremente disponibles es el *International Telecommunications Union CCITT Handbook*, también conocido como *The Blue Book* en el ambiente de las telecomunicaciones, y como *Corpus ITU* en el ambiente lingüístico. Este corpus es la principal colección de textos sobre telecomunicaciones existente y, debido a su gran tamaño, constituye un excelente marco de pruebas para el estudio del comportamiento de los sistemas de etiquetación.

El texto original no etiquetado tiene alrededor de 5 millones de palabras. En lo que se refiere al texto ya anotado, existen dos versiones: el corpus entero etiquetado con la versión para español del etiquetador de XEROX [Cutting *et al.* 1992, Sánchez y Nieto 1995a, Sánchez y Nieto 1995b], y un subcorpus de aproximadamente medio millón de palabras corregido a mano por la Universidad de Lancaster. Esta segunda versión es la que se ha utilizado como uno de los *corpora* de referencia del presente trabajo, aunque ambos se pueden descargar desde las URL,s[1] correspondientes a las páginas que soportan toda la información relativa al proyecto CRATER: *Corpus Resources And Terminology ExtRaction* [CRATER 1993].

A continuación describimos detalladamente las principales características de nuestro corpus de referencia:

- El corpus tiene 486.073 palabras. El tamaño del fichero es de 7.564.781 caracteres. La sintaxis de cada línea es de la forma

  ```
  palabra/etiqueta/lema  palabra/etiqueta/lema ... palabra/etiqueta/lema
  ```

  Cada una de las líneas del fichero es una frase. El corpus tiene 14.919 frases, es decir, un número medio de 33 palabras por frase.

- El corpus contiene 45.679 formas verbales, donde 581 formas están en voz pasiva, 870 son formas verbales compuestas, y 3.415 son formas verbales con un pronombre enclítico.

- Las características del lexicón o diccionario constituido por todas las formas que aparecen en el corpus son las siguientes:

[1] *Uniform Resource Locator* (dirección de un recurso de Internet).

15.745 formas con 1 etiqueta, 1.097 formas con 2 etiquetas,
223 formas con 3 etiquetas, 61 formas con 4 etiquetas,
8 formas con 5 etiquetas, 3 formas con 6 etiquetas y
1 forma con 7 etiquetas.

Esto es, 17.138 formas diferentes, con 18.917 etiquetas posibles, y correspondientes a 12.462 lemas diferentes. Si calculamos el porcentaje de formas ambiguas y el número medio de etiquetas por forma, obtenemos:

$$\%\text{ formas ambiguas} = \frac{\#\text{ formas ambiguas}}{\#\text{ formas}} \times 100 = \frac{17.138 - 15.745}{17.138} \times 100 = 8,13\,\%$$

$$\#\text{ medio de etiquetas por forma} = \frac{\#\text{ etiquetas}}{\#\text{ formas}} = \frac{18.917}{17.138} = 1,10 \text{ etiquetas por forma.}$$

- Mucho más interesante es calcular las mismas características directamente con todas las palabras del corpus, y obtenemos las siguientes cifras:

  263.592 palabras con 1 etiqueta, 107.746 palabras con 2 etiquetas,
  79.751 palabras con 3 etiquetas, 7.013 palabras con 4 etiquetas,
  18.949 palabras con 5 etiquetas, 8.528 palabras con 6 etiquetas y
  494 palabras con 7 etiquetas.

  Esto es, 486.073 palabras, con 895.760 etiquetas posibles. Si calculamos de nuevo el porcentaje de palabras ambiguas y el número medio de etiquetas por palabra, obtenemos:

$$\%\text{ palabras ambiguas} = \frac{\#\text{ palabras ambiguas}}{\#\text{ palabras}} \times 100 = \frac{486.073 - 263.592}{486.073} \times 100 = 45,77\,\%$$

$$\#\text{ medio de etiquetas por palabra} = \frac{\#\text{ etiquetas}}{\#\text{ palabras}} = \frac{895.760}{486.073} = 1,84 \text{ etiquetas por palabra.}$$

Ejemplo 2.1 A continuación mostramos una línea tomada directamente del corpus ITU, para ilustrar el aspecto general que presentan los datos:

```
En/PREP/en esta/DMPXFS/este colaboraci&oacute;n/NCFS/colaboraci&oacute;n
debe/VMPI3S/deber reconocerse/VCLI/reconocer el/ARTDMS/el
car&aacute;cter/NCMS/car&aacute;cter consultivo/ADJGMS/consultivo de/PREP/de
las/ARTDFP/el organizaciones/NCFP/organizaci&oacute;n que/CQUE/que
participan/VLPI3P/participar en/PREP/en los/ARTDMP/el trabajos/NCMP/trabajo
del/PDEL/del CCITT/NPTOS/CCITT ,/CM/, en_particular/ADVN/en_particular
a/PREP/a la/ARTDFS/el ISO/ACRNM/ISO ,/CM/, desde/PREP/desde el/ARTDMS/el
punto/NCMS/punto de/PREP/de vista/NCFS/vista de/PREP/de su/PPOSPS/suyo
labor/NCFS/labor con_respecto_a/PREP/con_respecto_a los/ARTDMP/el
sistemas/NCMP/sistema de/PREP/de datos/NCMP/dato y/CC/y a/PREP/a
las/ARTDFP/el comunicaciones/NCFP/comunicaci&oacute;n ./FS/.
```

□

Las etiquetas que aparecen a lo largo de este corpus pertenecen al juego de etiquetas desarrollado en el marco del proyecto CRATER, el cual obedece al estándar EAGLES[2]. La descripción de cada una de ellas se puede ver en la sección A.1.

2.2 El sistema GALENA

Tal y como ya se ha explicado en la introducción de la presente memoria, uno de los objetivos que perseguimos en este trabajo es la evaluación y comparación de distintos sistemas de etiquetación, y el estudio de sus comportamientos concretos cuando manejan textos en español. Entre esos sistemas a comparar figura nuestra propia herramienta: el sistema GALENA [Vilares *et al.* 1995]. Durante las fases de diseño e implementación del analizador léxico de dicha herramienta, se ha desarrollado un lexicón de tamaño medio para el español [Graña *et al.* 1994], el cual nos ha parecido interesante integrar en los distintos sistemas de etiquetación a evaluar, siempre que esto ha sido posible, para observar en qué medida un diccionario externo puede ayudar a mejorar el proceso de etiquetación en cada uno de los sistemas.

En este punto, para poder llevar a cabo correctamente esa integración, surge el problema de que el corpus ITU y el diccionario del sistema GALENA no utilizan el mismo juego de etiquetas. Para solventar esta dificultad es necesario establecer primero una correspondencia entre ambos juegos de etiquetas, el del proyecto CRATER y el del sistema GALENA, y después existen dos posibilidades:

1. Transformar las etiquetas de todas las palabras del corpus ITU en etiquetas del sistema GALENA.

2. Transformar las etiquetas de todas las palabras del diccionario del sistema GALENA en etiquetas CRATER.

Hemos optado por la primera de las posibilidades, por ser la más sencilla y por ser la que en principio no altera los recursos que nosotros mismos hemos generado y que en el fondo son los que más nos interesa contrastar.

Por tanto, el juego de etiquetas del sistema GALENA es el que ha sido utilizado en todos los experimentos relativos al idioma español que se han realizado en el presente trabajo. Así pues, las siguientes secciones describen detalladamente las características de los dos principales recursos lingüísticos aportados por el sistema GALENA: su juego de etiquetas y su diccionario.

2.2.1 El juego de etiquetas del sistema GALENA

La tabla 2.1 es una representación compacta del juego de etiquetas del sistema GALENA. Los campos están en la fila en negrita. Los valores de `Categoría` y `Tipo` están hacia abajo. Los valores para el resto de los campos están hacia arriba. Los cruces en el resto de la tabla muestran los valores permitidos para cada caso. Por ejemplo, `Grado` no tiene sentido para un `Verbo`, y sólo `singular` y `plural` están permitidos para el `Número persona` de un `Posesivo`. El género tiene sentido en los verbos cuando se trata de una forma verbal en participio y, en todo caso, dicha marca de género es el último carácter de las etiquetas de los verbos.

[2] *Expert Advisory Group on Language Engineering Standards* (EAGLES) es una iniciativa de la Comisión Europea iniciada en 1993 dentro del marco del programa de Investigación e Ingeniería Lingüística (LRE), y su objetivo es acelerar la provisión de estándares para la construcción de recursos lingüísticos a gran escala (*corpora* de textos y de habla, léxicos informatizados, etc.), así como para la manipulación de este conocimiento (formalismos lingüísticos, lenguajes de marcado y herramientas informáticas), y establecer mecanismos de evaluación de los recursos, herramientas y productos [Calzolari y McNaught 1996].

Categoría	Tipo	Subtipo	Género	Número	Grado	Persona	Número persona	Caso	Tiempo verbal	Modo
Sustantivo (S)	Común (c)		m f y	s p y						
	Propio (p)		m f y	s p y						
Adjetivo (A)			m f y	s p y	c 0					
Demostrativo (E)		n y	m f n y	s p y						
Relativo (T)		n d y	m f n y	s p y						
Indefinido (I)		n d y	m f n y	s p y						
Interrogativo (G)		n y	m f n y	s p y						
Posesivo (M)		n d y	m f n y	s p y		1 2 3	s p			
Numeral (N)	Cardinal (c)	n d y	m f y	s p						
	Ordinal (o)	n d y	m f y	s p						
	Partitivo (p)	n	m f y	s p						
	Múltiplo (m)	n	m f y	s p						
Artículo (D)			m f n	s p						
Pronombre Personal (R)	Tónico (t)		m f y (último)			1 2 3	s p y	n a d y p q		
	Proclítico átono (p)		m f y (último)			1 2 3	s p y	n a d y p q		
	Enclítico átono (e)		m f y (último)			1 2 3	s p y	n a d y p q		
Verbo (V)			m f 0 (último)			1 2 3 y 0	s p 0		p i s e f c 0 P 1 S E F C	i s m f F g G p
Preposición (P)										
Conjunción (C)	Coordinada (c)									
	Subordinada (s)									
Adverbio (W)	Nuclear (n)									
	Modificador (m)									
	Nuclear y modificador (y)									
	Relativo (r)									
	Exclamativo (i)									
	Interrogativo (v)									
Interjección (Y)										
Marca Puntuación (Q)	, . : ; ¿ ? ¡ ! ( ) - " ...									
Periféricos (Z)	Palabra extranjera (e)		m f y 0	s p y 0						
	Fórmula (f)		m f y 0	s p y 0						
	Abreviatura (a)		m f y 0	s p y 0						
	Símbolo (s)		m f y 0	s p y 0						
	Sigla (g)		m f y 0	s p y 0						
	Otros (o)		m f y 0	s p y 0						

Tabla 2.1: Juego de etiquetas del sistema GALENA

Ejemplo 2.2 Las tres etiquetas de la palabra `sobre` serían: `Scms` (sustantivo, común, masculino, singular), `P` (preposición), y `Vysps0` (verbo, primera y tercera personas, singular, presente, subjuntivo, género no aplicable). □

Esta tabla genera un conjunto de 1.048 etiquetas, aunque no todas ellas representan combinaciones correctas desde un punto de vista lingüístico. Por otro lado, para el presente estudio, hemos decidido extender el conjunto de etiquetas del sistema GALENA con el fin de marcar explícitamente las formas verbales compuestas, las formas verbales en voz pasiva y las formas verbales con pronombre enclíticos. Por tanto, el conjunto final de etiquetas que hemos utilizado consta de las 373 etiquetas que aparecen en la sección A.2.

A continuación describimos el procedimiento que se ha seguido para extender la marcación original de las formas verbales en el sistema GALENA. Básicamente, si `<etiqueta>` es la etiqueta que aparecía originalmente, la nueva marcación consiste en añadir un sufijo a esa `<etiqueta>`. En el caso de las formas verbales compuestas, tales como `he comido`, aparecen dos componentes:

1. El verbo auxiliar (`he`), que será marcado como `<etiqueta>TC1`.
2. El participio (`comido`), que será marcado como `<etiqueta>TC2`.

Para las formas verbales en voz pasiva, tales como `fue comido`, aparecen dos componentes:

1. El verbo auxiliar (`fue`), que será marcado como `<etiqueta>TCP1`.
2. El participio (`comido`), que será marcado como `<etiqueta>TCP2`.

O tres, en casos como `ha sido comido`:

1. El primer verbo auxiliar (`ha`), que será marcado como `<etiqueta>TCP1`.
2. El segundo verbo auxiliar (`sido`), que será marcado como `<etiqueta>TCP2`.
3. El participio (`comido`), que será marcado como `<etiqueta>TCP3`.

Hasta aquí, este procedimiento presenta las siguientes implicaciones:

- Cada una de las formas del verbo `haber` puede tener tres etiquetaciones diferentes:
  - `<etiqueta>`, cuando va solo.
  - `<etiqueta>TC1`, cuando es la primera parte de un tiempo compuesto.
  - `<etiqueta>TCP1`, cuando es la primera parte de una voz pasiva.
- Cada una de las formas del verbo `ser` puede tener dos etiquetaciones diferentes:
  - `<etiqueta>`, cuando va solo.
  - `<etiqueta>TCP1`, cuando es la primera parte de una voz pasiva.
- Cada participio verbal puede tener hasta cinco etiquetaciones diferentes:
  - `VOsOpm`, cuando es participio.
  - `VOsOpmTC2`, cuando es la segunda parte de un tiempo compuesto.
  - `VOsOpmTCP2`, cuando es la segunda parte de una voz pasiva.
  - `VOsOpmTCP3`, cuando es la tercera parte de una voz pasiva.
  - `AmsO`, cuando es adjetivo.

Ejemplo 2.3 Las cinco etiquetaciones anteriores son las del participio masculino singular. Más exactamente, por ejemplo para el participio del verbo **ver**, tenemos todas estas formas y etiquetaciones posibles:

```
visto V0s0pm V0s0pmTC2 V0s0pmTCP2 V0s0pmTCP3 Ams0
vista V0s0pf V0s0pfTCP2 V0s0pfTCP3 Afs0
vistos V0p0pm V0p0pmTCP2 V0p0pmTCP3 Amp0
vistas V0p0pf V0p0pfTCP2 V0p0pfTCP3 Afp0
```
□

Respecto a las formas verbales con pronombres enclíticos, tales como **tenerlo** o **verse**, en español pueden tener hasta tres de esos pronombres, como en **tráetemelo**. Sin embargo, el corpus ITU presenta formas con sólo un pronombre enclítico, que serán marcadas como `<etiqueta>PE1`.

Por último, sólo nos falta indicar cómo es exactamente la correspondencia entre las etiquetas CRATER y las etiquetas GALENA. El juego de etiquetas del proyecto CRATER resulta ser un poco más preciso que el del sistema GALENA, debido a que tiene un cardinal superior: 475 etiquetas frente a 373. A pesar de esto, dicha correspondencia se puede establecer sin problema alguno, tal y como se especifica en la sección A.3. La aplicación de esta correspondencia al corpus ITU nos ha proporcionado un corpus de referencia para nuestro experimento completamente etiquetado con el juego de etiquetas del sistema GALENA.

Ejemplo 2.4 A continuación reproducimos de nuevo la misma porción de texto que habíamos utilizado en el ejemplo 2.1 para ilustrar el aspecto general del corpus, pero esta vez ya con las etiquetas definitivas:

```
En/P/en esta/Eyfs/este colaboración/Scfs/colaboración debe/V3spi0/deber
reconocerse/V000f0PE1/reconocer el/Dms/el carácter/Scms/carácter
consultivo/Ams0/consultivo de/P/de las/Dfp/el
organizaciones/Scfp/organización que/Cs/que participan/V3ppi0/participar
en/P/en los/Dmp/el trabajos/Scmp/trabajo de/P/de el/Dms/el CCITT/Spys/CCITT
,/Q,/, en_particular/Wy/en_particular a/P/a la/Dfs/el ISO/Zgfs/ISO ,/Q,/,
desde/P/desde el/Dms/el punto/Scms/punto de/P/de vista/Scfs/vista de/P/de
su/Mdys3s/suyo labor/Scfs/labor con_respecto_a/P/con_respecto_a los/Dmp/el
sistemas/Scmp/sistema de/P/de datos/Scmp/dato y/Cc/y a/P/a las/Dfp/el
comunicaciones/Scfp/comunicación ./Q./.
```
□

Como se puede ver, no sólo se han efectuado cambios al nivel de las etiquetas, sino también al nivel de las palabras y los lemas[3], para adaptarlas y hacerlas totalmente compatibles con las entradas del diccionario del sistema GALENA, el cual se describe a continuación.

2.2.2 El diccionario del sistema GALENA

Como hemos visto anteriormente, el segundo de los recursos lingüísticos importantes que aporta el sistema GALENA es su diccionario, un lexicón de tamaño medio desarrollado durante las fases de diseño e implementación del analizador léxico de dicho sistema.

Sin embargo, antes de describir directamente este recurso, vamos a ver algunas cifras correpondientes a las características generales del sistema GALENA, en lo que al análisis léxico se refiere:

[3]Cabe destacar, por ejemplo, la presencia de acentos reales.

- La base de datos léxica contiene el siguiente número de raíces:

2.944 adjetivos,	125 adverbios,	2 artículos,	102 conjunciones,
14 demostrativos,	36 indefinidos,	40 interjecciones,	4 interrogativos,
62 numerales,	67 periféricas,	8 posesivos,	43 preposiciones,
29 pronombres,	5 relativos,	8.027 sustantivos,	5.600 verbos y
30 especiales.			

  Esto es, 17.138 raíces correspondientes a 13.667 lemas diferentes.

- Existe un proceso de compilación que transforma la base de datos léxica en una arquitectura de reconocimiento mucho más eficiente basada en un autómata finito acíclico determinista numerado[4], el cual consta de 11.985 estados y 31.258 transiciones.

- El tamaño del fichero compilado correspondiente a ese autómata finito es de 3.466.121 *bytes*.

- El tiempo de compilación es de aproximadamente 7 segundos, y la velocidad de reconocimiento de 40.000 palabras por segundo en una máquina con un procesador Pentium II a 300 MHz bajo sistema operativo Linux.

Para aquellas herramientas de etiquetación en las que sea posible la integración de un diccionario externo, y a partir de la base de datos léxica anteriormente descrita, hemos generado todas las formas reconocidas por el analizador léxico del sistema GALENA, y las características de este diccionario han resultado ser las siguientes:

265.418 formas con 1 etiqueta,	9.995 formas con 2 etiquetas,
588 formas con 3 etiquetas,	11.343 formas con 4 etiquetas,
4.097 formas con 5 etiquetas y	163 formas con 6 etiquetas.

Es decir, 291.604 formas diferentes, con 354.007 etiquetas posibles, y correspondientes, como ya hemos dicho, a 13.667 lemas diferentes.

Sin embargo, la base de datos léxica no almacena ninguna información sobre frecuencias o probabilidades, ya que esta información es relevante para las palabras, no para las raíces, y además depende de cada corpus concreto. Por tanto, si utilizamos únicamente la base de datos léxica, el diccionario generado contendrá las etiquetas candidatas de cada forma en un orden totalmente arbitrario y no resultará fiable para determinados sistemas de etiquetación, como puede ser el caso del etiquetador de Brill, el cual considera la primera etiqueta de cada palabra como la más probable [Brill 1992]. Para evitar este problema, hemos realizado un estudio de las clases de ambigüedad presentes en el corpus ITU, y hemos generado el diccionario correspondiente al sistema GALENA con las etiquetas de cada forma ordenadas según la combinación de etiquetas más frecuente en el corpus ITU para su clase de ambigüedad.

Para dar una idea aproximada de lo que este diccionario externo puede aportar, diremos que la intersección del diccionario del sistema GALENA y el diccionario constituido por todas las formas que aparecen en el corpus ITU contiene 6.594 formas, donde:

- 3.670 formas (que involucran a 166.573 palabras del corpus ITU, es decir, a un 34,27% de las palabras) están en la misma clase de ambigüedad en ambos diccionarios.

- 2.924 formas (que involucran a 216.106 palabras del corpus ITU, es decir, a un 44,46% de las palabras) están en diferentes clases de ambigüedad en ambos diccionarios.

[4] El método de construcción de este tipo autómatas se describe detalladamente en el capítulo 3.

Para no apartar demasiado las condiciones de nuestros experimentos de las que se producen realmente en la práctica, las etiquetas de todas estas formas en común, incluso de las que están en la misma clase de ambigüedad, han sido generadas con el orden obtenido al aplicar el mismo método general que hemos explicado anteriormente. Es cierto que podríamos haberlas generado con las etiquetas ordenadas exactamente como están en el diccionario constituido por todas las formas que aparecen en el corpus ITU, pero, como ya hemos esbozado anteriormente, disponer de un corpus sobre el cual se pueda realizar un estudio previo sobre el léxico o sobre las clases de ambigüedad no es lo usual.

2.3 El corpus SUSANNE

El corpus SUSANNE [Sampson 1994a] es el banco de árboles que se ha utilizado en nuestro estudio como corpus de referencia para el idioma inglés. El corpus SUSANNE se ha creado con el apoyo del *Economic and Social Research Council (UK)*, como parte del proceso de desarrollo de una taxonomía exhaustiva orientada al NLP y de un esquema para la grámatica del inglés, tanto a nivel lógico como de superficie: el esquema SUSANNE.

El esquema SUSANNE [Sampson 1994b] intenta proporcionar un método de representación de todos los aspectos de la gramática del inglés que están suficientemente bien definidos como para ser subceptibles de anotarse formalmente. El modelo ofrece un esquema de categorías y un conjunto de formas de aplicarlas que lo hacen muy práctico para los investigadores en NLP a la hora de registrar sistemáticamente y sin ambigüedades todo lo que ocurre en el uso real del lenguaje, y permite que investigadores de diferentes lugares puedan intercambiar datos gramaticales sin que surjan confusiones relacionadas con los usos y terminologías locales.

El corpus SUSANNE se ha producido casi en su totalidad de forma manual, no a través de un sistema de análisis sintáctico automático. El corpus en sí mismo consiste en un subconjunto de aproximadamente 150.000 palabras (correspondientes a unos 10.500 lemas diferentes) del corpus BROWN para el inglés americano [Francis y Kučera 1982], anotado de acuerdo con el esquema SUSANNE. Por tanto, el esquema SUSANNE para la anotación de los análisis sintácticos se ha desarrollado en base al inglés británico y al americano. No cubre otros lenguajes, aunque se espera que sus principios generales sirvan de ayuda al desarrollo de otras taxonomías comparables para ellos. El esquema está principalmente orientado al lenguaje escrito.

2.3.1 Estructura del corpus SUSANNE

El corpus SUSANNE consta de 64 ficheros, cada uno de los cuales contiene una versión anotada de un texto de unas 2.000 palabras del corpus BROWN. Los ficheros tienen un tamaño medio de unos 83 *kilobytes* y, por tanto, el corpus completo ocupa unos 5.3 *megabytes*. Los nombres de los ficheros son los mismos que los de los respectivos textos BROWN, por ejemplo `A01`, `N12`, etc. Se han analizado 16 textos de cada una de las cuatro categorías o géneros literarios BROWN siguientes:

- `A`: reportajes de prensa.
- `G`: bellas letras, biografías, memorias.
- `J`: textos eruditos, principalmente científicos y técnicos.
- `N`: aventuras y *Western* de ficción.

Cada fichero tiene una línea por cada palabra del texto original. Sin embargo, las *palabras* en el corpus SUSANNE son a menudo más pequeñas que las palabras en el sentido ortográfico

ordinario. Por ejemplo, las marcas de puntuación y los sufijos de apóstrofe s se tratan como palabras separadas y se les asigna líneas distintas. Por otra parte, allí donde los caracteres no se pueden representar adecuadamente mediante el uso directo del juego de caracteres estándar, éstos se representan mediante entidades cuyos nombres figuran entre ángulos. Cuando ha sido posible, esos nombres se han tomado del SGML[5]. Por ejemplo, `<eacute>` es el símbolo utilizado para una e minúscula con acento. Los símbolos entre ángulos se utilizan también para representar situaciones tales como los cambios tipográficos, los cuales también se representan dentro de la secuencia de palabras como elementos separados. Por ejemplo, `<bital>` significa *begin italics*, es decir, comienzo de una porción de texto en letra itálica.

Cada línea de un fichero SUSANNE consta de seis campos separados por tabuladores. Cada campo de cada línea contiene al menos un caracter. Los seis campos de cada línea son los siguientes:

1. Campo de referencia. El campo de referencia contiene 9 caracteres los cuales dan a cada línea un número de referencia que es único a lo largo de todo el corpus SUSANNE, por ejemplo, `N06:1530t`. Los primeros tres caracteres (`N06`, en el ejemplo anterior) son el nombre del fichero. El cuarto caracter es siempre el caracter de dos puntos. Los caracteres quinto al octavo (en el ejemplo `1530`) son el número de la línea en la versión *Bergen I* del corpus BROWN en la cual aparece la palabra. Los números de línea en el corpus BROWN se incrementan normalmente en diez unidades, apareciendo ocasionalmente otros números no múltiplos de diez intercalados. Y el noveno caracter es una letra minúscula que diferencia las palabras sucesivas que aparecen en la misma línea BROWN. Las líneas SUSANNE se enumeran de manera continua desde la `a`, omitiendo la `l` y la `o`.

2. Campo de estado. El campo de estado consta de un caracter. Las letras `A` y `S` indican que la palabra es una abreviatura o un símbolo, respectivamente, tal y como están definidos los códigos del corpus BROWN.

   El corpus SUSANNE está orientado a reflejar la incidencia de los errores que aparecen en el inglés escrito de la vida real y, por tanto, a reproducir esos errores tal y como aparecen en los textos originales en los que se basa el corpus BROWN, pero corrigiendo los errores que fueron introducidos durante el proceso de la construcción del corpus. Cuando un error, o un error aparente, refleja la forma encontrada en la publicación original, se conserva en el corpus SUSANNE marcada con una `E` en el campo de estado. En otro caso, el corpus SUSANNE recupera el texto de la publicación original y el campo de estado ignora el error. Cuando los errores son originales, la etiquetación de las palabras y el análisis gramatical se aplican al texto erróneo de la mejor manera posible estableciendo analogías con las formas correctas.

   En la gran mayoría de las líneas, no se aplica ninguna de estas tres categorías y el campo de estado contiene simplemente un guión.

3. Campo para la etiqueta de la palabra. El conjunto de etiquetas de las palabras en el corpus SUSANNE está basado en el juego de etiquetas LANCASTER [Garside *et al.* 1987]. No obstante, en este juego de etiquetas se han efectuado algunas distinciones gramaticales adicionales indicadas mediante letras en minúsculas que se añaden como sufijos a las etiquetas LANCASTER[6]. Aparte de estas extensiones en minúsculas, las etiquetas son normalmente idénticas a las etiquetas LANCASTER. A las marcas de puntuación se les

[5] *Standard Generalized Markup Language* es un metalenguaje para la marcación o codificación electrónica de textos, adoptado como estándar internacional en 1986 (ISO 8879) [Burnard 1995].

[6] Por ejemplo, `revealing` se etiqueta como `VVG` (participio presente de un verbo) en el esquema LANCASTER, pero como `VVGt` (participio presente de un verbo transitivo) en el esquema SUSANNE.

asignan etiquetas alfabéticas que comienzan por Y[7], y el signo del dólar que aparece en algunas etiquetas LANCASTER para el genitivo se reemplaza por G[8], de manera que las etiquetas LANCASTER modificadas siempre contienen caracteres alfanuméricos y comienzan con letras mayúsculas.

La etiqueta YG aparece en este campo para representar un elemento *fantasma* o una *traza*, es decir, la posición lógica de un constituyente que en la estructura gramatical de superficie ha sido eliminado o desplazado a otra posición.

El juego de etiquetas SUSANNE contiene 425 etiquetas distintas, y se muestra en la sección A.4.

4. Campo para la palabra. Este campo contiene un segmento de texto que normalmente coincide con una palabra en el sentido ortográfico, pero que a veces, tal y como hemos indicado anteriormente, incluye sólo parte de esa palabra ortográfica. Hemos visto también que, en general, este campo representa los mismos fenómenos tipográficos que aparecen en el corpus BROWN, aunque en determinados casos el corpus SUSANNE ha considerado los documentos originales con el fin de reconstruir detalles tipográficos omitidos por el corpus BROWN.

   Algunos caracteres tienen significados especiales en este campo:

   - + aparece sólo como primer caracter del campo de la palabra para indicar que en el texto original el contenido del campo no estaba separado del segmento de texto inmediatamente precedente mediante espacio en blanco[9].
   - - indica que la línea no corresponde a nigún texto material, sino que representa la *traza* de un elemento que ha sido movido.
   - <...> encierran nombres de entidad para características tipográficas especiales, tal y como se discutió anteriormente.

5. Campo para el lema. El campo para el lema muestra la entrada de diccionario de la cual la palabra del texto es una forma, es decir, muestra las formas canónicas de las palabras que aparecen flexionadas en el texto, y también elimina las variaciones tipográficas, tales como la primera letra mayúscula, que no son inherentes a la palabra, sino al contexto en el que se usa. En el caso de las palabras para las cuales el concepto de entrada de diccionario es inapropiado, por ejemplo los numerales y las marcas de puntuación, el campo del lema contiene un guión.

6. Campo para el análisis sintáctico. El contenido de este sexto campo representa la información central del corpus SUSANNE. En él se codifica la estructura gramatical de los textos como una secuencia de árboles etiquetados, los cuales tienen un nodo hoja para cada una de las líneas del corpus.

   Cada texto se trata como una secuencia de párrafos separados por cabeceras. Un párrafo coincide normalmente con un párrafo ortográfico ordinario. Una cabecera puede constar de un texto material real, o puede ser una mera división tipográfica de párrafo, simbolizada mediante <minbrk> en el campo de la palabra. Conceptualmente, la estructura de cada párrafo o cabecera es un árbol con un nodo raíz etiquetado como O (Oh para una cabecera), y con un nodo hoja etiquetado con una etiqueta de palabra para cada palabra SUSANNE

[7] Por ejemplo, YC para la coma.

[8] Por ejemplo, GG para el sufijo de apóstrofe s.

[9] Por ejemplo, en el caso de una marca de puntuación, o en el caso de una secüencia con guiones que ha sido rota en varias líneas SUSANNE.

o para cada traza, es decir, para cada línea del corpus. Comúnmente existirán muchos nodos intermedios también etiquetados.

Estos árboles se representan mediante cadenas de caracteres parentizadas o con corchetes en la forma usual, con las etiquetas de los nodos no terminales escritas dentro de los dos corchetes, es decir, a la derecha del corchete de apertura y a la izquierda del corchete de cierre. Esta cadena se adapta como sigue para incluir en ella sucesivos campos de análisis. Cada vez que un corchete de apertura sigue a un corchete de cierre, la cadena se segmenta produciéndose un segmento por cada nodo hoja. Y dentro de cada uno de esos segmentos, la secuencia `[ etiqueta ]` que representa al nodo hoja se sustituye por un punto. Por tanto, cada campo de análisis contiene exactamente un punto, que corresponde al nodo terminal etiquetado con el contenido del campo 3, el campo para la etiqueta de la palabra, a veces precedido por etiquetas de nodos y corchetes de apertura y a veces seguido por etiquetas de nodos y corchetes de cierre, los cuales correponden a los subárboles que empiezan o finalizan con la palabra de la línea en cuestión.

En total los árboles de análisis del corpus SUSANNE contienen 267.046 nodos, de los cuales 4.383 son raíces y 156.584 son hojas. La notación para las etiquetas de los nodos no terminales se muestra en el apéndice B.

Ejemplo 2.5 Para ilustrar el aspecto general que presentan los datos, a continuación mostramos un conjunto de líneas tomadas directamente del corpus SUSANNE:

```
...
A01:0010b  -  AT     The            the            [O[S[Nns:s.
A01:0010c  -  NP1s   Fulton         Fulton         [Nns.
A01:0010d  -  NNL1cb County         county         .Nns]
A01:0010e  -  JJ     Grand          grand          .
A01:0010f  -  NN1c   Jury           jury           .Nns:s]
A01:0010g  -  VVDv   said           say            [Vd.Vd]
A01:0010h  -  NPD1   Friday         Friday         [Nns:t.Nns:t]
A01:0010i  -  AT1    an             an             [Fn:o[Ns:s.
A01:0010j  -  NN1n   investigation  investigation  .
A01:0020a  -  IO     of             of             [Po.
A01:0020b  -  NP1t   Atlanta        Atlanta        [Ns[G[Nns.Nns]
A01:0020c  -  GG     +<apos>s       -              .G]
A01:0020d  -  JJ     recent         recent         .
A01:0020e  -  JJ     primary        primary        .
A01:0020f  -  NN1n   election       election       .Ns]Po]Ns:s]
A01:0020g  -  VVDv   produced       produce        [Vd.Vd]
A01:0020h  -  YIL    <ldquo>        -              .
A01:0020i  -  ATn    +no            no             [Ns:o.
A01:0020j  -  NN1u   evidence       evidence       .
A01:0020k  -  YIR    +<rdquo>       -              .
A01:0020m  -  CST    that           that           [Fn.
A01:0030a  -  DDy    any            any            [Np:s.
A01:0030b  -  NN2    irregularities irregularity   .Np:s]
A01:0030c  -  VVDv   took           take           [Vd.Vd]
A01:0030d  -  NNL1c  place          place          [Ns:o.Ns:o]Fn]Ns:o]Fn:o]S]
A01:0030e  -  YF     +.             -              .O]
...
```

Este conjunto de líneas constituyen una frase completa. El sexto campo de todas estas líneas es una representación parentizada del árbol de análisis sintáctico que se muestra gráficamente en la figura 2.1. □

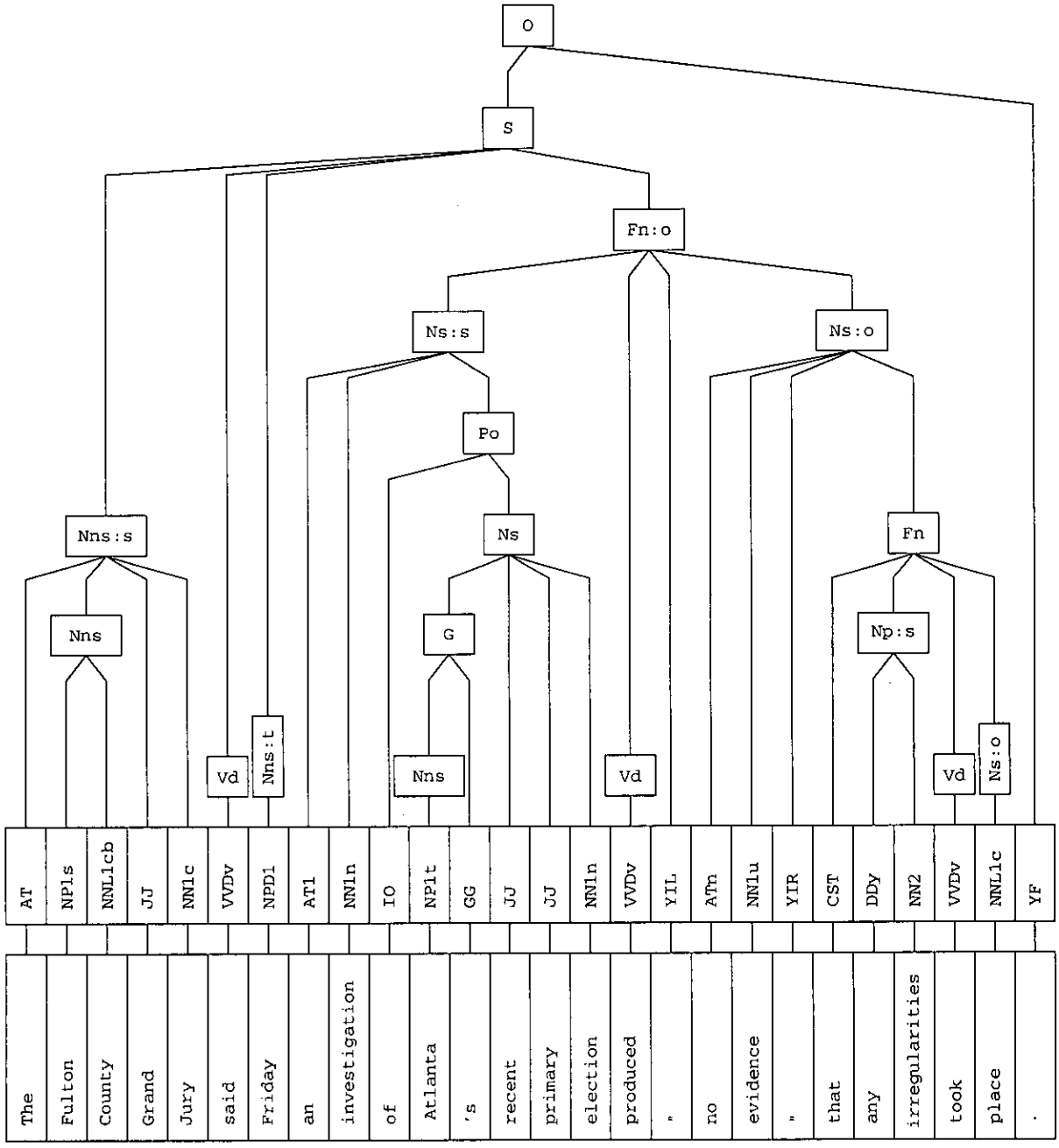


Figura 2.1: Ejemplo de árbol sintáctico para una frase del corpus SUSANNE

La siguiente sección está destinada a describir las características de los recursos lingüísticos que hemos extraído a partir del corpus SUSANNE: una gramática probabilística y dos *corpora* etiquetados.

2.3.2 Extracción de recursos del corpus SUSANNE

La primera transformación real que hemos realizado con el corpus SUSANNE ha sido integrar en una sola línea toda la información léxica y sintáctica disponible para cada frase, manteniedo en los análisis la notación ya introducida, esto es, [y] cada vez que comienza y finaliza un subárbol, respectivamente. Sin embargo, como podemos ver en el apéndice B, el corpus SUSANNE contiene símbolos no terminales para estructuras de nivel más alto que la frase, como por ejemplo párrafos y títulos. Por tanto, hemos tomado cada análisis y hemos eliminado todo lo innecesario, por ejemplo, las marcas de párrafos y las de títulos, para mantener sólo los subanálisis que corresponden a frases reales, es decir, aquéllos cuyo símbolo no terminal raíz comienza con S, F, T, W, A, Z o L.

2.3.2.1 Una gramática probabilística

Nuestro principal objetivo ahora es el de extraer una gramática para realizar nuestros experimentos sobre el corpus SUSANNE.

Como ya hemos visto también, el corpus SUSANNE contiene dos tipos de frases: con trazas y sin trazas. Las frases con trazas pueden ser útiles, por ejemplo, para estudiar otro tipo de fenómenos lingüísticos en los cuales se hace referencia de una manera no explícita a otros componentes de la frase, o incluso de otras frases, tales como la anáfora o la catáfora. Sin embargo, en nuestro caso, si hubiésemos utilizado este tipo de frases durante el proceso de extracción de la gramática, podríamos haber obtenido símbolos no terminales que no se encuentran directamente ligados a componentes lingüísticos reales, o incluso una gramática no independiente del contexto y, por tanto, no directamente tratable por nuestro analizador sintáctico.

Debido a esto, hemos preferido realizar una transformación más, la cual ha consistido en dividir el corpus SUSANNE en dos partes: una parte con todas las frases sin trazas (4.292 frases), a partir de la cual hemos extraído nuestra gramática, y otra parte con todas las frases con trazas (2.188 frases), la cual se ha utilizado como banco de experimentos.

Ejemplo 2.6 A continuación se muestra el aspecto general que presentan la parte de las frases sin trazas:

```
...
[ Fa [ CSf For CSf ] [ Ds:s [ DD1q each DD1q ] [ Po [ IO of IO ] [ Np [ DD2i
these DD2i ] [ NN2 lines NN2 ] Np ] Po ] Ds:s ] [ Vz [ VVZv meets VVZv ]
Vz ] [ YTL <bital> YTL ] [ N:o [ FOx Q FOx ] N:o ] [ YTR <eital> YTR ] [ P:p
[ II in II ] [ Np [ MC three MC ] [ NN2 points NN2 ] [ YC , YC ] [ REX
namely REX ] [ N@ [ MC two MC ] [ NN2 points NN2 ] [ P [ II on II ] [ FOx g
FOx ] P ] [ Ns+ [ CC and CC ] [ MC1 one MC1 ] [ NNL1n point NNL1n ] [ P [ II
on II ] [ Ms [ MC1 one MC1 ] [ Po [ IO of IO ] [ Np [ AT the AT ] [ JJ
multiple JJ ] [ NN2 secants NN2 ] Np ] Po ] Ms ] P ] Ns+ ] N@ ] Np ] P:p ]
Fa ]
...
```

y la parte de las frases con trazas:

```
...
[ Fa [ CSf For CSf ] [ Ni:s [ PPH1 it PPH1 ] Ni:s ] [ Vz [ VVZt includes
VVZt ] Vz ] [ Np:o173 [ AT the AT ] [ JJ emotional JJ ] [ NN2 ties NN2 ]
[ Fr [ CST that CST ] [ s173 [ YG - YG ] s173 ] [ V [ VVOv bind VVOv ] V ]
```

```
[ Np:o [ NN2 men NN2 ] Np:o ] [ P:q [ IIt to IIt ] [ Ns [ APPGh2 their
APPGh2 ] [ NN1c homeland NN1c ] Ns ] P:q ] Fr ] [ Np+:175 [ CC and CC ] [ AT
the AT ] [ JJ complex JJ ] [ NN2 motivations NN2 ] [ Fr [ CST that CST ]
[ s175 [ YG - YG ] s175 ] [ V [ VVOv hold VVOv ] V ] [ Ns:o [ AT1 a AT1 ]
[ JJ large JJ ] [ NNJ1c group NNJ1c ] [ Po [ IO of IO ] [ NNc people NNc ]
Po ] Ns:o ] [ R:w [ RL together RL ] R:w ] [ P:j [ IIa as IIa ] [ Ns [ AT1
a AT1 ] [ NN1c unit NN1c ] Ns ] P:j ] Fr ] Np+:175 ] Np:o173 ] Fa ]
...
```
□

Sobre la parte de frases sin trazas, hemos realizado todavía una transformación más, destinada a los siguientes propósitos. Primero, añadir un nuevo símbolo no terminal, X, que será el axioma o símbolo inicial de nuestra gramática. Segundo, sustituir las etiquetas de las palabras por su número correspondiente dentro del juego de etiquetas del corpus SUSANNE. Y finalmente, eliminar los símbolos no terminales que aparecen repetidos antes de cada corchete de cierre, con el fin de tener disponible el banco de árboles de referencia en el mismo formato de salida producido por nuestro analizador, y de esta manera facilitar la posterior comparación de los árboles de análisis.

Ejemplo 2.7 Este es, por tanto, el aspecto definitivo que presenta la parte de frases sin trazas, a partir de la cual se ha realizado la extracción de la gramática:

```
...
[ X [ Fa [ :33 For ] [ Ds_s [ :59 each ] [ Po [ :121 of ] [ Np [ :74 these ]
[ :172 lines ] ] ] ] [ Vz [ :403 meets ] ] [ :422 <bital> ] [ N_o [ :96
Q ] ] [ :423 <eital> ] [ P_p [ :106 in ] [ Np [ :146 three ] [ :172 points ]
[ :408 , ] [ :302 namely ] [ N@ [ :146 two ] [ :172 points ] [ P [ :106 on ]
[ :96 g ] ] [ Ns+ [ :14 and ] [ :147 one ] [ :182 point ] [ P [ :106 on ]
[ Ms [ :147 one ] [ Po [ :121 of ] [ Np [ :8 the ] [ :133 multiple ] [ :172
secants ] ] ] ] ] ] ] ] ] ]
...
```
□

La extracción de las reglas de la gramática a partir de este tipo de árboles parentizados se puede realizar mediante un sencillo algoritmo que hace uso de una estructura de pila inicialmente vacía, en la cual se van almacenando los símbolos que constituirán cada regla.

Algoritmo 2.1 El algoritmo de extracción de reglas gramaticales a partir de un árbol parentizado simplemente estudia los símbolos que forman parte de dicho árbol parentizado y considera los tres casos siguientes:

- Si aparece el símbolo [estamos ante el comienzo de una regla. Por tanto, se toma el siguiente símbolo del árbol parentizado, se añade a la parte derecha de la regla que está en la cima de la pila, y se crea un nuevo elemento en la pila para una nueva regla cuya parte izquierda es dicho símbolo.
- Si aparece el símbolo] estamos ante el final de una regla. Es decir, la regla que está en la cima de la pila en ese instante está ya completa, con lo cual se toma nota de ella y se elimina de la pila.
- Si aparece cualquier otro símbolo, necesariamente se tratará de una palabra, la cual simplemente se añade a la parte derecha de la regla que está en la cima de la pila en ese momento.

Los pasos básicos de este algoritmo son los siguientes:

```
function Extraer_Reglas (Árbol_Parentizado) =
  begin
    P ← Pila_Vacía;
    R ← Conjunto_Vacío;

    while (queden símbolos del Árbol_Parentizado por procesar) do
      begin
        A ← siguiente símbolo del Árbol_Parentizado;

        case A of
          [ :
            A ← siguiente símbolo del Árbol_Parentizado;
            Regla ← pop (P);
            Añadir A en la parte derecha de Regla;
            push (P, Regla);
            push (P, A → );

          ] :
            Regla ← pop (P);
            Añadir Regla a R;

          cualquier otro símbolo :
            Regla ← pop (P);
            Añadir A en la parte derecha de Regla;
            push (P, Regla)
        end
      end;

    return R
  end;
```

Este proceso de extracción genera gramáticas formadas por reglas *no parcialmente lexicalizadas*. En este tipo de gramáticas, los símbolos terminales aparecen sólo en reglas de la forma $A \rightarrow w_1\, w_2 \ldots w_k$, donde A es una etiqueta y los w_i son símbolos terminales, es decir, palabras. La mayoría de las veces k será igual a 1. Los casos donde $k > 1$ corresponden a las unidades multipalabra, tales como palabras compuestas, expresiones hechas o locuciones. □

Ejemplo 2.8 La figura 2.2 es un ejemplo de la aplicación del algoritmo de extracción de reglas al árbol parentizado:

```
[ X [ S [ :89 There ] [ Vsb [ :368 was ] ] [ Ns_s [ :11 no ] [ :167
chance ] ] ] ]
```

y muestra que efectivamente las reglas gramaticales involucradas en él son:

```
X -> S                  :89 -> There
S -> :89 Vsb Ns_s       :368 -> was
Vsb -> :368             :11 -> no
Ns_s -> :11 :167        :167 -> chance
```

□

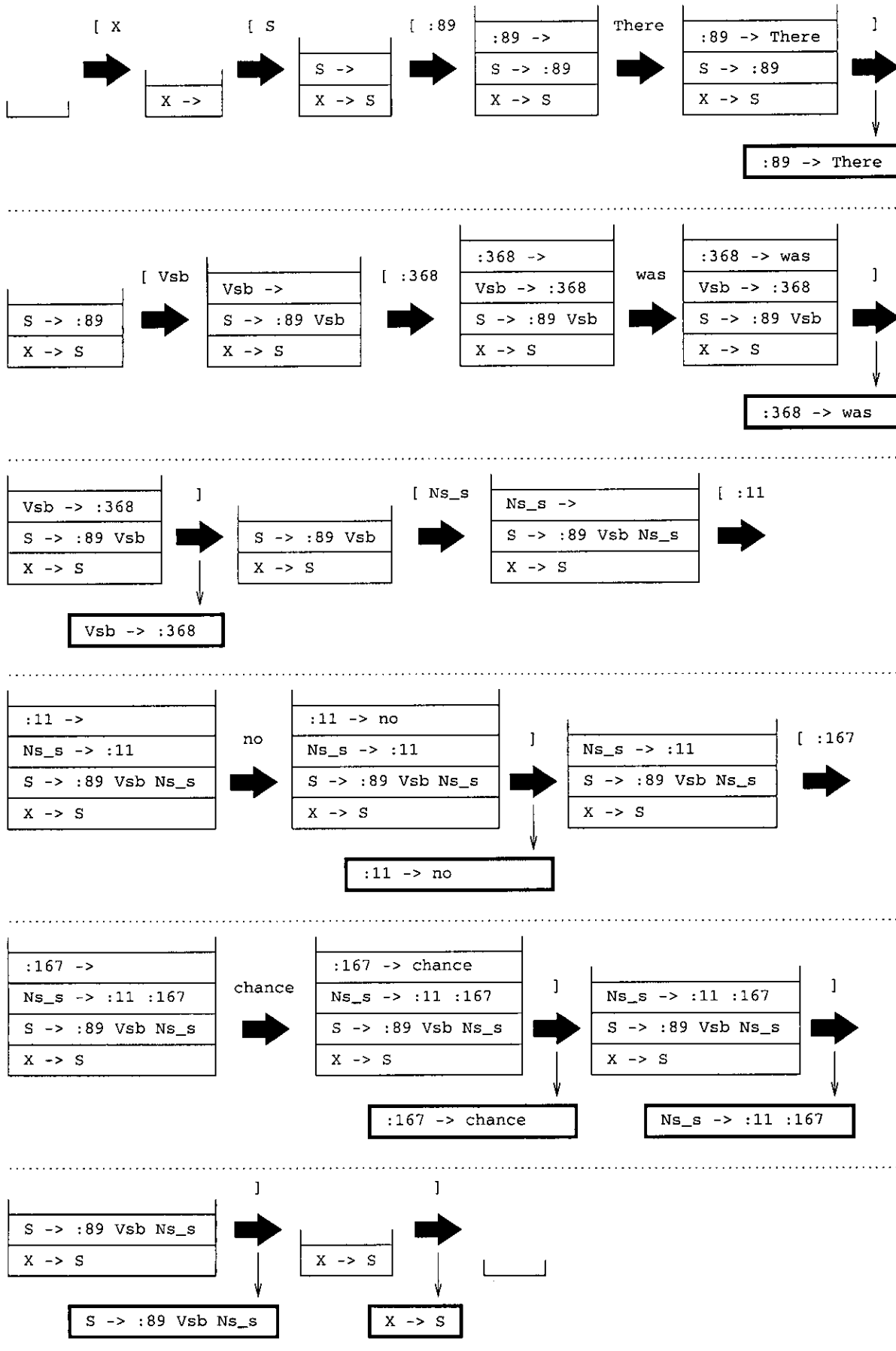


Figura 2.2: Ejemplo de ejecución de la función `Extraer_Reglas`

En la práctica, todas estas reglas se pueden entender como reglas gramaticales, pero no se suelen almacenar juntas. Generalmente, sólo forman parte de la gramática las reglas cuya parte izquierda es un símbolo no terminal, mientras que las reglas cuya parte izquierda es una etiqueta y por tanto cuya parte derecha es una palabra son precisamente las que constituyen el lexicón.

En todo caso, una vez que este proceso de extracción ha sido aplicado a todas las frases sin trazas, y tal y como indica el paradigma de anlisis sintáctico estocástico que estudiaremos con detalle en el capítulo 8, se calcula la probabilidad de cada una de las reglas en relación al resto de producciones que comparten la misma parte izquierda. Es decir, se debe verificar que

$$\sum_{i=1}^{n} P(A \rightarrow \alpha_i) = 1$$

y, por tanto, la probabilidad de una regla genérica $A \rightarrow \alpha$ se calcula como:

$$P(A \rightarrow \alpha) = \frac{\text{número de veces que aparece la regla } A \rightarrow \alpha}{\sum_{i=1}^{n} \text{número de veces que aparece la regla } A \rightarrow \alpha_i}$$

donde: o bien A es cualquier símbolo no terminal, y entonces α y α_i son cualquier combinación de uno o varios símbolos no terminales y/o etiquetas; o bien A es una etiqueta, y entonces α y α_i son palabras. En ambos casos, n es el número de reglas diferentes cuya parte izquierda es A.

Antes de utilizar la gramática, hemos comprobado si contenía ciclos o no, ya que un ciclo podría hacer entrar al analizador sintáctico en un lazo sin fin. Esta comprobación se puede realizar mediante una sencilla operación de matrices. Dado que trabajaremos en todo momento con un analizador sintáctico ascendente, los ciclos sólo pueden ser producidos por las reglas unitarias, esto es, por reglas con un único símbolo no terminal tanto en la parte izquierda como en la parte derecha de la flecha. En el conjunto de reglas extraídas existían 206 reglas unitarias que involucraban a 187 símbolos no terminales. Se construyó entonces una matriz cuadrada U de $187 \times 187 = 34.969$ celdas, de tal manera que dados cualesquiera dos símbolos no terminales X e Y, la celda $U(X,Y)$ contenía la probabilidad de la regla $X \rightarrow Y$. Por tanto, 206 de esas celdas contenían las probabilidades de las reglas unitarias consideradas, mientras que el resto de las celdas estaban a cero. Sabiendo que

$$\sum_{i=0}^{\infty} x^i = 1 + x + x^2 + \ldots = \frac{1}{1-x}$$

calculamos entonces

$$V = \sum_{i=0}^{\infty} U^i = (I - U)^{-1} \tag{2.1}$$

donde I es la matriz identidad de orden 187. Si ahora se calcula $U \times V$, o lo que es lo mismo, $V - I$, se obtiene una matriz cuyas celdas contienen, para cualesquiera dos símbolos no terminales X e Y, lá probabilidad de $X \stackrel{*}{\Rightarrow} Y$, es decir, la probabilidad de que X genere Y en un número finito de pasos. Por tanto, si existen valores diferentes de cero en la diagonal de $U \times V$, estamos ante la presencia de ciclos[10]. Después de la aplicación de este procedimiento, se detectó un único ciclo causado por las reglas `Ot` $\rightarrow$ `Nns` y `Nns` $\rightarrow$ `Ot`. La primera de estas reglas aparece numerosas veces a lo largo de las frases del corpus, mientras que la segunda aparecía una única vez en este punto concreto del corpus:

[10]Por supuesto, dado que este método de detección de ciclos está basado en series formales de endomorfismos, es decir, en correspondencias lineales, el cálculo propuesto no siempre es válido: la serie formal debe converger, es decir, $\lim_{i \rightarrow \infty} U^i = 0$. Esto es equivalente a decir que el radio espectral de la matriz U debe ser menor que 1 [Ciarlet 1988, Teorema 1.5.1, pp. 21-22]. La demostración de este teorema se encuentra también en [Ciarlet *et al.* 1991, Ejercicio 1.5.7, pp. 23-24]. En general, el *radio espectral* de una matriz A, que denotaremos

```
...
G22:0450j -AT The the [Nns[Ot[Np.
G22:0460a -JJ New new [Nns.
G22:0460b -NP1t York York .Nns]
G22:0460c -NNT2 Times time .Np]Ot]Nns]Po]
...
```

La forma de proceder fue suponer que esto se debía a un error de transcripción, reemplazar el subárbol `[ Nns [ Ot [ ... ] Ot ] Nns ]` por `[ Ot [ Nns [ ... ] Nns ] Ot ]` manualmente, e informar de tal anomalía a los constructores del corpus. Tras realizar de nuevo la extración de las reglas, este ciclo desapareció y obtuvimos una gramática directamente utilizable por nuestro analizador sintáctico. Esta gramática está compuesta por 17.669 reglas y 1.525 símbolos no terminales, y la representación arborescente de las reglas (veáse la sección 8.4.7) contiene 28.117 nodos.

Ejemplo 2.9 A continuación, para mostrar el aspecto general que presentan las reglas, incluimos un pequeño conjunto de líneas tomadas directamente de la gramática:

```
X -> Fa (0.002796)                    X -> Fa% (0.000233)
X -> Fa_121 (0.000233)                X -> Fc (0.000466)
X -> Ff (0.000233)                    X -> Fr (0.000233)
X -> L (0.009320)                     X -> L! (0.000233)
X -> L+ (0.000233)                    X -> L? (0.000233)
X -> L?+ (0.000466)                   X -> S (0.916356)
X -> S! (0.001165)                    X -> S!+ (0.000233)
X -> S% (0.000233)                    X -> S* (0.011650)
X -> S*+ (0.000699)                   X -> S+ (0.029590)
X -> S? (0.017008)                    X -> S?+ (0.000932)
X -> S@ (0.000699)                    X -> S_129 (0.000233)
X -> S_133 (0.000233)                 X -> S_145 (0.000233)
X -> S_149 (0.000233)                 X -> S_151 (0.000233)
X -> S_205 (0.000233)                 X -> S_209 (0.000233)
X -> S_223 (0.000233)                 X -> Tb (0.000466)
X -> Tb! (0.000233)                   X -> Tb? (0.000466)
X -> Tg (0.002563)                    X -> Tg+ (0.000233)
X -> Ti (0.000466)                    X -> Tn (0.000466)
A -> :27 Ni_s Vzb P_r (0.105263)      A -> :27 Ns_S Vzp (0.0526316)
A -> :27 P_p (0.0526316)              A -> :27 R_t (0.0526316)
A -> :27 R_t Jh_e (0.0526316)         A -> :27 Ti_z (0.0526316)
```

mediante $\rho(A)$, se calcula como sigue. Dada A, una matriz cuadrada de orden n, calculamos el determinante $|A - \lambda I|$, donde I es la matriz identidad también de orden n. Lo que obtenemos no es un valor concreto, sino un polinomio de grado n en λ, de la forma $a_n\lambda^n + a_{n-1}\lambda^{n-1} + \ldots + a_1\lambda + a_0$. Dicho polinomio se denomina *polinomio característico* de A, y sus raíces, $r_1, r_2, \ldots, r_n$, son los *valores propios* de A. Pues bien, si de cada uno de los valores propios consideramos su valor absoluto, el radio espectral de la matriz A será el máximo de esos valores absolutos. Es decir, $\rho(A) = \max\{|r_i|, \text{tal que } r_i, 1 \leq i \leq n, \text{es un valor propio de } A\}$. Por tanto, si $\rho(U) < 1$, el sumatorio de la ecuación (2.1) tiene sentido y converge efectivamente hacia el valor $(I - U)^{-1}$. Pero en definitiva, es suficiente con intentar el cálculo de la inversa de la matriz $I - U$. Si dicha matriz no resulta inversible, la matriz V debe ser calculada a través de un procedimiento iterativo que acumule la suma de las sucesivas potencias de U, tal y como se deduce de los métodos de búsqueda de caminos propuestos en la teoría de grafos [Grassmann y Tremblay 1996, Teorema 7.4, pp. 365-366], aunque obviamente no consideraremos las potencias hasta infinito, sino hasta el número que nos interese para volver al punto de partida y detectar así los ciclos (en nuestro caso, hasta la potencia 187).

```
A -> :27 Vg Fn_o (0.0526316)          A -> :27 Vn P_p (0.105263)
A -> :27 Vn P_q (0.0526316)           A -> :27 Vn P_r (0.0526316)
A -> :27 Vn P_u (0.0526316)           A -> :27 Vn Pb_a (0.315789)
A+ -> :14 Vn R_q P_q (0.5)            A+ -> :20 :27 Np_s V Ns_o (0.5)
A_c -> :27 Ni_s Vd R_n (0.333333)     A_c -> :27 Ni_s Vsb (0.333333)
A_c -> :27 Ns_s Vd Ni_o (0.333333)    ...                                  □
```

2.3.2.2 Dos *corpora* etiquetados

Para ser utilizado como recurso de referencia por las herramientas de etiquetación ya existentes, es también sencillo obtener un corpus etiquetado a la manera tradicional a partir del corpus SUSANNE, fijándonos únicamente en los nodos hoja, es decir, en las palabras y en sus etiquetas correspondientes, y despreciando la información relativa al resto de nodos. De esta forma, tendremos un buen marco de pruebas para el estudio comparativo de las técnicas de etiquetación tradicionales y las propuestas en este trabajo. Siguiendo la misma filosofía utilizada a la hora de extraer la gramática, hemos generado dos *corpora* etiquetados: uno correspondiente a la parte de frases sin trazas y otro a la parte de frases con trazas. A continuación se describen las características de cada uno de ellos:

- El corpus etiquetado correspondiente a la parte de frases sin trazas tiene 4.292 frases y 77.275 palabras, es decir, un número medio de 18 palabras por frase. El tamaño del fichero es de 751.359 caracteres. Las características del lexicón o diccionario constituido por todas las formas que aparecen en este corpus son las siguientes:

  10.938 formas con 1 etiqueta, 864 formas con 2 etiquetas,
  87 formas con 3 etiquetas, 27 formas con 4 etiquetas,
  9 formas con 5 etiquetas, 5 formas con 6 etiquetas,
  2 formas con 7 etiquetas, 1 forma con 8 etiquetas y
  2 formas con 9 etiquetas.

  Esto es, 11.935 formas diferentes, con 13.150 etiquetas posibles. Si calculamos el porcentaje de formas ambiguas y el número medio de etiquetas por forma, obtenemos el siguiente resultado:

$$\%\ \text{formas ambiguas} = \frac{\#\ \text{formas ambiguas}}{\#\ \text{formas}} \times 100 = \frac{11.935 - 10.938}{11.935} \times 100 = 8,35\ \%$$

$$\#\ \text{medio de etiquetas por forma} = \frac{\#\ \text{etiquetas}}{\#\ \text{formas}} = \frac{13.150}{11.935} = 1,10\ \text{etiquetas por forma.}$$

  Mucho más interesante es calcular las mismas características directamente con todas las palabras del corpus, y obtenemos las siguientes cifras:

  42.550 palabras con 1 etiqueta, 13.004 palabras con 2 etiquetas,
  9.225 palabras con 3 etiquetas, 2.175 palabras con 4 etiquetas,
  2.174 palabras con 5 etiquetas, 2.015 palabras con 6 etiquetas,
  1.539 palabras con 7 etiquetas, 2.574 palabras con 8 etiquetas y
  2.019 palabras con 9 etiquetas.

  Esto es, 77.275 palabras, con 177.429 etiquetas posibles. Si calculamos de nuevo el porcentaje de palabras ambiguas y el número medio de etiquetas por palabra, obtenemos entonces:

$$\%\ \text{palabras ambiguas} = \frac{\#\ \text{palabras ambiguas}}{\#\ \text{palabras}} \times 100 = \frac{77.275 - 42.550}{77.275} \times 100 = 44,93\ \%$$

$$\#\ \text{medio de etiquetas por palabra} = \frac{\#\ \text{etiquetas}}{\#\ \text{palabras}} = \frac{177.429}{77.275} = 2,30\ \text{etiquetas por palabra.}$$

- El corpus etiquetado correspondiente a la parte de frases con trazas tiene 2.188 frases y 60.759 palabras, es decir, un número medio de 28 palabras por frase. El tamaño del fichero es de 593.735 caracteres. Las características del lexicón o diccionario constituido por todas las formas que aparecen en este corpus son las siguientes:

  9.322 formas con 1 etiqueta, 687 formas con 2 etiquetas,
  70 formas con 3 etiquetas, 21 formas con 4 etiquetas,
  8 formas con 5 etiquetas, 3 formas con 6 etiquetas,
  2 formas con 7 etiquetas, 1 forma con 8 etiquetas y
  1 formas con 14 etiquetas.

  Esto es, 10.115 formas diferentes, con 11.084 etiquetas posibles. Si calculamos el porcentaje de formas ambiguas y el número medio de etiquetas por forma, obtenemos el siguiente resultado:

$$\%\ \text{formas ambiguas} = \frac{\#\ \text{formas ambiguas}}{\#\ \text{formas}} \times 100 = \frac{10.115 - 9.322}{10.115} \times 100 = 7,84\ \%$$

$$\#\ \text{medio de etiquetas por forma} = \frac{\#\ \text{etiquetas}}{\#\ \text{formas}} = \frac{11.084}{10.115} = 1,06\ \text{etiquetas por forma.}$$

  Mucho más interesante es calcular las mismas características directamente con todas las palabras del corpus, y obtenemos las siguientes cifras:

  34.329 palabras con 1 etiqueta, 8.957 palabras con 2 etiquetas,
  3.580 palabras con 3 etiquetas, 2.232 palabras con 4 etiquetas,
  4.984 palabras con 5 etiquetas, 2.268 palabras con 6 etiquetas,
  2.820 palabras con 7 etiquetas, 1.236 palabras con 8 etiquetas y
  353 palabras con 14 etiquetas.

  Esto es, 60.759 palabras, con 145.009 etiquetas posibles. Si calculamos de nuevo el porcentaje de palabras ambiguas y el número medio de etiquetas por palabra, obtenemos entonces:

$$\%\ \text{palabras ambiguas} = \frac{\#\ \text{palabras ambiguas}}{\#\ \text{palabras}} \times 100 = \frac{60.759 - 34.329}{60.759} \times 100 = 43,50\ \%$$

$$\#\ \text{medio de etiquetas por palabra} = \frac{\#\ \text{etiquetas}}{\#\ \text{palabras}} = \frac{145.009}{60.759} = 2,39\ \text{etiquetas por palabra.}$$

La figura 2.3 resume gráficamente el proceso que hemos seguido para extraer los recursos lingüísticos a partir del corpus SUSANNE, junto con las características básicas de los mismos.

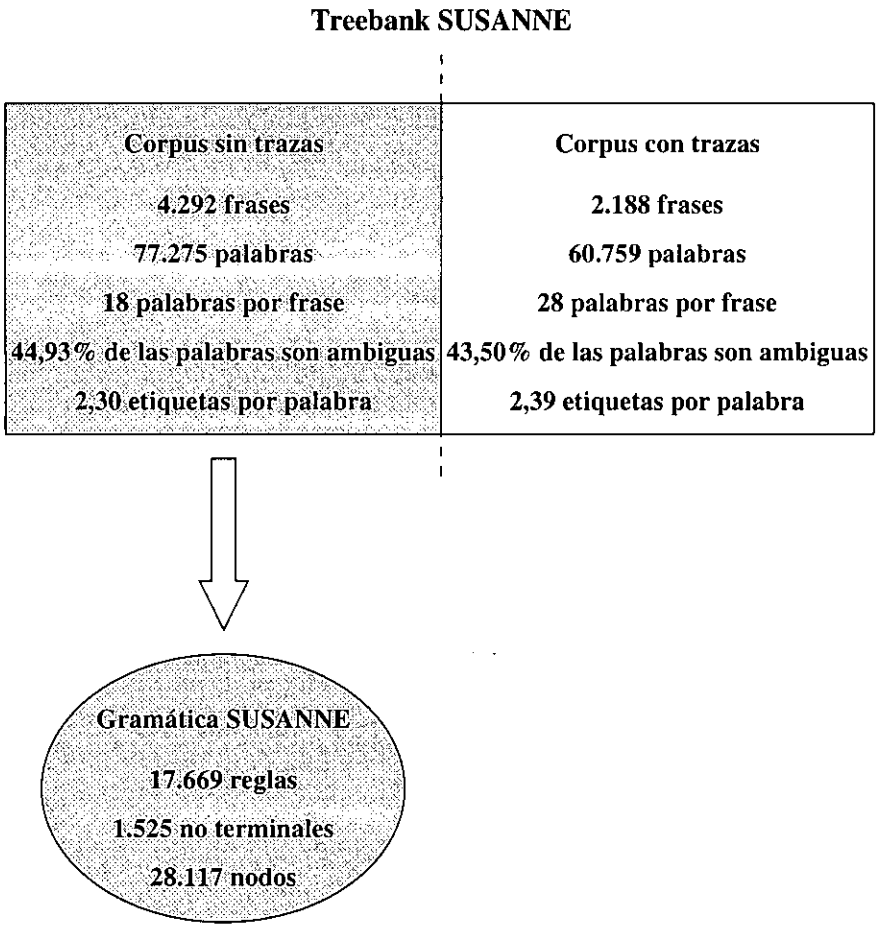


Figura 2.3: División y extracción de recursos lingüísticos a partir del corpus SUSANNE

Los siguientes capítulos explicarán detalladamente cómo hemos utilizado todos los recursos lingüísticos presentados aquí, no sólo para testear la calidad de los mismos, sino también para evaluar el rendimiento de las distintas técnicas de etiquetación, tanto las tradicionales como las de nuevo diseño.

Capítulo 3

Análisis léxico de grandes diccionarios

El problema de la correcta etiquetación o del análisis sintáctico de una frase dada puede resultar complejo si se aborda tratando directamente el flujo de los caracteres de entrada que conforman esa frase. Para evitar dicha complejidad, normalmente existirá un paso de procesamiento previo, cuya misión es la de transformar el flujo de caracteres de entrada en un flujo de elementos de más alto nivel de significado[1], que típicamente serán las palabras de la frase en cuestión, y la de obtener rápida y cómodamente todas las etiquetas candidatas de esas palabras. Dicho paso previo se denomina análisis léxico[2].

Este capítulo está destinado a esbozar las distintas técnicas existentes para la realización de esta tarea. No obstante, comenzaremos presentando en detalle la visión particular que se ha utilizado a lo largo de este trabajo para modelizar los diccionarios, y la técnica concreta con la que se han implementado.

3.1 Modelización de un diccionario

Si bien es cierto que muchas de las palabras que aparecen en un diccionario se pueden capturar automáticamente a partir de los textos etiquetados, otras muchas serán introducidas manualmente por los expertos lingüistas, con el fin de cubrir de manera exhaustiva algunas categorías de palabras poco pobladas, que constituyen el núcleo invariable de un idioma (artículos, preposiciones, conjunciones, etc.), o alguna terminología particular correspondiente a un determinado ámbito de aplicación. Sin embargo, cuando nos enfrentamos a lenguajes naturales que presentan un paradigma de inflexión de gran complejidad, resulta impensable que el usuario tenga que introducir en el diccionario todas y cada una de las formas derivadas de un lema dado[3]. En lugar de esto, resulta mucho más conveniente realizar un estudio previo que identifique los diferentes grupos de inflexión (género, número, irregularidad verbal, etc.), de manera que a la hora de introducir posteriormente un nuevo término en el diccionario, el usuario sólo tenga que hacer mención de las raíces involucradas en él y de los grupos de inflexión mediante los cuales se realiza la derivación de formas desde cada una de esas raíces [Graña *et al.* 1994].

Por supuesto, una interfaz gráfica que genere temporalmente las formas correspondientes a la operación de inserción que se va a realizar, tal y como se muestra en la figura 3.1, puede resultar de gran ayuda para el usuario al permitirle comprobar si efectivamente está realizando

[1] Normalmente denominados *tokens*.

[2] O también *scanning*.

[3] Por ejemplo, el paradigma de conjugación verbal del español puede utilizar hasta 118 formas distintas para un mismo verbo.

la inserción de las raíces en el grupo correcto o no [Vilares *et al.* 1997]. Pero tal y como vimos en la sección 2.2.2, en un primer momento la representación inicial de un diccionario consiste en una base de datos que almacena la información léxica correspondiente sólo a las raíces y a sus grupos de inflexión correspondientes.

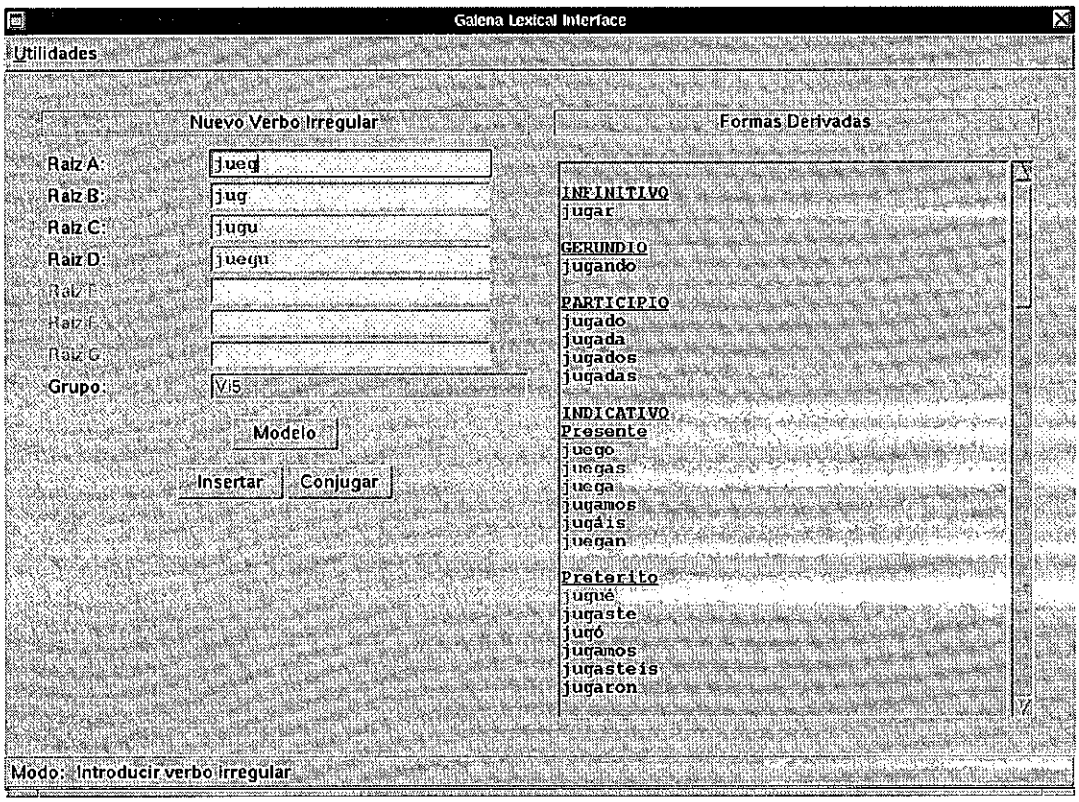


Figura 3.1: Interfaz gráfica para la introducción de términos en el diccionario

Sin embargo, en este punto surgen dos problemas:

1. El primero de ellos es un problema inherente al uso de bases de datos. Las bases de datos se presentan como herramientas válidas para el almacenamiento de gran cantidad de información estructurada, ya que resultan muy flexibles a la hora de realizar tareas de gestión y mantenimiento: las inserciones, actualizaciones, borrados y consultas se realizan mediante operaciones relacionales muy potentes que pueden implicar a uno o varios campos de información, de la misma o de diferentes tablas de datos. Pero cuando se procesan textos grandes, quizás de millones de palabras, no sólo interesa un acceso flexible a los datos, sino también un acceso muy rápido, y una base de datos no es el mejor mecanismo a la hora de recuperar este tipo de información, y menos aún si en ella residen las raíces y no las palabras concretas que aparecen en los textos. Existen mecanismos mucho más eficientes para esta última tarea, como pueden ser los autómatas finitos que describiremos en la siguiente sección.

2. El segundo de los problemas es que muchas aplicaciones necesitan incorporar información adicional relativa a las palabras, no a las raíces. Tal es el caso de determinados paradigmas de etiquetación estocástica o de análisis sintáctico estocástico, que necesitan asociar una probabilidad a cada una de las combinaciones posibles palabra-etiqueta. Por ejemplo, en el contexto de los modelos de Markov ocultos, que veremos más adelante, esa probabilidad representa la *probabilidad de emisión* de la palabra dentro del conjunto de

todas las palabras que tienen esa misma etiqueta. O bien, dentro del marco del análisis sintáctico estocástico, esa misma probabilidad puede verse como la probabilidad de la regla gramatical *etiqueta* → *palabra*. Así que una vez más, la información de las raíces no es suficiente. Necesitamos disponer de todas las palabras que se pueden generar a partir de esas raíces. El mismo procedimiento que aparece en la interfaz gráfica de introducción de raíces se puede también utilizar aquí para generar todas las palabras y, posteriormente, a través de otro procedimiento para la estimación de las probabilidades, que veremos con detalle en el próximo capítulo, podemos integrar toda la información necesaria, de manera que resida junta en un mismo recurso.

Por lo tanto, en este segundo momento, nuestra visión de un diccionario o lexicón es simplemente un fichero de texto, donde cada línea tiene el siguiente formato:

```
palabra etiqueta lema probabilidad
```

Las palabras ambiguas, es decir, con varias etiquetaciones posibles, utilizarán una línea diferente para cada una de esas etiquetas.

Ejemplo 3.1 Sin pérdida de generalidad, las palabras podrían estar ordenadas alfabéticamente, de tal manera que, en el caso del diccionario del sistema GALENA [Vilares *et al.* 1995], el punto donde aparece la ambigüedad de la palabra **sobre** presenta el siguiente aspecto:

```
...
sobraste V2sei0 sobrar 0.00162206
sobrasteis V2pei0 sobrar 0.00377715
sobre P sobre 0.113229
sobre Scms sobre 0.00126295
sobre Vysps0 sobrar 0.0117647
sobrecarga Scfs sobrecarga 0.00383284
sobrecarga V2spm0 sobrecargar 0.00175131
sobrecarga V3spi0 sobrecargar 0.000629723
sobrecargaba Vysii0 sobrecargar 0.0026455
...
```

Retomando los datos de la sección 2.2.2, recordemos que el diccionario del sistema GALENA tiene 291.604 palabras diferentes, con 354.007 etiquetaciones posibles. Esta última cifra es precisamente el número de líneas del fichero anterior. Para una discusión posterior, diremos ahora que la primera etiquetación de la palabra **sobre**, es decir, como preposición,

```
sobre P sobre 0.113229
```

aparece en la línea 325.611 dentro de ese fichero. Y diremos también que, en el conjunto de todas las 291.604 palabras diferentes ordenadas alfabéticamente, la palabra **sobre** ocupa la posición 268.249. □

Por supuesto, ésta tampoco es todavía la versión operativa final de un diccionario. El problema del acceso eficiente a los datos sigue aún presente, pero éste es un problema que, como ya hemos dicho, resolveremos en la siguiente sección. Lo realmente importante ahora es obtener una versión compilada que represente de una manera más compacta todo este gran volumen de información. Esta versión compilada y su forma de operar se muestran en la figura 3.2, donde se pueden identificar cada uno de los siguientes elementos:

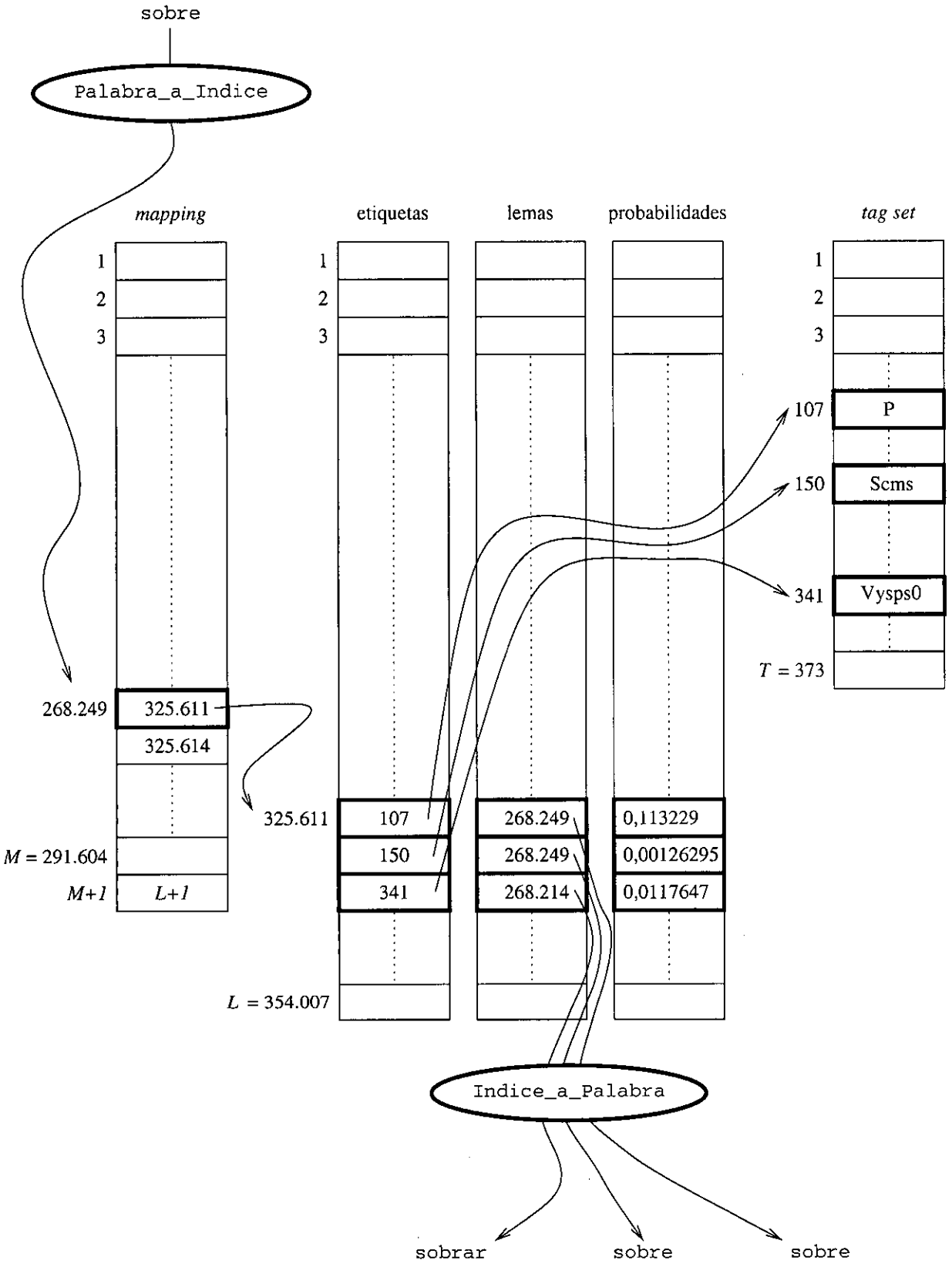


Figura 3.2: Modelización compacta de un diccionario

- La función `Palabra_a_Índice`, que explicaremos con detalle en la siguiente sección, es capaz de convertir una palabra en un número que representa la posición que ocupa esa palabra dentro del conjunto de todas las palabras diferentes ordenadas alfabéticamente. Por ejemplo, esta función transforma `sobre` en el número 268.249.

- Ese número sirve para indexar un tablero de correspondencia[4] de tamaño $M + 1$, que transforma la posición relativa de cada palabra en la posición absoluta dentro del lexicón original. En el caso del diccionario del sistema GALENA, M es igual a 291.604, el número de palabras distintas. En el caso de `sobre`, la posición relativa 268.249 se transforma en la posición absoluta 325.611.

- Ese número sirve para indexar los tableros de *etiquetas*, *lemas* y *probabilidades*. Todos estos tableros son de tamaño L. En el caso del diccionario del sistema GALENA, L es igual a 354.007, el número de etiquetaciones distintas.

- El tablero de *etiquetas* almacena números. Una representación numérica de las etiquetas es más compacta que los nombres de las etiquetas en sí. Las etiquetas originales se pueden recuperar indexando con esos números el tablero del juego de etiquetas[5]. El tablero del juego de etiquetas tiene un tamaño T. En el caso del diccionario del sistema GALENA, T es igual a 373, el número de etiquetas distintas.

  Dado que hemos partido de una ordenación alfabética, está garantizado que las etiquetas de una misma palabra aparecen contiguas. No obstante, necesitamos saber de alguna manera dónde terminan las etiquetas de una palabra dada. Para ello, es suficiente con restarle el valor de la posición absoluta de la palabra al valor de la siguiente casilla en el tablero de correspondencia. En nuestro caso, la palabra `sobre` tiene $325.614 - 325.611 = 3$ etiquetas. Esta operación es también válida para acceder correctamente a la información de los tableros de *lemas* y *probabilidades*.

- El tablero de *lemas* almacena también números. Un lema es una palabra que debe estar también presente en el diccionario. El número que la función `Palabra_a_Índice` obtendría para esa palabra es el número que se almacena aquí, siendo esta representación mucho más compacta que el lema en sí. El lema se puede recuperar aplicando la función `Índice_a_Palabra`, que explicaremos con detalle en la siguiente sección.

- El tablero de *probabilidades* almacena directamente las probabilidades. En este caso no es posible realizar ninguna compactación.

Ésta es por tanto la representación más compacta que se puede diseñar para albergar toda la información léxica relativa a las palabras presentes en un diccionario. Es además una representación muy flexible en el sentido de que resulta particularmente sencillo incorporar nuevos tableros, si es que se necesita algún otro tipo de información adicional. Por ejemplo, algunas aplicaciones de lexicografía computacional que realicen estudios sobre el uso de un idioma podrían utilizar un tablero de números enteros que almacene la frecuencia de aparición de las palabras en un determinado texto. De igual manera, aquellos tableros que no se utilicen se pueden eliminar, con el fin de ahorrar su espacio correspondiente[6].

Por otra parte, ideando un nuevo método de acceso o una nueva disposición de los elementos de los tableros, es también sencillo transformar esta estructura en un generador de formas, es

[4] O tablero de *mapping*.

[5] O tablero de *tag set*.

[6] Por ejemplo, no todas las aplicaciones hacen uso del lema y de la probabilidad, pudiendo ser suficiente sólo con las etiquetas.

decir, en un mecanismo bidireccional que no sólo reconozca palabras, sino que también, dado un lema y una etiqueta, genere la forma flexionada correspondiente. Esta funcionalidad es indispensable en aplicaciones de generación de lenguaje, es decir, en aquellas aplicaciones donde el flujo de información que va desde el sistema hacia el usuario se realiza utilizando lenguaje natural [Vilares *et al.* 1996a].

Otro aspecto más de la flexibilidad de esta representación es el que se describe a continuación. Cuando al procesar un texto aparece una palabra que no está presente en nuestro diccionario, el tratamiento normal será asumir que es desconocida y posteriormente intentar asignarle la etiqueta que describe su papel en la frase, igual que si se tratara de una palabra normal. La diferencia radica en que para la palabra desconocida no podemos obtener un conjunto inicial de etiqueta candidatas, de manera que habrá que definir una serie de categorías *abiertas* que permitan inicializar dicho conjunto. Sin embargo, hay aplicaciones en las que puede ser más conveniente suponer que todas las categorías están *cerradas* y que por tanto esa palabra es realmente una palabra del diccionario, pero que presenta algún error lexicográfico que hay que corregir. Ocurre entonces que el hecho de tener la información de etiquetación totalmente separada del mecanismo de reconocimiento de las palabras en sí facilita la posterior incorporación de algoritmos de corrección automática de este tipo de errores lexicográficos [Vilares *et al.* 1996b].

Finalmente, para completar la modelización de diccionarios que hemos presentado, sólo nos resta detallar la implementación de las funciones `Palabra_a_Índice` e `Índice_a_Palabra`. Ambas funciones trabajan sobre un tipo especial de autómatas finitos, los autómatas finitos acíclicos deterministas numerados, que se describen a continuación.

3.2 Autómatas finitos acíclicos deterministas numerados

La manera más eficiente de implementar analizadores léxicos[7] es quizás mediante el uso de autómatas finitos [Hopcroft y Ullman 1979]. La aplicación más tradicional de esta idea la podemos encontrar en algunas de las fases de construcción de compiladores para los lenguajes de programación [Aho *et al.* 1985]. El caso del procesamiento de los lenguajes naturales es cuantitativamente diferente, ya que surge la necesidad de representar diccionarios léxicos que muchas veces pueden llegar a involucrar a cientos de miles de palabras. Sin embargo, el uso de los autómatas finitos para las tareas de análisis y reconocimiento de las palabras sigue siendo una técnica perfectamente válida.

Definición 3.1 Un *autómata finito* es una estructura algebraica que se define formalmente como una 5-tupla $A = (Q, \Sigma, \delta, q_0, F)$, donde:

- Q es un conjunto finito de estados,
- Σ es un alfabeto finito de símbolos de entrada, es decir, el alfabeto de los caracteres que conforman las palabras,
- δ es una función del tipo $Q \times \Sigma \rightarrow P(Q)$ que define las transiciones del autómata,
- q_0 es el estado inicial del autómata, y
- F es el subconjunto de Q al que pertenecen los estados que son finales.

El estado o conjunto de estados que se alcanza mediante la transición etiquetada con el símbolo a desde el estado q se denota como $q.a = \delta(q, a)$. Cuando este estado es único, es decir, cuando la función δ es del tipo $Q \times \Sigma \rightarrow Q$, se dice que el autómata finito es *determinista*. □

[7] También denominados *scanners*.

La anterior notación es transitiva, es decir, si w es una palabra de n letras, entonces $q.w$ denota el estado alcanzado desde q utilizando las transiciones etiquetadas con cada una de las letras w_1, w_2, ..., w_n de w. Una palabra w es aceptada por el autómata si $q_0.w$ es un estado final.

Definición 3.2 Se denota como $L(A)$ el *lenguaje reconocido por el autómata* A, es decir, el conjunto de todas las palabras w tales que $q_0.w \in F$. □

Definición 3.3 Un autómata finito es *acíclico* cuando su grafo subyacente es acíclico. Los autómatas finitos acíclicos reconocen lenguajes formados por conjuntos finitos de palabras. □

3.2.1 Construcción del autómata

La primera estructura que nos viene a la mente para implementar un reconocedor de un conjunto finito de palabras dado es un *árbol de letras.*

Ejemplo 3.2 El árbol de letras de la figura 3.3 reconoce todas las formas de los verbos ingleses `discount`, `dismount`, `recount` y `remount`, es decir, el conjunto finito de palabras[8]:

`discount`	`discounted`	`discounting`	`discounts`
`dismount`	`dismounted`	`dismounting`	`dismounts`
`recount`	`recounted`	`recounting`	`recounts`
`remount`	`remounted`	`remounting`	`remounts`

Esta estructura es en sí misma un autómata finito acíclico determinista, donde el estado inicial es el 0 y los estados finales aparecen marcados con un círculo más grueso. □

Como en todo autómata finito, la complejidad de reconocimiento de un árbol de letras es lineal respecto a la longitud de la palabra a analizar, y no depende para nada ni del tamaño del diccionario, ni del tamaño de dicho autómata.

Sin embargo, los requerimientos de memoria de esta estructura de árbol pueden llegar a ser elevadísimos cuando el diccionario es muy grande. Por ejemplo, el dicionario del sistema GALENA necesitaría un árbol de más de un millón de nodos para reconocer las 291.604 palabras diferentes. Por tanto, en lugar de utilizar directamente esta estructura, le aplicaremos un proceso de minimización para obtener otro autómata finito con menos estados y menos transiciones. Los autómatas finitos tienen una propiedad que garantiza que este proceso de minimización siempre se puede llevar a cabo, y que además el nuevo autómata resultante es equivalente, es decir, reconoce exactamente el mismo conjunto de palabras que el autómata original [Hopcroft y Ullman 1979]. Por otra parte, en el caso de los autómatas finitos acíclicos deterministas, este proceso de minimización es particularmente sencillo, tal y como veremos más adelante.

Ejemplo 3.3 El autómata finito acíclico determinista mínimo correspondiente al árbol de letras de la figura 3.3 es el que se muestra en la figura 3.4. □

Por otra parte, y debido una vez más a esos mismos requerimientos de memoria, no resulta conveniente construir un diccionario insertando primero todas y cada una de las palabras en un árbol de letras y obteniendo después el autómata mínimo correspondiente a dicho árbol. En lugar de esto, es mucho más aconsejable realizar varias etapas de inserción y minimización. Por tanto, la construcción del autómata se realiza de acuerdo con los pasos básicos del siguiente algoritmo.

[8] El símbolo # que aparece en las figuras denota el fin de palabra.

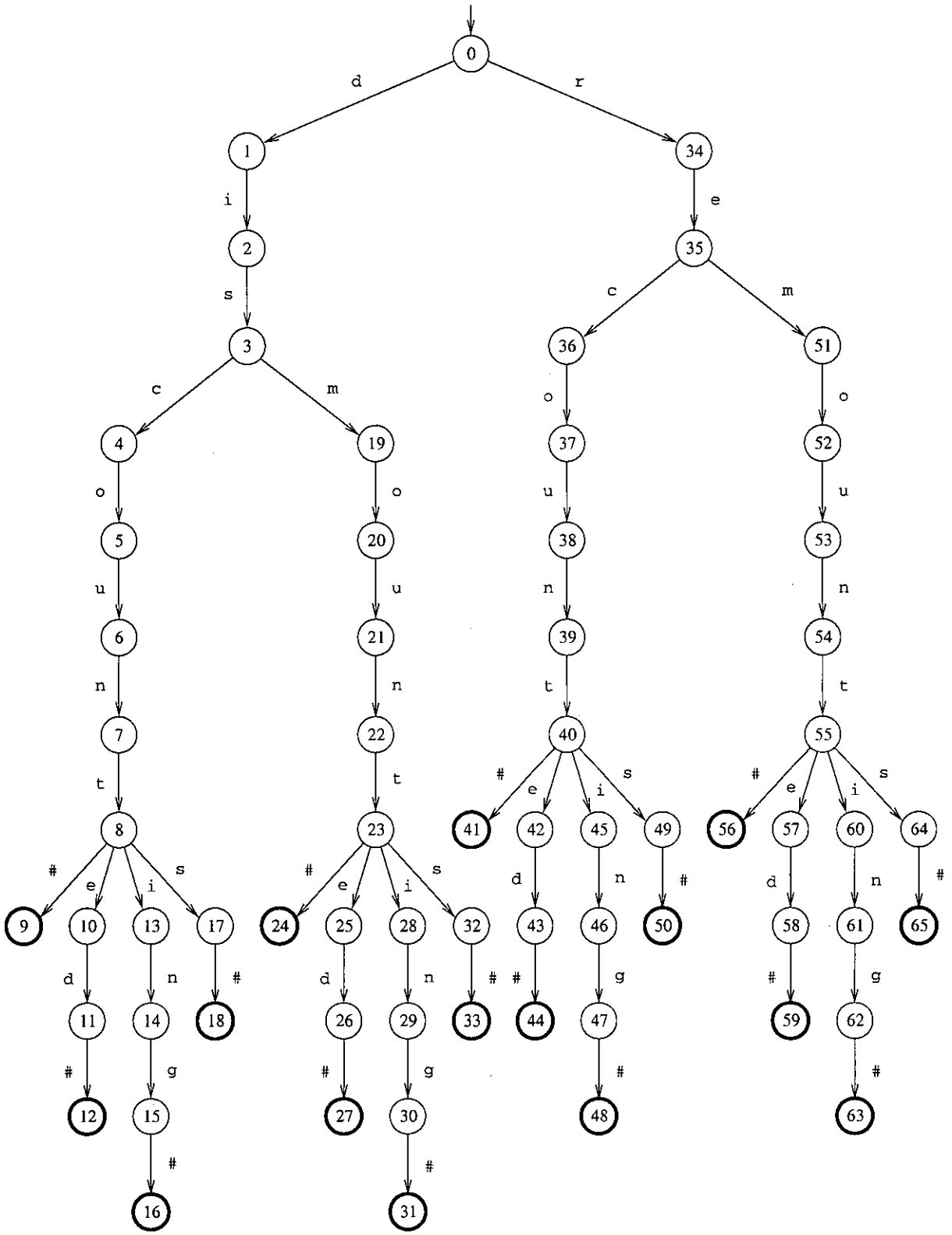


Figura 3.3: Árbol de letras para las formas de los verbos `discount`, `dismount`, `recount` y `remount`

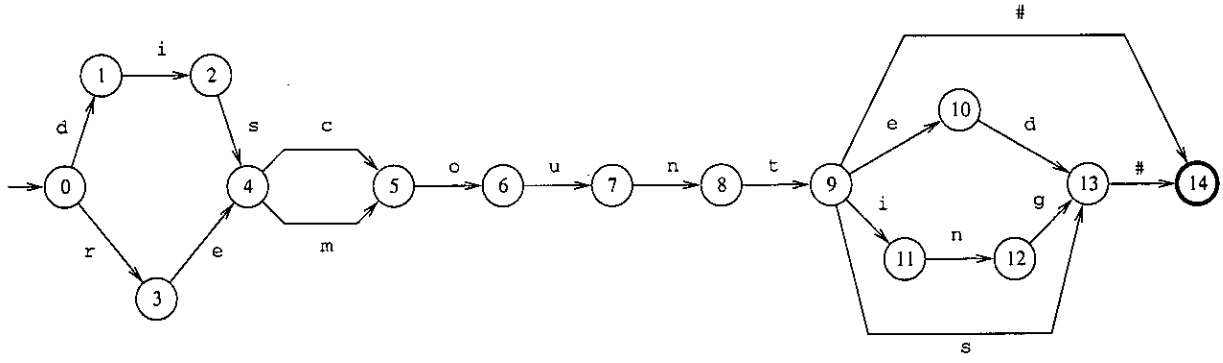


Figura 3.4: Autómata finito acíclico determinista mínimo para las formas de los verbos `discount`, `dismount`, `recount` y `remount`

Algoritmo 3.1 Algoritmo para la construcción de un autómata finito acíclico determinista a partir de un lexicón:

function *Construir_Autómata* (*Lexicón*) =
  begin
    $A \leftarrow$ *Autómata_Vacío*;

    while (*queden palabras del Lexicón por insertar*) **do**
      begin
        if (*A está lleno*) **then**
          $A \leftarrow$ *Minimizar_Autómata* (*A*);
        Insertar la siguiente palabra del Lexicón en A
      end;

    $A \leftarrow$ *Minimizar_Autómata* (*A*);
    return *A*
  end;

Eligiendo un tamaño máximo previo para el autómata, este algoritmo de construcción permite que los consumos tanto de memoria como de tiempo por parte de los procesos de inserción y minimización sean muchísimo más moderados. □

Ejemplo 3.4 Fijando el tamaño máximo en 65.536 estados[9], el proceso de construcción del autómata finito acíclico determinista mínimo para el diccionario del sistema GALENA necesitó 10 etapas de inserción y minimización. La evolución de dicho proceso se puede observar en la tabla 3.1. □

En este momento es importante indicar cómo se debe realizar la correcta inserción de nuevas palabras en un autómata durante el proceso de construcción del mismo. Dicha inserción se puede llevar a cabo mediante una sencilla operación recursiva que hace uso de las transiciones que ya han aparecido, con el fin de compartir los caminos ya existentes en el grafo. Pero ocurre que este procedimiento *estándar* de inserción sólo es válido para los árboles de letras, y realmente nuestro

[9]La razón para elegir este número como cota superior del tamaño del autómata es que es el número máximo de enteros diferentes que se pueden almacenar en dos *bytes* de memoria, y además se ha comprobado empíricamente que resulta adecuado ya que con cotas mayores los pasos de minimización se alargan excesivamente, y con cotas menores el número de palabras que se pueden insertar en cada etapa de construcción es muy pequeño y como consecuencia el número de etapas necesarias crece considerablemente.

Etapa de construcción	Palabras insertadas	Antes de la minimización		Después de la minimización	
		Estados	Transiciones	Estados	Transiciones
1	34.839	65.526	100.363	2.277	5.219
2	68.356	65.527	101.986	4.641	11.431
3	100.325	65.526	104.285	6.419	16.163
4	132.094	65.530	107.043	7.377	18.560
5	163.426	65.532	108.047	8.350	21.094
6	193.368	65.525	108.211	9.858	24.877
7	223.743	65.530	110.924	10.646	27.166
8	253.703	65.527	112.007	11.221	28.850
9	283.281	65.528	112.735	11.779	30.617
10	291.604	26.987	54.148	11.985	31.258

Tabla 3.1: Evolución del proceso de construcción del autómata finito acíclico determinista mínimo para el diccionario del sistema GALENA

autómata sólo es un árbol de letras antes de la primera minimización. Si el procedimiento de inserción estándar se aplica después de minimizar, es decir, cuando el autómata ya no es un árbol de letras, las inserciones pueden dar lugar a errores de construcción.

Ejemplo 3.5 Tal es el caso de la inserción que se plantea en la figura 3.5. Nuestra intención aquí es incorporar la palabra `removal` en el autómata de la figura 3.4. Si añadimos esta nueva palabra mediante un procedimiento de inserción estándar, éste aprovecha la existencia del camino $0 \xrightarrow{r} 3 \xrightarrow{e} 4 \xrightarrow{m} 5 \xrightarrow{o} 6$ para añadir el menor número posible de nuevos estados y transiciones, y la operación degenera recursivamente en la inserción de la subpalabra `val` en el subautómata que comienza en el estado 6. Aparentemente todo es correcto, pero si nos fijamos bien observaremos que de repente hemos hecho válida no sólo la palabra `removal` sino también las palabras `discoval`, `dismoval` y `recoval`, lo cual no era nuestra intención, y además dichas palabras no existen en inglés. □

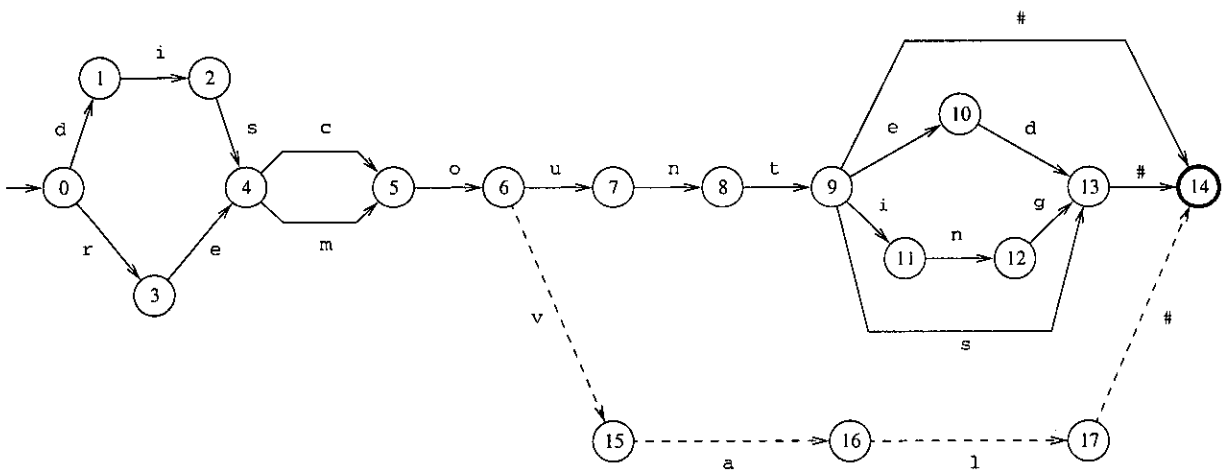


Figura 3.5: El problema de la inserción de nuevas palabras en un autómata finito acíclico determinista

La solución al problema planteado pasa por diseñar un nuevo procedimiento de inserción *especial*. Dicho procedimiento comprueba en todo momento que está haciendo uso de transiciones

cuyo estado destino tiene una única transición entrante. Si se llega a un estado con más de una transición entrante, entonces se trata de un estado al que se puede llegar también al menos por otro camino distinto al que está recorriendo el proceso de inserción en ese momento. Si esto ocurre, entonces ese estado problemático se duplica, el estado destino de la transición implicada se cambia por el estado copia, y se continúa recursivamente la inserción de la palabra en el subautómata que comienza en este nuevo estado.

Ejemplo 3.6 Como se puede observar en la figura 3.6, esta operación de duplicado de un estado podría ser necesario realizarla más de una vez durante el proceso de inserción de una misma palabra, ya que es posible que se alcancen varias veces estados con más de una transición entrante, en nuestro caso, los estados 4, 5 y 6. □

Aparentemente, esta operación de rotura y duplicación de determinadas partes del autómata contradice el objetivo perseguido, que es el de obtener el autómata mínimo. Pero a medida que se inserten más y más palabras irán apareciendo nuevos fenómenos léxicos subceptibles de ser compartidos durante la siguiente aplicación del proceso de minimización.

En cualquier caso, lo que sí es importante señalar es que este procedimiento de inserción especial es más complejo que el procedimiento de inserción estándar. Pero ocurre que si insertamos las palabras en el autómata según su orden alfabético, entonces está garantizado que sólo es necesario aplicar el procedimiento de inserción especial a la primera palabra que aparece después de cada minimización, pudiéndose realizar el resto de inserciones con el procedimiento estándar. La demostración de esta afirmación es sencilla de razonar: si después de una inserción especial, al intentar insertar una nueva palabra, aparece algún estado problemático con más de una transición entrante, necesariamente dicha inserción ha de corresponder a una palabra lexicográficamente menor a la última palabra insertada en el autómata, y dado que las palabras se insertan en orden alfabético dicha palabra no puede aparecer.

Ejemplo 3.7 Si observamos otra vez la figura 3.6, es fácil comprobar intuitivamente que la inserción de cualquier palabra lexicográficamente mayor que `removal` caerá por debajo del camino de líneas punteadas y no pasará por ningún estado problemático, y por tanto se puede insertar normalmente en el autómata como si de un árbol de letras se tratara. □

En el momento en que el autómata se llena de nuevo, se realiza una minimización, una inserción especial, y se continúa con el proceso hasta que todas las palabras del lexicón hayan sido insertadas.

3.2.2 El algoritmo de minimización de la altura

Para definir formalmente el algoritmo de minimización, es necesario introducir primero los siguientes conceptos.

Definición 3.4 Se dice que dos autómatas son *equivalentes* si y sólo si reconocen el mismo lenguaje. Se dice también que dos estados p y q de un autómata dado son *equivalentes* si y sólo si el subautómata que comienza con p como estado inicial y el que comienza con q son equivalentes. O lo que es lo mismo, si para toda palabra w tal que $p.w$ es un estado final, entonces $q.w$ es también un estado final, y viceversa. □

Definición 3.5 El concepto contrario es que dos estados p y q se dicen *distinguibles* o *no equivalentes* si y sólo si existe una palabra w tal que $p.w$ es un estado final y $q.w$ no lo es, o viceversa. □

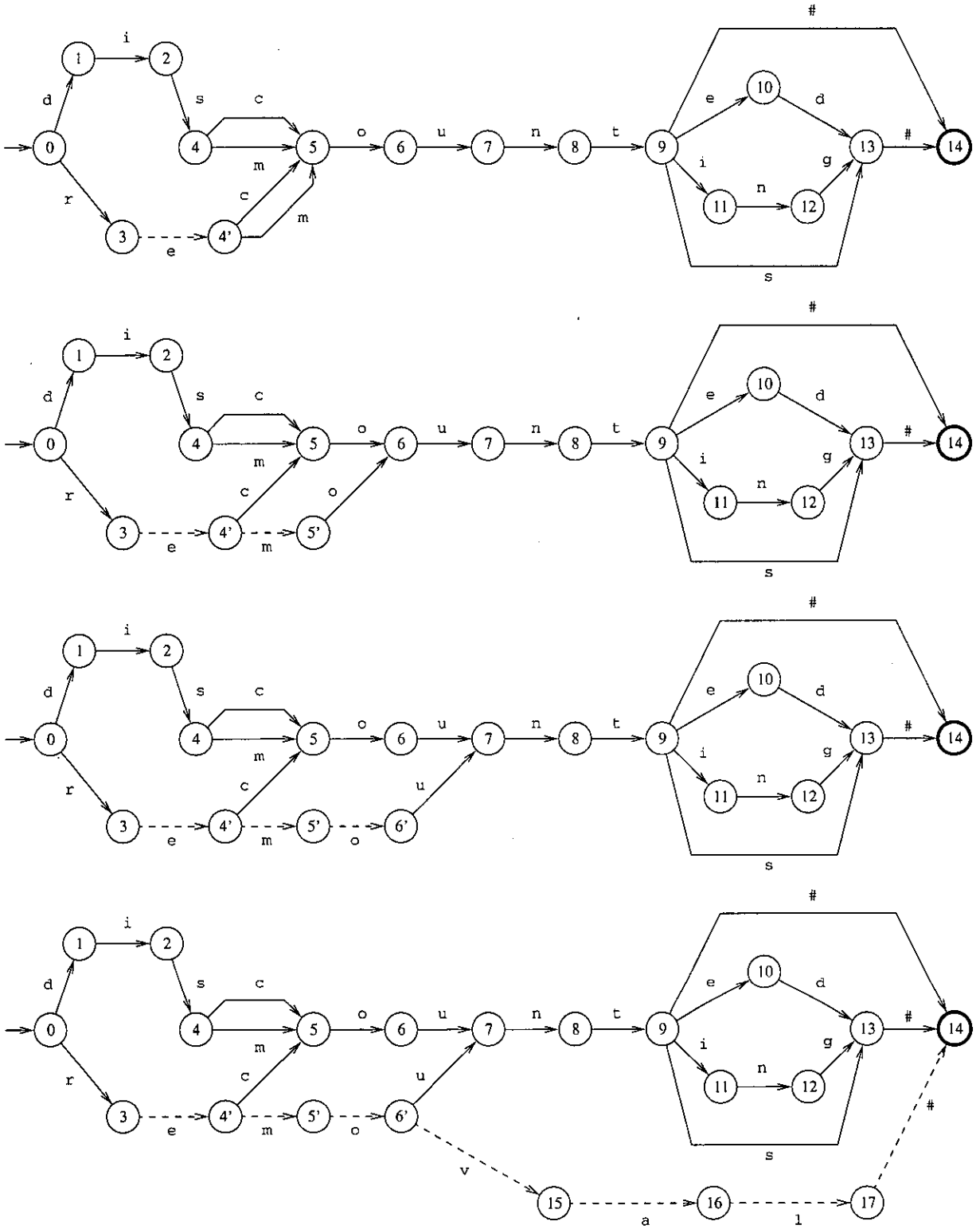


Figura 3.6: Inserción correcta de nuevas palabras en un autómata finito acíclico determinista

Definición 3.6 Si A es un autómata, existe un único autómata M, mínimo en el número de estados, que reconoce el mismo lenguaje, es decir, tal que $L(A) = L(M)$. Un autómata *mínimo* es aquél que no contiene ningún par de estados equivalentes. Y un autómata mínimo para un lenguaje dado L es por tanto aquél que contiene el menor número de estados posible de entre todos los autómatas que reconocen L. □

Definición 3.7 Si denotamos la *altura* de un estado s como $h(s)$, entonces $h(s) = max\{|w| \text{ tal que } s.w \in F\}$. Es decir, la altura de un estado s es la longitud del camino más largo de entre todos los que empiezan en dicho estado s y terminan en alguno de los estados finales. Esta función de altura establece una partición Π sobre Q, de tal manera que Π_i denota el conjunto de todos los estados de altura i. Diremos que el conjunto Π_i es *distinguible* si todos sus estados son *distinguibles*, es decir, si no contiene ningún par de estados equivalentes. □

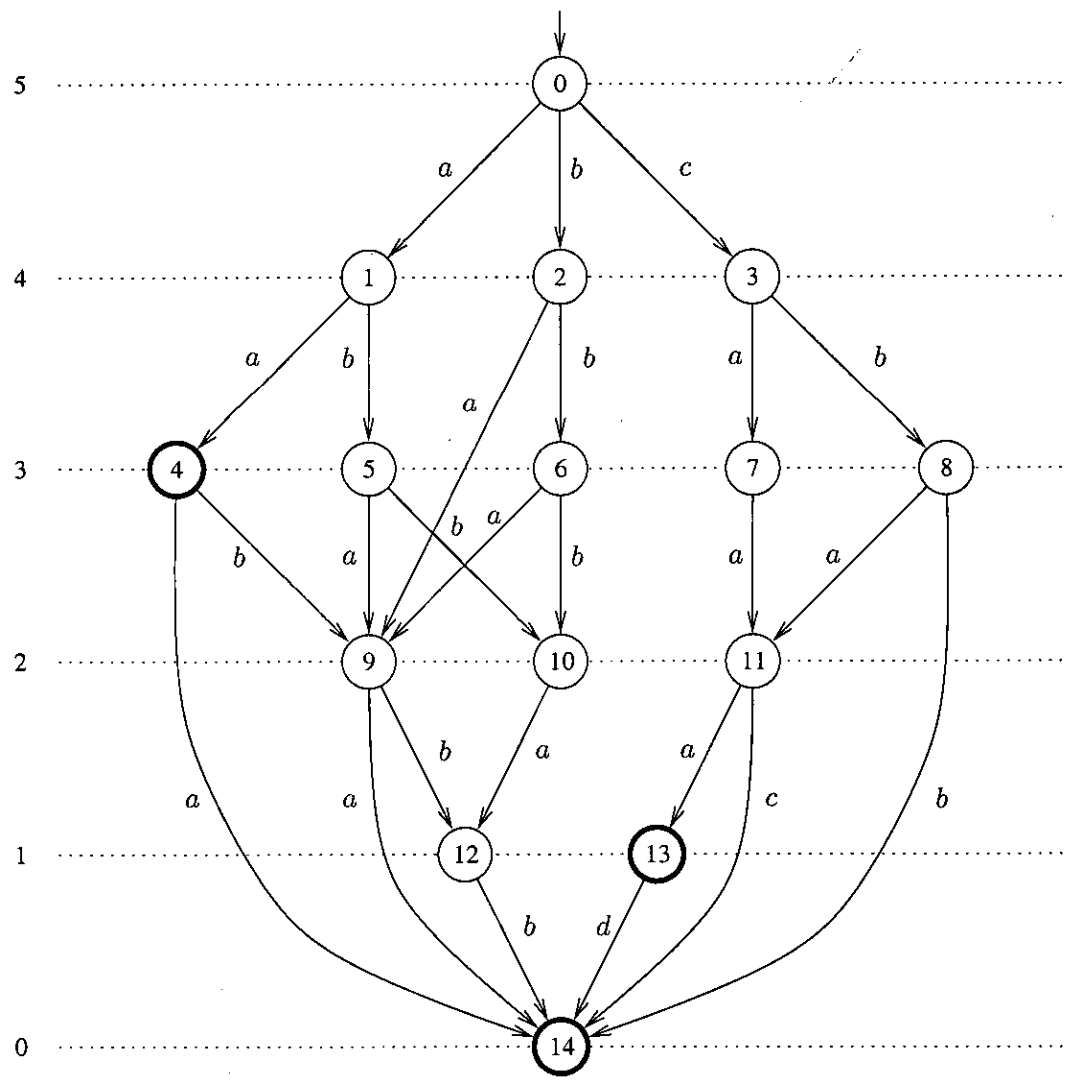


Figura 3.7: Un autómata finito acíclico determinista no mínimo

Ejemplo 3.8 La figura 3.7 muestra un autómata finito acíclico determinista, donde el estado inicial es el 0 y los estados finales son el 4, el 13 y el 14, y que por tanto reconoce el lenguaje $L = \{aa, aaa, aaba, aabbb, abaa, ababb, abbab, baa, babb, bbaa, bbabb, bbbab, caaa, caaad, caac,$

cbaa, *cbaad*, *cbac*, *cbb*}. Los estados de la misma altura aparecen conectados por una línea punteada horizontal. Este autómata no es mínimo, ya que los estados 5 y 6 de altura 3 son equivalentes. Podemos colapsar dichos estados en uno solo eliminando uno de ellos, por ejemplo el 5, y cambiando el estado destino de sus transiciones entrantes por el otro estado, es decir, cambiando la transición $1 \xrightarrow{b} 5$ por $1 \xrightarrow{b} 6$. □

Una vez que ha sido definida la altura de un estado, estamos en condiciones de introducir el siguiente teorema.

Teorema 3.1 Si todos los Π_j con $j < i$ son distinguibles, entonces dos estados p y q pertenecientes a Π_i son equivalentes si y sólo si para cualquier letra $a \in \Sigma$ la igualdad $p.a = q.a$ se verifica.

Demostración: Si dicha igualdad se verifica, los estados son equivalentes, por la propia definición de estado equivalente. Así que para la demostración de esta propiedad, basta estudiar los casos en los que dicha igualdad no se verifica, y comprobar que efectivamente los estados no son equivalentes. Entonces, dados p y q dos estados de Π_i con $p.a \neq q.a$, tenemos dos posibilidades:

1. Cuando $p.a$ y $q.a$ pertenecen al mismo Π_j. Dado que $j < i$, que el autómata es acíclico, y que por hipótesis todos los estados de Π_j son distinguibles, entonces p y q también son distinguibles.

2. Cuando $p.a \in \Pi_j$ y $q.a \in \Pi_k$, $j \neq k$. Supongamos sin pérdida de generalidad que $k < j$. Entonces, por la definición de Π, existe una palabra w de longitud j tal que $(p.a).w$ es final y $(q.a).w$ no lo es. Por tanto, los estados $p.a$ y $q.a$ son distinguibles, y entonces p y q también son distinguibles.

Este resultado se conoce también como la *propiedad de la altura.* □

El algoritmo de minimización se puede deducir ahora de manera sencilla a partir de la propiedad de la altura [Revuz 1992].

Algoritmo 3.2 Los pasos básicos del algoritmo de minimización de un autómata finito acíclico determinista son los siguientes:

```
procedure Minimizar_Autómata (Autómata) =
  begin
    Calcular Π;

    for i ← 0 to h(q0) do
      begin
        Ordenar los estados de Πi según sus transiciones;
        Colapsar los estados equivalentes
      end
  end;
```

Primero creamos la partición del conjunto de estados según su altura. Esta partición se puede calcular mediante un recorrido recursivo estándar sobre el autómata, cuya complejidad temporal es $\mathcal{O}(t)$, donde t es el número de transiciones del autómata. No obstante, si el autómata no es un árbol, se puede ganar algo de velocidad mediante una marca que indique si la altura de un estado ha sido ya calculada o no. De igual manera, los estados no útiles no tendrán ninguna altura asignada y ya se pueden eliminar durante este recorrido. Posteriormente, se procesa cada uno de los Π_i, desde $i = 0$ hasta la altura del estado inicial, ordenando los estados según sus transiciones y colapsando los estados que resulten ser equivalentes. □

Utilizando un esquema de ordenación de complejidad temporal $\mathcal{O}(f(e))$, donde e es el número de elementos a ordenar, el algoritmo 3.2 presenta una complejidad

$$\mathcal{O}(t + \sum_{i=0}^{h(q_0)} f(|\Pi_i|))$$

que en el caso de los autómatas finitos acíclicos deterministas es menor que la complejidad del algoritmo general de Hopcroft para cualquier tipo de autómatas finitos deterministas: $\mathcal{O}(n \times \log n)$, donde n es el número de estados [Hopcroft y Ullman 1979].

3.2.3 Asignación y uso de los números de indexación

Hemos visto que los autómatas finitos acíclicos deterministas son la estructura más compacta que se puede diseñar para el reconocimiento de un conjunto finito de palabras dado. Los resultados de compresión son excelentes, y el tiempo de reconocimiento es lineal respecto a la longitud de la palabra a analizar, y no depende ni del tamaño del diccionario, ni del tamaño del autómata.

Sin embargo, si detenemos el proceso de construcción del autómata en este punto, dispondremos de una estructura que solamente es capaz de indicarnos si una palabra dada pertenece o no al diccionario, y esto no es suficiente para el esquema de modelización de diccionarios que hemos desarrollado anteriormente. Dicha modelización necesita un mecanismo que transfome cada palabra en una clave numérica unívoca, y viceversa.

Definición 3.8 Esta transformación se puede llevar a cabo fácilmente si el autómata incorpora, para cada estado, un entero que indique el número de palabras que se pueden aceptar mediante el subautómata que comienza en ese estado [Lucchesi y Kowaltowski 1993]. Nos referiremos a este autómata como *autómata finito acíclico determinista numerado.* □

Ejemplo 3.9 La versión numerada del autómata de la figura 3.4 es la que se muestra en la figura 3.8. □

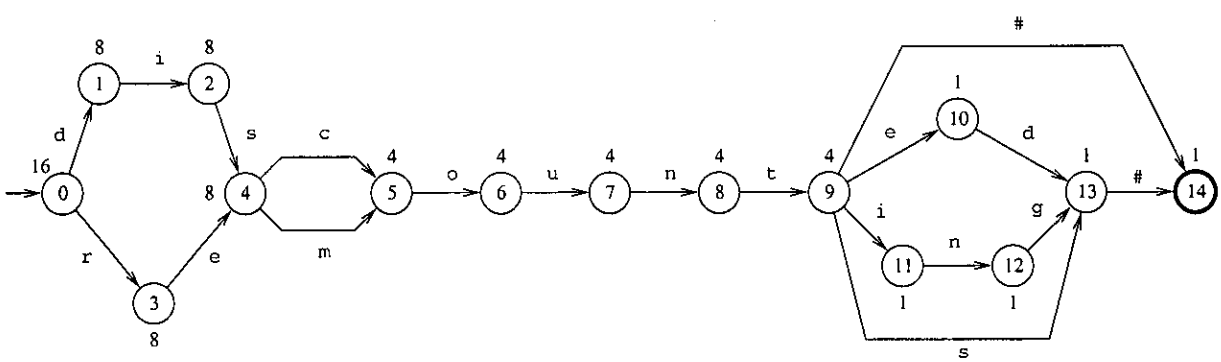


Figura 3.8: Autómata finito acíclico determinista mínimo numerado para las formas de los verbos `discount`, `dismount`, `recount` y `remount`

La asignación de los números de indexación a cada estado se puede realizar mediante un sencillo recorrido recursivo sobre el autómata, una vez que éste ha sido correctamente construido y minimizado. Por tanto, la versión definitiva de la función `Construir_Autómata` es la que se muestra a continuación.

Algoritmo 3.3 Algoritmo para la construcción de un autómata finito acíclico determinista a partir de un lexicón:

```
function Construir_Autómata (Lexicón) =
  begin
    A ← Autómata_Vacío;

    while (queden palabras del Lexicón por insertar) do
      if (A está lleno) then
        begin
          A ← Minimizar_Autómata (A);
          Inserción especial de la siguiente palabra del Lexicón en A
        end
      else
        Inserción estándar de la siguiente palabra del Lexicón en A;

    A ← Minimizar_Autómata (A);
    Asignar los números de indexación a los estados de A;
    return A
  end;
```

Este nuevo proceso de construcción del autómata completa y substituye al visto anteriormente en el algoritmo 3.1. □

Una vez que el autómata ha sido numerado, podemos ya escribir las funciones `Palabra_a_Índice` e `Índice_a_Palabra`, que son las que realizan la correspondencia uno a uno entre las palabras del diccionario y los números 1 a M, donde M es el número de total de palabras distintas aceptadas por el autómata.

Algoritmo 3.4 Pseudo-código de la función `Palabra_a_Índice`:

```
function Palabra_a_Índice (Palabra) =
  begin
    Índice ← 1;
    Estado_Actual ← Estado_Inicial;

    for i ← 1 to Longitud (Palabra) do
      if (Transición_Válida (Estado_Actual, Palabra[i])) then
        begin
          for c ← Primera_Letra to Predecesor (Palabra[i]) do
            if (Transición_Válida (Estado_Actual, c)) then
              Índice ← Índice + Estado_Actual[c].Número;
          Estado_Actual ← Estado_Actual[Palabra[i]];
        end
      else
        return palabra desconocida;

    if (Es_Estado_Final (Estado_Actual)) then
      return Índice
    else
      return palabra desconocida
  end;
```

Esta función parte con un índice igual a 1 y va transitando por el autómata desde el estado inicial utilizando cada una de las letras de la palabra a analizar. En cada uno de los estados por los que pasa el camino correspondiente a dicha palabra, el índice se va incrementando con el número de indexación del estado destino de aquellas transiciones que son lexicográficamente precedentes a la transición utilizada. Si después de procesar todos los caracteres de la palabra llegamos al estado final, entonces el índice contendrá la clave numérica de la palabra. En caso contrario, la palabra no pertenece al lexicón que se está manejando. Como consecuencia, el valor del índice no es correcto, y en su lugar la función devuelve un valor que indica que la palabra es desconocida. □

Algoritmo 3.5 Pseudo-código de la función `Índice_a_Palabra`:

```
function Índice_a_Palabra (Índice) =
  begin
    Estado_Actual ← Estado_Inicial;
    Número ← Índice;
    Palabra ← Palabra_Vacía;
    i ← 1;

    repeat
      for c ← Primera_Letra to Última_Letra do
        if (Transición_Válida (Estado_Actual, c)) then
          begin
            Estado_Auxiliar ← Estado_Actual[c];
            if (Número > Estado_Auxiliar.Número) then
              Número ← Número − Estado_Auxiliar.Número
            else
              begin
                Palabra[i] ← c;
                i ← i + 1;
                Estado_Actual ← Estado_Auxiliar;
                if (Es_Estado_Final (Estado_Actual)) then
                  Número ← Número − 1;
                exit forloop
              end
          end
    until (Número = 0);

    return Palabra
  end;
```

Esta función parte del índice y realiza las operaciones análogas a las del algoritmo 3.4, para deducir cuáles son las transiciones que dan lugar a ese índice, y a partir de esas transiciones obtiene las letras que forman la palabra que se está buscando. □

Ejemplo 3.10 En el caso del autómata numerado de la figura 3.8, la correspondencia individual de cada palabra con su índice es como sigue:

```
 1 <-> discount    2 <-> discounted    3 <-> discounting    4 <-> discounts
 5 <-> dismount    6 <-> dismounted    7 <-> dismounting    8 <-> dismounts
 9 <-> recount    10 <-> recounted    11 <-> recounting    12 <-> recounts
13 <-> remount    14 <-> remounted    15 <-> remounting    16 <-> remounts
```

Se puede observar que M, en este caso 16, corresponde efectivamente con el número de indexación del estado inicial, que es el que indica el número total de palabras que puede reconocer el autómata. □

Los requerimientos de almacenamiento y la complejidad temporal extra en el proceso de reconocimiento implicada por la incorporación de los números de indexación resulta modesta. Así que finalmente sólo nos queda hacer una pequeña referencia a las prestaciones de nuestro analizador léxico, que simplemente confirma algunos de los resultados ya comentados en la sección 2.2.2: el autómata finito acíclico determinista numerado correspondiente al diccionario GALENA consta de 11.985 estados y 31.258 transiciones; el tamaño del fichero compilado correspondiente a dicho autómata es de 3.466.121 *bytes*; el tiempo de compilación es de aproximadamente 29 segundos; y la velocidad de reconocimiento en una máquina con un procesador Pentium II a 300 MHz bajo sistema operativo Linux es de aproximadamente 40.000 palabras por segundo.

3.2.4 Algoritmos de construcción incrementales

Como hemos visto, los métodos tradicionales para la construcción de autómatas finitos acíclicos deterministas mínimos a partir de un conjunto de palabras consisten en combinar dos fases de operación: la construcción de un árbol o de un autómata parcial, y su posterior minimización. Sin embargo, existen métodos de construcción incremental, capaces de realizar las operaciones de minimización en línea, es decir, al mismo tiempo que se realizan las inserciones de las palabras en el autómata [Daciuk *et al.* 2000]. Estos métodos son mucho más rápidos, y sus requerimientos de memoria son también significativamente menores que los de los métodos descritos anteriormente.

Para construir un autómata palabra a palabra de manera incremental es necesario combinar el proceso de minimización con el proceso de inserción de nuevas palabras. Por tanto, hay dos preguntas cruciales que hay que responder:

1. ¿Qué estados, o clases de equivalencia de estados, son subceptibles de cambiar cuando se insertan nuevas palabras en el autómata?

2. ¿Existe alguna manera de minimizar el número de estados que es necesario cambiar durante la inserción de una palabra?

Como ya sabemos, si las palabras están ordenadas lexicográficamente, cuando se añade una nueva, sólo pueden cambiar los estados que se atraviesan al aceptar la palabra insertada previamente. El resto del autómata permanece inalterado, ya que la nueva palabra:

- O bien comienza con un símbolo diferente del primer símbolo de todas las palabras ya presentes en el autómata, con lo cual el símbolo inicial de la nueva palabra es situado lexicográficamente después de todos esos símbolos.

- O bien comparte algunos de los símbolos iniciales de la palabra añadida previamente. En este caso, el algoritmo localiza el último estado en el camino de ese prefijo común y crea una nueva rama desde ese estado. Esto es debido a que el símbolo que etiqueta la nueva transición será lexicográficamente mayor que los símbolos del resto de transiciones salientes que ya existen en ese estado.

Por tanto, cuando la palabra previa es prefijo de la nueva palabra a insertar, los únicos estados que pueden cambiar son los estados del camino de reconocimiento de la palabra previa que no están en el camino del prefijo común. La nueva palabra puede tener una finalización igual a la

de otras palabras ya insertadas, lo cual implica la necesidad de crear enlaces a algunas partes del autómata. Esas partes, sin embargo, no se verán afectadas.

A continuación describimos el algoritmo de construcción incremental a partir de un conjunto de palabras ordenado lexicográficamente. Dicho algoritmo se apoya en una estructura denominada *Registro* que mantiene en todo momento un único representante de cada una de las clases de equivalencia de estados del autómata. Es decir, el *Registro* constituye en sí mismo el autómata mínimo en cada instante.

Algoritmo 3.6 El algoritmo de construcción incremental de un autómata finito acíclico determinista, a partir de un conjunto de palabras ordenado lexicográficamente, consta básicamente de dos funciones: la función principal *Construir_Autómata_Incremental* y la función *Reemplazar_o_Registrar*. Los pasos de la función principal son los siguientes:

```
function Construir_Autómata_Incremental (Lexicón) =
  begin
    Registro ← ∅;

    while (queden palabras del Lexicón por insertar) do
      begin
        Palabra ← siguiente palabra del Lexicón en orden lexicográfico;
        Prefijo_Común ← Prefijo_Común (Palabra);
        Último_Estado ← q0.Prefijo_Común;
        Sufijo_Actual ← Palabra[(Longitud (Prefijo_Común) + 1) ... Longitud (Palabra)];
        if (Tiene_Hijos (Último_Estado)) then
          Registro ← Reemplazar_o_Registrar (Último_Estado, Registro);
        Añadir_Sufijo (Último_Estado, Sufijo_Actual);
      end;

    Registro ← Reemplazar_o_Registrar (q0, Registro);
    return Registro
  end;
```

El esquema de la función *Reemplazar_o_Registrar* es como sigue:

```
function Reemplazar_o_Registrar (Estado, Registro) =
  begin
    Hijo ← Último_Hijo (Estado);
    if (Tiene_Hijos (Hijo)) then
      Registro ← Reemplazar_o_Registrar (Hijo, Registro);
    if (∃ q ∈ Q : q ∈ Registro ∧ q ≡ Hijo) then
      begin
        Último_Hijo (Estado) ← q;
        Eliminar (Hijo)
      end
    else
      Registro ← Registro ∪ Hijo;
    return Registro
  end;
```

El lazo principal del algoritmo lee las palabras y establece qué parte de cada palabra está ya en el autómata, es decir, el *Prefijo_Común*, y qué parte no está, es decir, el *Sufijo_Actual*. Un

paso importante es determinar cuál es el último estado en el camino del prefijo común, que en el algoritmo se denota como *Último_Estado*. Si *Último_Estado* ya tiene hijos, quiere decir que no todos los estados del camino de la palabra añadida previamente están en el camino del prefijo común. En ese caso, mediante la función *Reemplazar_o_Registrar*, hacemos que el proceso de minimización trabaje sobre los estados del camino de la palabra añadida previamente que no están en el camino del prefijo común. Posteriormente, añadimos desde el *Último_Estado* una cadena de nuevos estados capaz de reconocer el *Sufijo_Actual*.

La función *Prefijo_Común* busca el prefijo más largo de la palabra a insertar que es prefijo de alguna palabra ya insertada. La función *Añadir_Sufijo* crea una nueva rama que extiende el autómata, la cual representa el sufijo no encontrado de la palabra que se va a insertar. La función *Último_Hijo* devuelve una referencia al estado alcanzado por la última transición en orden lexicográfico que sale del estado argumento. Dado que los datos de entrada están ordenados, dicha transición es la transición añadida más recientemente en el estado argumento, durante la inserción de la palabra previa. La función *Tiene_Hijos* es cierta si y sólo si el estado argumento tiene transiciones salientes.

La función *Reemplazar_o_Registrar* trabaja efectivamente sobre el último hijo del estado argumento. Dicho argumento es el último estado del camino del prefijo común, o bien el estado inicial del autómata en la última llamada de la función principal. Es necesario que el estado argumento cambie su última transición en aquellos casos en los que el hijo va a ser reemplazado por otro estado equivalente ya registrado. En primer lugar, la función se llama recursivamente a sí misma hasta alcanzar el final del camino de la palabra insertada previamente. Nótese que cada vez que se encuentra un estado con más de un hijo, siempre se elige el último. La longitud limitada de las palabras garantiza el fin de la recursividad. Por tanto, al volver de cada llamada recursiva, se comprueba si ya existe en el registro algún estado equivalente al estado actual. Si es así, el estado actual se reemplaza por el estado equivalente encontrado en el registro. Si no, el estado actual se registra como representante de una nueva clase de equivalencia. Es importante destacar que la función *Reemplazar_o_Registrar* sólo procesa estados pertenecientes al camino de la palabra insertada previamente, y que esos estados no se vuelven a procesar. □

Durante la construcción, los estados del autómata o están en el registro o están en el camino de la última palabra insertada. Todos los estados del registro son estados que formarán parte del autómata mínimo resultante. Así pues, el número de estados durante la construcción es siempre menor que el número de estados del autómata resultante más la longitud de la palabra más larga. Por tanto, la complejidad espacial del algoritmo es $\mathcal{O}(n)$, es decir, la cantidad de memoria que precisa el algoritmo es proporcional a n, el número final de estados del autómata mínimo. Respecto al tiempo de ejecución, éste es dependiente de la estructura de datos implementada para manejar las búsquedas de estados equivalentes y las inserciones de nuevos estados representantes en el registro. Utilizando criterios de búsqueda e inserción basados en el número de transiciones salientes de los estados, en sus alturas y en sus números de indexación, los cuales se pueden también calcular dinámicamente, la complejidad temporal se puede rebajar hasta $\mathcal{O}(\log n)$.

Con este algoritmo incremental, el tiempo de construcción del autómata correspondiente al diccionario del sistema GALENA se reduce considerablemente: de 29 segundos a 2,5 segundos, en una máquina con un procesador Pentium II a 300 MHz bajo sistema operativo Linux. La incorporación de la información relativa a las etiquetas, lemas y probabilidades consume un tiempo extra aproximado de 4,5 segundos, lo que hace un tiempo de compilación total de 7 segundos, tal y como se había indicado en la sección 2.2.2.

En el mismo trabajo, los autores proponen también un método incremental para la construcción de autómatas finitos acíclicos deterministas mínimos a partir de conjuntos de palabras no ordenados [Daciuk *et al.* 2000]. Este método se apoya también en la clonación o duplicación de los estados que se van volviendo conflictivos a medida que aparecen las nuevas

palabras. Por esta razón, la construcción incremental a partir de datos desordenados es más lenta y consume más memoria que la construcción incremental a partir de esos mismos datos ordenados. No obstante, el método puede ser de gran utilidad en situaciones donde el tamaño de los diccionarios es tan elevado que el propio proceso de ordenación podría resultar problemático.

3.3 Otros métodos de análisis léxico

Los procesos productivos o derivativos presentes en todos los idiomas constituyen una gran fuente de complicaciones para el análisis morfológico. Debido a ellos, siempre existirán multitud de palabras que no estarán incluidas en un diccionario morfológico estático, sin importar lo grande o preciso que éste sea. En otras palabras, se podría decir que el tamaño del lexicón de cualquier lengua es virtualmente infinito. Inevitablemente, éste es un problema al que deben enfrentarse los etiquetadores. Tal y como veremos con detalle en los capítulos siguientes, todo etiquetador debe incluir un módulo de tratamiento de palabras desconocidas, independientemente de los mecanismos que tenga integrados para el análisis léxico y para el acceso al diccionario.

Pero efectivamente, muchas de las palabras desconocidas serán formas derivadas a partir de una raíz y de una serie de reglas de inflexión. Por ejemplo, es muy probable que la palabra `reanalizable` no esté incluida en un diccionario dado, pero parece claro que esta palabra es parte del lenguaje, en el sentido de que puede ser utilizada, es comprensible y podría ser fácilmente deducida a partir de la palabra `analizar`[10] y de reglas de derivación tales como la incorporación de prefijos o la transformación de los verbos en sus respectivos adjetivos de calidad.

Además, no siempre es necesario manejar una información tan particular para cada palabra concreta como puede ser su frecuencia o su probabilidad. Fuera de los paradigmas de aproximación estocástica al NLP, la mayoría de las aplicaciones utilizan sólo la etiqueta, o como mucho la etiqueta y el lema. En esta última sección, por tanto, enumeramos brevemente otros posibles métodos de análisis léxico que han sido desarrollados.

La aproximación quizás más importante, y que ha servido de base para multitud de trabajos relacionados con la construcción automática de analizadores léxicos a partir de la especificación de las reglas morfológicas de flexión y derivación de una determinada lengua, es el modelo de *morfología de dos niveles*[11] [Koskenniemi 1983]. Este modelo se basa en la distinción tradicional que hacen los lingüistas entre, por un lado, el conjunto teórico de morfemas y el orden en el cual pueden ocurrir, y por otro lado, las formas alternativas que presentan dichos morfemas dentro del contexto fonológico en el que aparecen. Esto establece la diferencia entre lo que se denomina la *cadena superficial* del lenguaje (por ejemplo, la palabra `quiero`), y su correspondiente *cadena léxica* o *teórica* (en este caso, `quero` = `quer` + `o`). Por tanto, `quer` y `quier` se consideran alomorfos o formas alternativas del mismo morfema. El modelo de Koskenniemi es de dos niveles en el sentido de que cualquier palabra se puede representar mediante una correspondencia directa letra a letra entre su cadena de superficie y la cadena teórica subyacente.

Ejemplo 3.11 Para la palabra inglesa `skies`, podemos asumir que existe una raíz `sky` y una terminación de plural `-es`, salvo que, como ocurre en inglés con todos los sustantivos terminados en `y`, la `y` se realiza como una `i`, y por tanto las cadenas teórica y superficial serían, respectivamente:

```
s k y - e s
s k i 0 e s
```

En este ejemplo, el `-` inidica el límite de morfemas, y el `0` un caracter nulo o no existente. □

[10] Suponiendo que ésta sí está presente en el lexicón.

[11] *Two-level morphology.*

Posibles representaciones de las correspondencias letra a letra son `s:s` (no necesaria, porque por defecto cada letra se corresponde consigo misma, y por tanto dicha relación se puede expresar simplemente como `s`), o bien `-:0`, o también `y:i`. Respecto a esta última relación habría que especificar lo siguiente:

- No es cierto que cualquier `y` se corresponda con una `i`. Esto sólo es válido para el proceso de formación de plural que estamos considerando.
- En este caso concreto, la `y` se corresponde sólo con una `i`, y con ninguna otra letra, ni siquiera consigo misma.

El modelo de morfología de dos niveles incorpora un componente más que permite expresar este tipo de restricciones mediante reglas de transformación. Por ejemplo, esta relación *si y sólo si* podría denotarse mediante una regla de doble flecha como la siguiente:

```
y:i <=> _ -: e: s: ;
```

Un conjunto dado de reglas de este tipo se puede transformar en un traductor de estado finito [Karttunen y Beesley 1992]. Un *traductor de estado finito* no es más que un autómata finito que, en lugar de caracteres simples, incorpora correspondencias letra a letra como etiquetas de sus transiciones, permitiendo transformar la cadena superficial en la cadena teórica, y viceversa.

Ejemplo 3.12 La figura 3.9 representa un traductor de estado finito capaz de asociar cada una de las formas verbales del verbo `leave` con su lema y su correspondiente etiqueta, es decir: `leave` con `leave+VB`, `leaves` con `leave+VBZ` y `left` con `leave+VBD`[12]. □

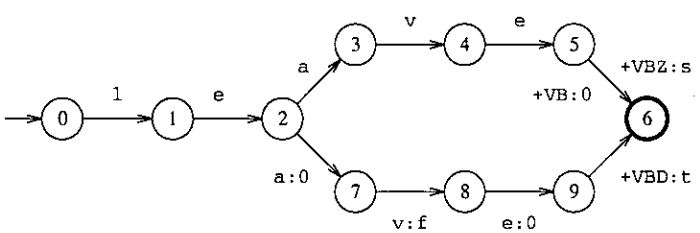


Figura 3.9: Traductor de estado finito para el verbo `leave`

Para cada palabra, existe en el traductor un camino que contiene la forma flexionada de la cadena superficial y el par lema-etiqueta de la cadena teórica. Dicho camino se puede encontrar utilizando como clave de entrada cualquiera de las dos cadenas. Esta característica de procesamiento bidireccional de los traductores finitos hace que el modelo de morfología de dos niveles resulte válido tanto para el análisis morfológico, como para la generación de formas.

Sin embargo, para los traductores, la noción de determinismo normalmente tiene sentido si se considera sólo una de las direcciones de aplicación. Si un traductor presenta este tipo de determinismo unidireccional, se dice que es o bien *secuencial por arriba*[13] si es determinista en la dirección de la cadena superficial, o bien *secuencial por abajo*[14] si es determinista en la dirección de la cadena teórica. Por ejemplo, el traductor de la figura 3.9 es secuencial por arriba, pero no secuencial por abajo, ya que desde el estado 2 con el símbolo de entrada a se puede transitar a dos estados diferentes, el 3 y el 7.

[12] La notación de las etiquetas es una adaptación del juego de etiquetas utilizado en el corpus BROWN [Francis y Kučera 1982]: `VB` significa verbo en infinitivo, `VBZ` es verbo en tercera persona del singular y `VBD` es verbo en tiempo pasado.

[13] *Sequential upward.*

[14] *Sequential downward.*

No obstante, al procesar una cadena dada, es probable que uno de los caminos termine con éxito y el otro no. Es decir, esta ambigüedad local podría ser resuelta algunas transiciones más tarde, y de hecho en este caso es así, ya que la relación que establece el traductor de la figura 3.9 es no ambigua en ambas direcciones. Si la relación es no ambigua en una dirección dada, y si todas las ambigüedades locales en esa dirección se pueden resolver a sí mismas en un número finito de pasos, se dice que el traductor es *secuenciable*, es decir, se puede construir un traductor secuencial equivalente en la misma dirección [Roche y Schabes 1995, Mohri 1995].

Ejemplo 3.13 La ambigüedad local del traductor de la figura 3.9 se resuelve en el estado 5. Por tanto, se puede incorporar un camino desde el estado inicial hasta el estado 5 que transforma la secuencia ave en la cadena vacía, lo cual, tal y como se muestra en la figura 3.10, permite construir un traductor secuencial por abajo, pero sólo en esa dirección, ya que ahora existen dos arcos con a en la cadena superior partiendo del estado 5. □

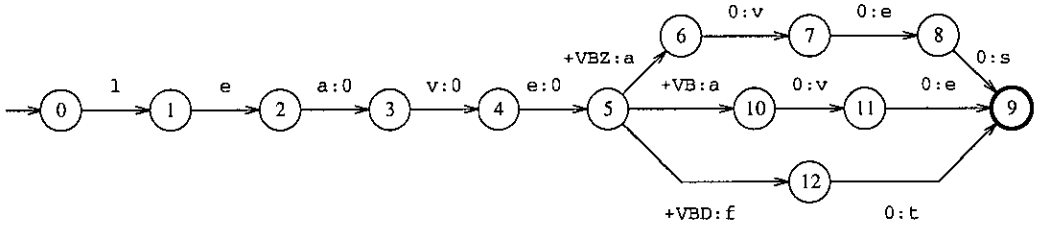


Figura 3.10: Traductor de estado finito secuencial por abajo para el verbo leave

Los traductores secuenciales se pueden extender para permitir que emitan desde los estados finales una cadena de salida adicional (*traductores subsecuenciales*) o un número finito p de cadenas de salida (*traductores p-subsecuenciales*). Estos últimos son los que permiten realizar el manejo de las ambigüedades léxicas presentes en los lenguajes naturales [Mohri 1995].

Y todos ellos, mediante la incorporación de una información numérica extra en cada estado, pueden pasar de ser simples conversores de una cadena en otra[15] a incorporar también la funcionalidad de conversión de una cadena en un peso numérico[16]. Típicamente, ese nuevo dato numérico podría ser una probabilidad logarítmica, de tal manera que la probabilidad de emisión de una palabra dada se calcularía simplemente sumando el dato de todos los estados por los que pasa el camino de procesamiento de dicha palabra en el traductor. Ésta es la forma de integrar probabilidades de emisión para las palabras en los traductores finitos [Mohri 1997].

De entre los múltiples trabajos basados en el modelo de Koskenniemi, destaca la herramienta MMORPH [Russell y Petitpierre 1995], un compilador de reglas morfológicas de dos niveles de libre distribución, desarrollado en el marco del proyecto MULTEXT[17].

Otros métodos de diseño de analizadores léxicos, que también han sido aplicados con éxito al idioma español, incluyen las aproximaciones basadas en unificación [Moreno 1991, Moreno y Goñi 1995, González Collar *et al.* 1995] y las basadas en árboles de decisión [Triviño 1995, Triviño y Morales 2000].

[15] Traductores *string-to-string*.

[16] Traductores *string-to-weight*.

[17] *Multilingual Text Tools and Corpora* es una iniciativa de la Comisión Europea (bajo los programas *Investigación e Ingeniería Lingüística*, *Copernicus*, y *Lenguas Regionales y Minoritarias*), de la Fundación Científica Nacional de los Estados Unidos, del Fondo Francófono para la Investigación, del CNRS (Centro Nacional francés para la Investigación Científica) y de la Universidad de Provenza. MULTEXT comprende una serie de proyectos cuyos objetivos son el desarrollo de estándares y especificaciones para la codificación y el procesamiento de textos, y para el desarrollo de herramientas, *corpora* y recursos lingüísticos bajo esos estándares. Los trabajos han sido realizados también en estrecha colaboración con los proyectos EAGLES y CRATER, e incluyen herramientas y recursos para la mayoría de los idiomas europeos.

Parte II

Sistemas de etiquetación tradicionales

Capítulo 4

Modelos de Markov ocultos (HMM,s)

Cuando se abordó inicialmente el problema de la etiquetación de textos en lenguaje natural, se comenzó diseñando manualmente reglas de etiquetación que intentaban describir el comportamiento del lenguaje humano. Sin embargo, en el momento en el que aparecen disponibles grandes cantidades de textos electrónicos, muchos de ellos incluso etiquetados, las aproximaciones estocásticas adquieren gran importancia en el proceso de etiquetación. Son muchas las ventajas de los etiquetadores estocásticos frente a los etiquetadores construidos manualmente. Además del hecho de que evitan la laboriosa construcción manual de las reglas, también capturan automáticamente información útil que el hombre podría no apreciar.

En este capítulo estudiaremos un método estocástico comúnmente denominado aproximación con modelos de Markov ocultos o HMM,s[1]. A pesar de sus limitaciones, los HMM,s y sus variantes son todavía la técnica más ampliamente utilizada en el proceso de etiquetación del lenguaje natural, y son generalmente referenciados como una de las técnicas más exitosas para dicha tarea. Los procesos de Markov fueron desarrollados inicialmente por Andrei A. Markov, alumno de Chebyshev, y su primera utilización tuvo un objetivo realmente lingüístico: modelizar las secuencias de letras de las palabras en la literatura rusa [Markov 1913]. Posteriormente, los modelos de Markov evolucionaron hasta convertirse en una herramienta estadística de propósito general. La suposición subyacente de las técnicas de etiquetación basadas en HMM,s es que la tarea de la etiquetación se puede formalizar como un proceso paramétrico aleatorio, cuyos parámetros se pueden estimar de una manera precisa y perfectamente definida.

Comenzaremos con un repaso de la teoría de cadenas de Markov, seguiremos con una extensión de las ideas mostradas hacia los modelos de Markov ocultos y centraremos nuestra atención en las tres cuestiones fundamentales que surgen a la hora de utilizar un HMM:

1. La evaluación de la probabilidad de una secuencia de observaciones, dado un HMM específico.

2. La determinación de la secuencia de estados más probable, dada una secuencia de observaciones específica.

3. La estimación de los parámetros del modelo para que éste se ajuste a las secuencias de observaciones disponibles.

Una vez que estos tres problemas fundamentales sean resueltos, veremos cómo se pueden aplicar inmediatamente los HMM,s al problema de la etiquetación del lenguaje natural, y estudiaremos con detalle las múltiples variantes de implementación que surgen al hacerlo.

[1] *Hidden Markov Models.*

4.1 Procesos de Markov de tiempo discreto

Consideremos un sistema que en cada instante de tiempo se encuentra en un determinado estado. Dicho estado pertenece a un conjunto finito de estados Q. De momento, para no complicar demasiado la notación, supondremos que existe una correspondencia entre el conjunto de estados Q y el conjunto de números enteros $\{1, 2, \ldots, N\}$, y etiquetaremos cada estado con uno de esos números, tal y como se ve en la figura 4.1, donde $N = 5$. Regularmente, transcurrido un espacio de tiempo discreto, el sistema cambia de estado (posiblemente volviendo al mismo), de acuerdo con un conjunto de probabilidades de transición asociadas a cada uno de los estados del modelo. Los instantes de tiempo asociados a cada cambio de estado se denotan como $t = 1, 2, \ldots, T$, y el estado actual en el instante de tiempo t se denota como q_t. En general, una descripción probabilística completa del sistema requeriría la especificación del estado actual, así como de todos los estados precedentes. Sin embargo, las cadenas de Markov presentan dos características muy importantes:

1. **Propiedad del horizonte limitado**. Esta propiedad permite truncar la dependencia probabilística del estado actual y considerar, no todos los estados precedentes, sino únicamente un subconjunto finito de ellos. En general, una cadena de Markov de orden n es la que utiliza n estados previos para predecir el siguiente estado. Por ejemplo, para el caso de las cadenas de Markov de tiempo discreto de primer orden tenemos que:

$$P(q_t = j | q_{t-1} = i, q_{t-2} = k, \ldots) = P(q_t = j | q_{t-1} = i). \tag{4.1}$$

2. **Propiedad del tiempo estacionario**. Esta propiedad nos permite considerar sólo aquellos procesos en los cuales la parte derecha de (4.1) es independiente del tiempo[2]. Esto nos lleva a una matriz $A = \{a_{ij}\}$ de probabilidades de transición entre estados de la forma

$$a_{ij} = P(q_t = j | q_{t-1} = i) = P(j|i), \qquad 1 \leq i, j \leq N,$$

independientes del tiempo, pero con las restricciones estocásticas estándar:

$$a_{ij} \geq 0, \qquad \forall i, j,$$

$$\sum_{j=1}^{N} a_{ij} = 1, \qquad \forall i.$$

Sin embargo, es necesario especificar también el vector $\pi = \{\pi_i\}$, que almacena la probabilidad que tiene cada uno de los estados de ser el estado inicial[3]:

$$\pi_i = P(q_1 = i), \qquad \pi_i \geq 0, \qquad 1 \leq i \leq N,$$

$$\sum_{i=1}^{N} \pi_i = 1.$$

[2]Como ya hemos visto en el capítulo 1, este es el punto central del famoso argumento de Chomsky contra el uso de los modelos de Markov para el procesamiento de lenguaje natural.

[3]La necesidad de este vector podría evitarse especificando que la cadena de Markov comienza siempre en un estado inicial extra, e incluyendo en la matriz A las probabilidades de transición de este nuevo estado: las del vector π para aquellas celdas que lo tienen como estado origen, y 0 para las que lo tienen como estado destino.

A un proceso estocástico que satisface estas características se le puede llamar un *modelo de Markov observable*, porque su salida es el conjunto de estados por los que pasa en cada instante de tiempo, y cada uno de estos estados se corresponde con un suceso observable.

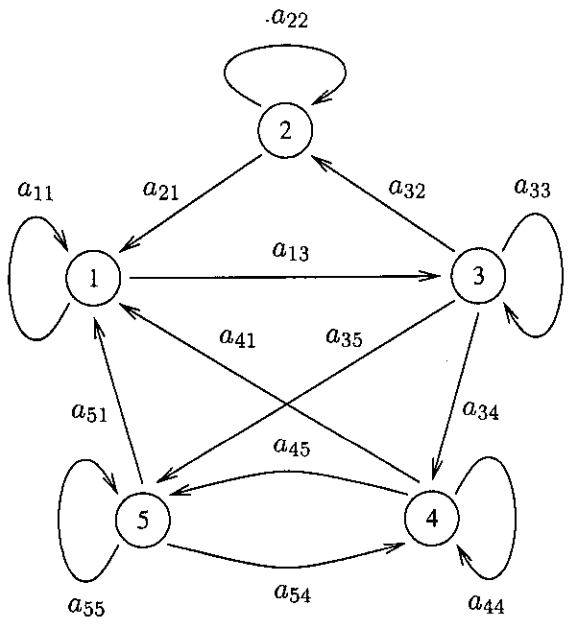


Figura 4.1: Una cadena de Markov de 5 estados con algunas transiciones entre ellos

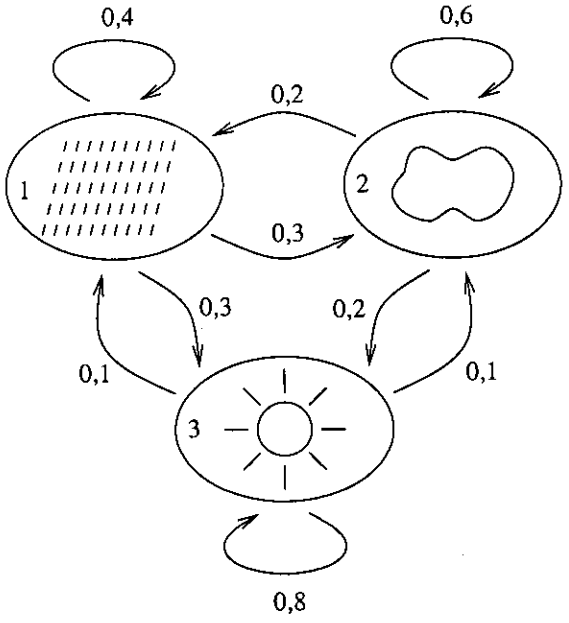


Figura 4.2: Un modelo de Markov para la evolución del clima

Ejemplo 4.1 Para remarcar las ideas, supongamos que una vez al día, por ejemplo al mediodía, se observa el clima, y las posibles observaciones resultan ser las siguientes: (1) precipitación de lluvia o nieve, (2) nublado, o (3) soleado. Supongamos también que el clima del día t se caracteriza por uno solo de esos tres estados, que la matriz A de las probabilidades de transición entre estados es

$$A = \{a_{ij}\} = \begin{bmatrix} 0,4 & 0,3 & 0,3 \\ 0,2 & 0,6 & 0,2 \\ 0,1 & 0,1 & 0,8 \end{bmatrix}$$

y que los tres estados tienen la misma probabilidad de ser el estado inicial, es decir, $\pi_i = \frac{1}{3}$, $1 \leq i \leq 3$. Tal y como se ve en la figura 4.2, no hemos hecho más que especificar un modelo de Markov para describir la evolución del clima. □

La probabilidad de observar una determinada secuencia de estados, es decir, la probabilidad de que una secuencia finita de T variables aleatorias con dependencia de primer orden, $q_1, q_2, \ldots, q_T$, tome unos determinados valores, $o_1, o_2, \ldots, o_T$, con todos los $o_i \in \{1, 2, \ldots, N\}$, es sencilla de obtener. Simplemente calculamos el producto de las probabilidades que figuran en las aristas del grafo o en la matriz de transiciones:

$$P(o_1, o_2, \ldots, o_T) =$$

$$= P(q_1 = o_1)P(q_2 = o_2|q_1 = o_1)P(q_3 = o_3|q_2 = o_2)\ldots P(q_T = o_T|q_{T-1} = o_{T-1}) =$$

$$= \pi_{o_1} \prod_{t=1}^{T-1} a_{o_t o_{t+1}}.$$

Ejemplo 4.2 Utilizando el modelo de Markov para la evolución del clima, podemos calcular la probabilidad de observar la secuencia de estados 2, 3, 2, 1, como sigue:

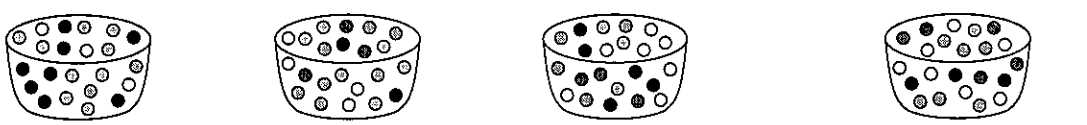
$$P(2,3,2,1) = P(2)P(3|2)P(2|3)P(1|2) = \pi_2 \times a_{23} \times a_{32} \times a_{21} = \frac{1}{3} \times 0,2 \times 0,1 \times 0,2 = 0,00133333.$$

Ésta es, por tanto, la probabilidad que corresponde a la secuencia *nublado-soleado-nublado-lluvia.* □

4.2 Extensión a modelos de Markov ocultos

En la sección anterior hemos considerado modelos de Markov en los cuales cada estado se corresponde de manera determinista con un único suceso observable. Es decir, la salida en un estado dado no es aleatoria, sino que es siempre la misma. Esta modelización puede resultar demasiado restrictiva a la hora de ser aplicada a problemas reales. En esta sección extendemos el concepto de modelos de Markov de tal manera que es posible incluir aquellos casos en los cuales la observación es una función probabilística del estado. El modelo resultante, denominado modelo de Markov oculto, es un modelo doblemente estocástico, ya que uno de los procesos no se puede observar directamente (está oculto), sino que se puede observar sólo a través de otro conjunto de procesos estocásticos, los cuales producen la secuencia de observaciones.

Ejemplo 4.3 Para ilustrar los conceptos básicos de un modelo de Markov oculto consideremos el sistema de urnas y bolas de la figura 4.3. El modelo consta de un gran número N de urnas colocadas dentro de una habitación. Cada urna contiene en su interior un número también grande de bolas. Cada bola tiene un único color, y el número de colores distintos para ellas es M. El proceso físico para obtener las observaciones es como sigue. Una persona se encuentra dentro de la habitación y, siguiendo un procedimiento totalmente aleatorio, elige una urna inicial, saca una bola, nos grita en alto su color y devuelve la bola a la urna. Posteriormente elige otra urna de acuerdo con un proceso de selección aleatorio asociado a la urna actual, y efectúa la extracción de una nueva bola. Este paso se repite un determinado número de veces, de manera que todo el proceso genera una secuencia finita de colores, la cual constituye la salida observable del modelo, quedando oculta para nosotros la secuencia de urnas que se ha seguido.



Urna 1	Urna 2	Urna 3		Urna N
$P(color\ 1) = b_{11}$	$P(color\ 1) = b_{21}$	$P(color\ 1) = b_{31}$	...	$P(color\ 1) = b_{N1}$
$P(color\ 2) = b_{12}$	$P(color\ 2) = b_{22}$	$P(color\ 2) = b_{32}$	...	$P(color\ 2) = b_{N2}$
$P(color\ 3) = b_{13}$	$P(color\ 3) = b_{23}$	$P(color\ 3) = b_{33}$	...	$P(color\ 3) = b_{N3}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$
$P(color\ M) = b_{1M}$	$P(color\ M) = b_{2M}$	$P(color\ M) = b_{3M}$	...	$P(color\ M) = b_{NM}$

Figura 4.3: Un modelo de Markov oculto de N urnas de bolas y M colores

Este proceso se modeliza de forma que cada urna se corresponde con un estado del HMM, la elección de las urnas se realiza de acuerdo con la matriz de probabilidades de transición entre estados asociada al modelo, y para cada uno de esos estados existe una función de probabilidad de los colores perfectamente definida. Es importante señalar también que lo que hace que una urna sea distinta de otra es la cantidad de bolas de cada color que tiene en su interior, pero los

colores de las bolas son los mismos para todas las urnas. Por tanto, una observación aislada de un determinado color de bola no nos dice automáticamente de qué urna viene esa bola. □

El ejemplo de las urnas y las bolas puede identificarse de una manera muy intuituva con el problema de la etiquetación de las palabras de un texto en lenguaje natural. En este caso concreto, las bolas representan a las palabras, las urnas representan a las distintas etiquetas o categorías gramaticales a las que pertenecen las palabras, y las secuencias de observaciones representan a las frases del texto. Una misma palabra puede estar en urnas distintas, de ahí la ambigüedad léxica, y puede estar también varias veces en la misma urna, de ahí que las probabilidades de las palabras que están dentro de una misma urna puedan ser distintas.

4.3 Elementos de un modelo de Markov oculto

Los ejemplos vistos anteriormente nos proporcionan una idea de lo que son los modelos de Markov ocultos y de cómo se pueden aplicar a escenarios prácticos reales, tales como la etiquetación de las palabras de un texto en lenguaje natural. Pero antes de tratar nuestro caso particular, vamos a definir formalmente cuáles son los elementos de un modelo de Markov oculto.

Definición 4.1 Un *HMM* se caracteriza por la 5-tupla (Q, V, π, A, B), donde:

1. Q es el conjunto de estados del modelo. Aunque los estados permanecen ocultos, para la mayoría de las aplicaciones prácticas se conocen *a priori*. Por ejemplo, para el caso de la etiquetación de palabras, cada etiqueta del juego de etiquetas utilizado sería un estado. Generalmente los estados están conectados de tal manera que cualquiera de ellos se puede alcanzar desde cualquier otro en un solo paso, aunque existen muchas otras posibilidades de interconexión. Los estados se etiquetan como $\{1, 2, \ldots, N\}$, y el estado actual en el instante de tiempo t se denota como q_t. El uso de instantes de tiempo es apropiado, por ejemplo, en la aplicación de los HMM,s al procesamiento de voz. No obstante, para el caso de la etiquetación de palabras, no hablaremos de los instantes de tiempo, sino de las posiciones de cada palabra dentro de la frase.

2. V es el conjunto de los distintos sucesos que se pueden observar en cada uno de los estados. Por tanto, cada uno de los símbolos individuales que un estado puede emitir se denota como $\{v_1, v_2, \ldots, v_M\}$. En el caso del modelo de las urnas y las bolas, M es el número de colores distintos y cada $v_k, 1 \leq k \leq M$, es un color distinto. En el caso de la etiquetación de palabras, M es el tamaño del diccionario y cada $v_k, 1 \leq k \leq M$, es una palabra distinta.

3. $\pi = \{\pi_i\}$, es la distribución de probabilidad del estado inicial. Por tanto,

$$\pi_i = P(q_1 = i), \qquad \pi_i \geq 0, \qquad 1 \leq i \leq N,$$

$$\sum_{i=1}^{N} \pi_i = 1.$$

4. $A = \{a_{ij}\}$ es la distribución de probabilidad de las transiciones entre estados, es decir,

$$a_{ij} = P(q_t = j | q_{t-1} = i) = P(j|i), \qquad 1 \leq i, j \leq N, \qquad 1 \leq t \leq T,$$

$$\sum_{j=1}^{N} a_{ij} = 1, \qquad \forall i.$$

Para el caso de un modelo con estados totalmente conexos en un solo paso, tenemos que $a_{ij} > 0$ para todo i, j. Para otro tipo de HMM,s podría existir algún $a_{ij} = 0$.

5. $B = \{b_j(v_k)\}$ es la distribución de probabilidad de los sucesos observables, es decir,

$$b_j(v_k) = P(o_t = v_k|q_t = j) = P(v_k|j), \quad b_j(v_k) \geq 0, \quad 1 \leq j \leq N,\ 1 \leq k \leq M,\ 1 \leq t \leq T.$$

$$\sum_{k=1}^{M} b_j(v_k) = 1, \qquad \forall j.$$

Este conjunto de probabilidades se conoce también con el nombre de *conjunto de probabilidades de emisión.*

Tal y como hemos visto, una descripción estricta de un HMM necesita la especificación de Q y V, el conjunto de estados y el conjunto de los símbolos que forman las secuencias de observación, respectivamente, y la especificación de los tres conjuntos de probabilidades π, A y B. Pero dado que los dos primeros conjuntos normalmente se conocen *a priori*, y que en todo caso los tres últimos elementos del HMM ya incluyen de manera explícita al resto de los parámetros, utilizaremos la notación compacta

$$\mu = (\pi, A, B)$$

a lo largo de las siguientes secciones, y ésta seguirá siendo una representación completa de un HMM. □

Dada una especificación de un HMM, podemos simular un proceso estocástico que genere secuencias de datos, donde las leyes de producción de dichas secuencias están perfectamente definidas en el modelo. Sin embargo, es mucho más interesante tomar una secuencia de datos, suponer que efectivamente ha sido generada por un HMM, y estudiar distintas propiedades sobre ella, tales como su probabilidad, o la secuencia de estados más probable por la que ha pasado. La siguiente sección se ocupa de este tipo de cuestiones.

4.4 Las tres preguntas fundamentales al usar un HMM

Existen tres preguntas fundamentales que debemos saber responder para poder utilizar los HMM,s en aplicaciones reales. Estas tres preguntas son las siguientes:

1. Dada una secuencia de observaciones $O = (o_1, o_2, \ldots, o_T)$ y dado un modelo $\mu = (\pi, A, B)$, ¿cómo calculamos de una manera eficiente $P(O|\mu)$, es decir, la probabilidad de dicha secuencia dado el modelo?

2. Dada una secuencia de observaciones $O = (o_1, o_2, \ldots, o_T)$ y dado un modelo $\mu = (\pi, A, B)$, ¿cómo elegimos la secuencia de estados $S = (q_1, q_2, \ldots, q_T)$ óptima, es decir, la que mejor *explica* la secuencia de observaciones?

3. Dada una secuencia de observaciones $O = (o_1, o_2, \ldots, o_T)$, ¿cómo estimamos los parámetros del modelo $\mu = (\pi, A, B)$ para maximizar $P(O|\mu)$?, es decir, ¿cómo podemos encontrar el modelo que mejor *explica* los datos observados?

Normalmente, los problemas que manejaremos en la práctica no son tan sencillos como el modelo de las urnas y las bolas, es decir, lo normal es que no conozcamos *a priori* los parámetros del modelo y tengamos que estimarlos a partir de los datos observados. La secuencia de observación utilizada para ajustar los parámetros del modelo se denomina secuencia de *entrenamiento*, ya que a partir de ella se entrena el HMM. Ésta es la problemática que se aborda en la tercera pregunta.

La segunda pregunta trata el problema de descubrir la parte oculta del modelo, es decir, trata sobre cómo podemos adivinar qué camino ha seguido la cadena de Markov. Este camino oculto se puede utilizar para clasificar las observaciones. En el caso de la etiquetación de un texto en lenguaje natural, dicho camino nos ofrece la secuencia de etiquetas más probable para las palabras del texto.

La primera pregunta representa el problema de la evaluación de una secuencia de observaciones dado el modelo. La resolución de este problema nos proporciona la probabilidad de que la secuencia haya sido generada por ese modelo. Un ejemplo práctico puede ser suponer la existencia de varios HMM,s compitiendo entre sí. La evaluación de la secuencia en cada uno de ellos nos permitirá elegir el modelo que mejor encaja con los datos observados.

A continuación se describen formalmente todos los algoritmos matemáticos que son necesarios para responder a estas tres preguntas.

4.4.1 Cálculo de la probabilidad de una observación

Dada una secuencia de observaciones $O = (o_1, o_2, \ldots, o_T)$ y un modelo $\mu = (\pi, A, B)$, queremos calcular de una manera eficiente $P(O|\mu)$, es decir, la probabilidad de dicha secuencia dado el modelo. La forma más directa de hacerlo es enumerando primero todas las posibles secuencias de estados de longitud T, el número de observaciones. Existen N^T secuencias distintas. Considerando una de esas secuencias, $S = (q_1, q_2, \ldots, q_T)$, la probabilidad de dicha secuencia de estados es

$$P(S|\mu) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \ldots a_{q_{T-1} q_T} \tag{4.2}$$

y la probabilidad de observar O a través de la secuencia S es

$$P(O|S, \mu) = \prod_{t=1}^{T} P(o_t|q_t, \mu) = b_{q_1}(o_1)\, b_{q_2}(o_2) \ldots b_{q_T}(o_T) \tag{4.3}$$

La probabilidad conjunta de O y S, es decir, la probabilidad de que O y S ocurran simultáneamente, es simplemente el producto de (4.2) y (4.3), es decir,

$$P(O, S|\mu) = P(S|\mu)\, P(O|S, \mu)$$

Por tanto, la probabilidad de O dado el modelo se obtiene sumando esta probabilidad conjunta para todas las posibles secuencias S:

$$P(O|\mu) = \sum_{S} P(S|\mu)\, P(O|S, \mu).$$

Sin embargo, si calculamos $P(O|\mu)$ de esta forma, necesitamos realizar exactamente $(2T-1)N^T$ multiplicaciones y $N^T - 1$ sumas, es decir, un total de $2TN^T - 1$ operaciones. Estas cifras no son computacionalmente admisibles ni siquiera para valores pequeños de N y T. Por ejemplo, para $N = 5$ estados y $T = 100$ observaciones, el número de operaciones es del orden de 10^{72}. El secreto para evitar esta complejidad tan elevada está en utilizar técnicas de programación dinámica, con el fin de recordar los resultados parciales, en lugar de recalcularlos. A continuación se presentan varios procedimientos que calculan $P(O|\mu)$ utilizando dichas técnicas.

4.4.1.1 Procedimiento hacia adelante

Consideremos la variable $\alpha_t(i)$ definida como

$$\alpha_t(i) = P(o_1, o_2, \ldots, o_t, q_t = i|\mu),$$

es decir, la probabilidad conjunta de obtener $o_1, o_2, \ldots, o_t$, la secuencia parcial de observaciones hasta el instante de tiempo t, y de estar en el estado i en ese instante de tiempo t, dado el modelo μ. Los valores de $\alpha_t(i)$, para los distintos estados y para los distintos instantes de tiempo, se pueden obtener iterativamente, y pueden ser utilizados para calcular $P(O|\mu)$ mediante los pasos del siguiente algoritmo.

Algoritmo 4.1 Cálculo hacia adelante de la probabilidad de una secuencia de observaciones:

1. Inicialización:
$$\alpha_1(i) = \pi_i \, b_i(o_1), \qquad 1 \leq i \leq N.$$

2. Recurrencia:
$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) \, a_{ij}\right] b_j(o_{t+1}), \qquad t = 1, 2, \ldots, T-1, \qquad 1 \leq j \leq N. \tag{4.4}$$

3. Terminación:
$$P(O|\mu) = \sum_{i=1}^{N} \alpha_T(i).$$

El paso 1 inicializa las N probabilidades $\alpha_1(i)$ como la probabilidad conjunta de que i sea el estado inicial y genere la primera observación o_1. El paso de recurrencia, que es el punto central de este algoritmo, se ilustra en la figura 4.4 (a). Esta figura muestra cómo el estado j se puede alcanzar en el instante de tiempo $t+1$ desde los N posibles estados i del instante de tiempo t. Para cada uno de esos estados, dado que $\alpha_t(i)$ es la probabilidad conjunta de haber observado $o_1, o_2, \ldots, o_t$, y de estar en el estado i en el instante t, $\alpha_t(i) \, a_{ij}$ será la probabilidad conjunta de haber observado $o_1, o_2, \ldots, o_t$, y de haber alcanzado el estado j en el instante $t+1$ desde el estado i del instante t. Sumando todos estos productos sobre los N posibles estados i del instante t, y multiplicando esa suma por $b_j(o_{t+1})$, la probabilidad de que el estado j emita el símbolo o_{t+1}, obtenemos $\alpha_{t+1}(j)$, es decir, la probabilidad conjunta de obtener $o_1, o_2, \ldots, o_{t+1}$, la secuencia parcial de observaciones hasta el instante de tiempo $t+1$, y de estar en el estado j en ese instante de tiempo $t+1$. El cálculo de la ecuación (4.4) se realiza para los N estados j de un mismo instante de tiempo t, y para todos los instantes de tiempo $t = 1, 2, \ldots, T-1$. Finalmente, el paso 3 obtiene $P(O|\mu)$ sumando el valor de las N variables $\alpha_T(i)$. □

Los cálculos globales involucrados en este proceso requieren del orden de N^2T operaciones, tal y como se muestra en el enrejado de la figura 4.4 (b). Más concretamente, se trata de $N(N+1)(T-1) + N$ multiplicaciones y $N(N-1)(T-1) + N - 1$ sumas, es decir, un total de $2(T-1)N^2 + 2N - 1$ operaciones. Para $N = 5$ estados y $T = 100$ observaciones, el número total de operaciones es aproximadamente 5.000, que comparado con 10^{72} supone un ahorro de 69 órdenes de magnitud.

4.4.1.2 Procedimiento hacia atrás

De manera similar, podemos considerar la variable $\beta_t(i)$ definida como

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots, o_T | q_t = i, \mu),$$

es decir, la probabilidad de la secuencia de observación parcial desde el instante de tiempo $t+1$ hasta el final, dado que el estado en el instante de tiempo t es i y dado el modelo μ. Nuevamente, podemos resolver las variables $\beta_t(i)$ y calcular el valor de $P(O|\mu)$ de manera recurrente mediante los pasos del siguiente algoritmo.

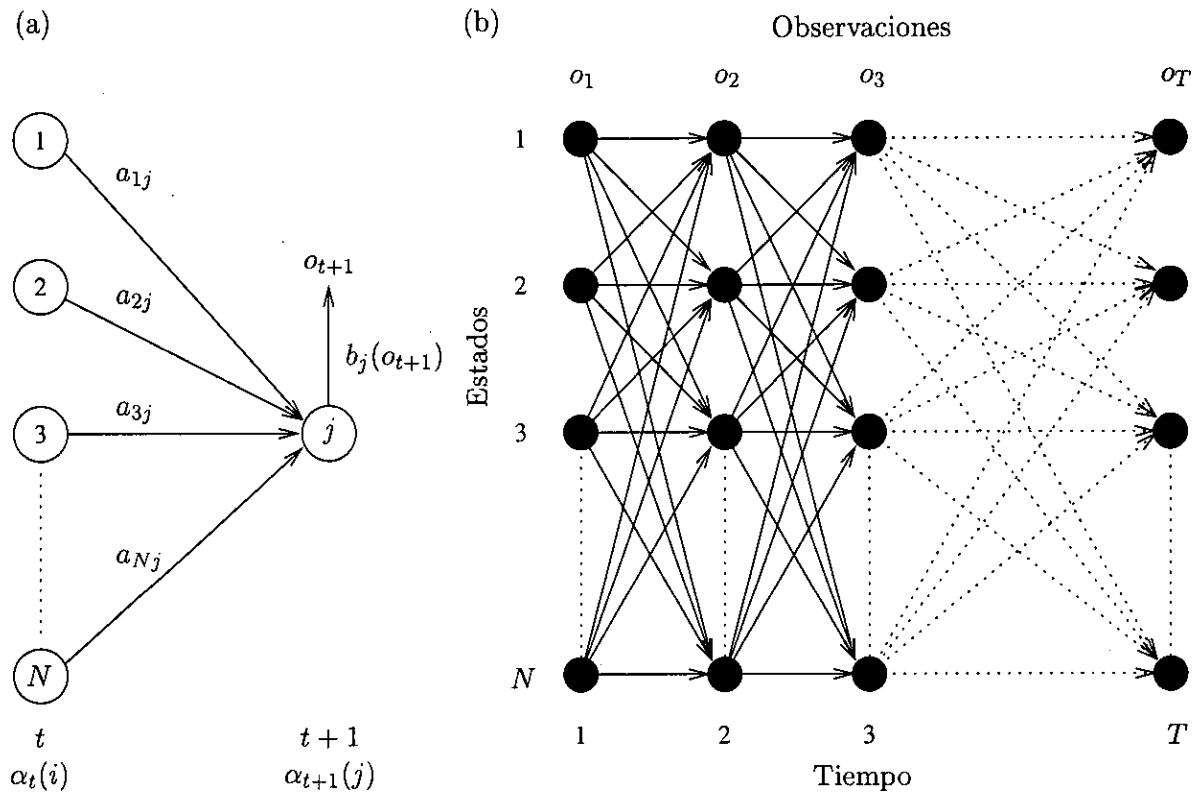


Figura 4.4: (a) Detalle de la secuencia de operaciones necesarias para el cálculo hacia adelante de la variable $\alpha_{t+1}(j)$ y (b) implementación genérica del cálculo hacia adelante de la variable $\alpha_t(i)$ mediante un enrejado de T observaciones y N estados

Algoritmo 4.2 Cálculo hacia atrás de la probabilidad de una secuencia de observaciones:

1. Inicialización:

$$\beta_T(i) = 1, \qquad 1 \leq i \leq N.$$

2. Recurrencia:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, \beta_{t+1}(j)\, b_j(o_{t+1}), \qquad t = T-1, T-2, \ldots, 1, \qquad 1 \leq i \leq N.$$

3. Terminación:

$$P(O|\mu) = \sum_{i=1}^{N} \beta_1(i)\, \pi_i\, b_i(o_1).$$

El paso de inicialización asigna 1 a las N variables $\beta_T(i)$. El paso de recurrencia, tal y como se muestra en la figura 4.5, calcula el valor de $\beta_t(i)$ en función de las N variables $\beta_{t+1}(j)$ del instante siguiente, en función de las probabilidades de transición entre estados a_{ij}, y en función de las probabilidades de emisión del símbolo o_{t+1} desde los N estados j. Finalmente, el paso 3 obtiene $P(O|\mu)$ sumando el valor de las N variables $\beta_1(i)$ multiplicado por la probabilidad de que el estado i sea el estado inicial y por la probabilidad de que emita el primer símbolo de observación o_1. □

Una vez más, el cálculo de las variables $\beta_t(i), 1 \leq t \leq T, 1 \leq i \leq N$, requiere del orden de N^2T operaciones y se puede resolver sobre un enrejado similar al de la figura 4.4 (b). Por tanto,

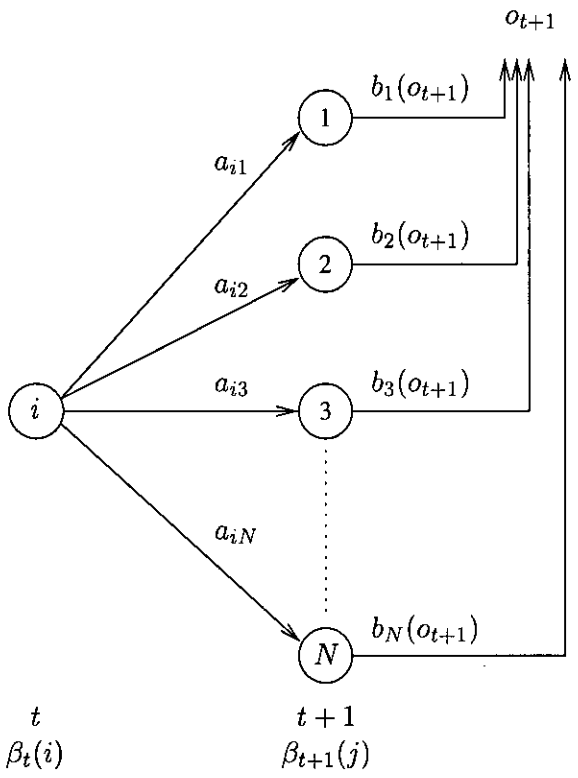


Figura 4.5: Detalle de la secuencia de operaciones necesarias para el cálculo hacia atrás de $\beta_t(i)$

la verdadera razón para haber introducido este procedimiento hacia atrás es que los cálculos involucrados en él son de vital importancia para resolver las preguntas fundamentales 2 y 3 sobre los HMM,s, es decir, el cálculo de la secuencia de estados óptima y la estimación de los parámetros del modelo, tal y como veremos a continuación.

4.4.2 Elección de la secuencia de estados más probable

El segundo problema ha sido definido vagamente como el problema de encontrar la secuencia de estados óptima que mejor *explica* las observaciones. Debido a esta definición informal del problema, podrían existir varias formas de abordarlo, es decir, se podrían considerar diferentes criterios para esa optimación. Uno de ellos podría ser la elección de los estados que son *individualmente* más probables en cada instante de tiempo. Para implementar este criterio, podemos considerar la cantidad $\gamma_t(i)$ definida como

$$\gamma_t(i) = P(q_t = i|O, \mu), \tag{4.5}$$

es decir, la probabilidad de estar en el estado i en el instante t, dada la secuencia de obsevaciones O y dado el modelo μ. Dicha cantidad se puede obtener a partir de

$$\gamma_t(i) = P(q_t = i|O, \mu) = \frac{P(q_t = i, O|\mu)}{P(O|\mu)} = \frac{P(q_t = i, O|\mu)}{\sum_{j=1}^{N} P(q_t = j, O|\mu)} = \frac{\alpha_t(i)\,\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\,\beta_t(j)} \tag{4.6}$$

donde $\alpha_t(i)$ almacena la probabilidad de la observación parcial $o_1, o_2, \ldots, o_t$, y $\beta_t(i)$ almacena la probabilidad de la observación parcial $o_{t+1}, o_{t+2}, \ldots, o_T$, dado el estado i en el instante t.

Utilizando $\gamma_t(i)$ podemos obtener q_t^*, el estado individualmente más probable en el instante t, como

$$q_t^* = \arg \max_{1 \leq i \leq N} [\gamma_t(i)], \qquad 1 \leq t \leq T. \tag{4.7}$$

La ecuación (4.7) maximiza el número esperado de estados correctos, eligiendo el estado más probable en cada instante t. Sin embargo, podrían surgir problemas con la secuencia de estados resultante. Por ejemplo, en un HMM con alguna transición entre estados de probabilidad cero ($a_{ij} = 0$ para algún i y algún j), podría ocurrir que esos estados i y j aparecieran contiguos en la secuencia óptima, cuando de hecho esa secuencia ni siquiera sería una secuencia válida. Esto es debido a que la solución que nos proporciona la ecuación (4.7) determina simplemente el estado más probable en cada instante, sin tener en cuenta la probabilidad de la secuencia de estados resultante.

Una posible solución al problema anterior es modificar el criterio de optimalidad para considerar, por ejemplo, la secuencia que maximiza el número esperado de pares de estados correctos (q_{t-1}, q_t), o de triplas de estados correctos (q_{t-2}, q_{t-1}, q_t), etc. Aunque estos criterios podrían ser razonables para algunas aplicaciones, el criterio más ampliamente utilizado consiste en encontrar la mejor secuencia considerando globalmente todos los instantes de tiempo, es decir, la secuencia de estados $S = (q_1, q_2, \ldots, q_T)$ que maximiza $P(S|O, \mu)$, lo cual es equivalente a maximizar $P(S, O|\mu)$. Existe un procedimiento formal y eficiente, basado también en técnicas de programación dinámica, para obtener esa secuencia S. Dicho procedimiento es el algoritmo de Viterbi [Viterbi 1967, Forney 1973].

4.4.2.1 Algoritmo de Viterbi

Para encontrar la secuencia de estados más probable, $S = (q_1, q_2, \ldots, q_T)$, dada la observación $O = (o_1, o_2, \ldots, o_T)$, consideramos la variable $\delta_t(i)$ definida como

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1, q_2, \ldots, q_{t-1}, q_t = i, o_1, o_2, \ldots, o_t|\mu),$$

es decir, $\delta_t(i)$ almacena la probabilidad del mejor camino que termina en el estado i, teniendo en cuenta las t primeras observaciones. Se demuestra fácilmente que

$$\delta_{t+1}(j) = \left[\max_{1 \leq i \leq N} \delta_t(i)\, a_{ij}\right] b_j(o_{t+1}). \tag{4.8}$$

Una vez calculadas las $\delta_t(i)$ para todos los estados y para todos los instantes de tiempo, la secuencia de estados se construye realmente hacia atrás a través de una traza que recuerda el argumento que maximizó la ecuación (4.8) para cada instante t y para cada estado j. Esta traza se almacena en las correspondientes variables $\psi_t(j)$. La descripción completa del algoritmo es como sigue.

Algoritmo 4.3 Cálculo de la secuencia de estados más probable para una secuencia de observaciones dada (algoritmo de Viterbi):

1. Inicialización:

$$\delta_1(i) = \pi_i\, b_i(o_1), \qquad 1 \leq i \leq N.$$

2. Recurrencia:

$$\delta_{t+1}(j) = \left[\max_{1 \leq i \leq N} \delta_t(i)\, a_{ij}\right] b_j(o_{t+1}), \qquad t = 1, 2, \ldots, T-1, \qquad 1 \leq j \leq N. \tag{4.9}$$

$$\psi_{t+1}(j) = \arg \max_{1 \leq i \leq N} \delta_t(i)\, a_{ij}, \qquad t = 1, 2, \ldots, T-1, \qquad 1 \leq j \leq N.$$

3. Terminación:

$$q_T^* = \arg \max_{1 \le i \le N} \delta_T(i).$$

4. Construcción hacia atrás de la secuencia de estados:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \qquad t = T-1, T-2, \ldots, 1.$$

El algoritmo de Viterbi es similar al cálculo hacia adelante de la probabilidad de una observación que vimos en el algoritmo 4.1. Las únicas diferencias reseñables son que el sumatorio de la ecuación (4.4) se ha cambiado por la maximización de la ecuación (4.9), y que se ha añadido el paso final para construir hacia atrás la secuencia de estados. En todo caso, la complejidad del algoritmo es del orden de N^2T operaciones y se puede resolver también sobre un enrejado similar al de la figura 4.4 (b). □

Por supuesto, durante los cálculos del algoritmo de Viterbi se podrían obtener empates. Si esto ocurre, la elección del camino se realizaría aleatoriamente. Por otra parte, existen a menudo aplicaciones prácticas en las cuales se utiliza no sólo la mejor secuencia de estados, sino las n mejores secuencias. Para todos estos casos, se podría considerar una implementación del algoritmo de Viterbi que devolviera varias secuencias de estados.

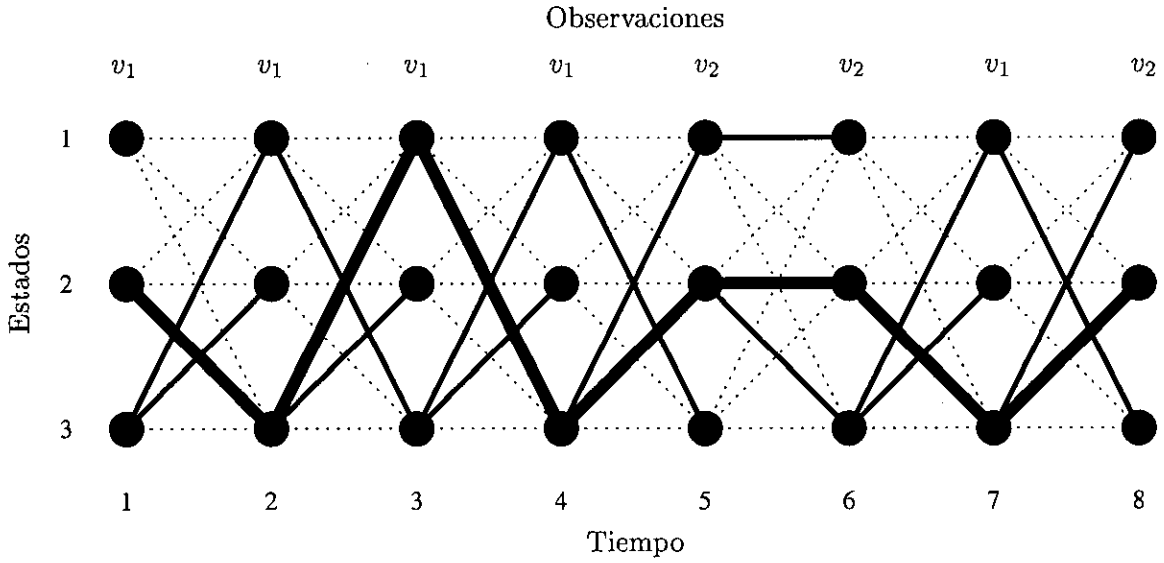


Figura 4.6: Ejemplo de ejecución del algoritmo de Viterbi

Ejemplo 4.4 A continuación presentamos un ejemplo de ejecución del algoritmo de Viterbi. Supongamos un HMM con tres estados y dos símbolos de observación, $\mu = (Q, V, \pi, A, B)$, donde

$$Q = \{1, 2, 3\}, \qquad V = \{v_1, v_2\}, \qquad \pi = \begin{bmatrix} 0,25 \\ 0,50 \\ 0,25 \end{bmatrix},$$

$$A = \begin{bmatrix} 0,25 & 0,25 & 0,50 \\ 0 & 0,25 & 0,75 \\ 0,50 & 0,50 & 0 \end{bmatrix} \quad y \quad B = \begin{bmatrix} 0,50 & 0,50 \\ 0,25 & 0,75 \\ 0,75 & 0,25 \end{bmatrix}.$$

Los cálculos para encontrar la secuencia de estados más probable dada la observación

$$O = (v_1, v_1, v_1, v_1, v_2, v_2, v_1, v_2)$$

de longitud $T = 8$, son los siguientes:

$$\delta_1(1) = \pi_1\, b_1(v_1) = (0,25)(0,50)$$
$$\delta_1(2) = \pi_2\, b_2(v_1) = (0,50)(0,25)$$
$$\delta_1(3) = \pi_3\, b_3(v_1) = (0,25)(0,75)$$

$$\delta_2(1) = \max\left[\delta_1(1)\,a_{11}, \delta_1(2)\,a_{21}, \underline{\delta_1(3)\,a_{31}}\right] b_1(v_1) = (0,25)\,(0,50)^2\,(0,75) \quad \psi_2(1) = 3$$
$$\delta_2(2) = \max\left[\delta_1(1)\,a_{12}, \delta_1(2)\,a_{22}, \underline{\delta_1(3)\,a_{32}}\right] b_2(v_1) = (0,25)^2\,(0,50)\,(0,75) \quad \psi_2(2) = 3$$
$$\delta_2(3) = \max\left[\delta_1(1)\,a_{13}, \underline{\delta_1(2)\,a_{23}}, \delta_1(3)\,a_{33}\right] b_3(v_1) = (0,25)\,(0,50)\,(0,75)^2 \quad \psi_2(3) = 2$$

$$\delta_3(1) = \max\left[\delta_2(1)\,a_{11}, \delta_2(2)\,a_{21}, \underline{\delta_2(3)\,a_{31}}\right] b_1(v_1) = (0,25)\,(0,50)^3\,(0,75)^2 \quad \psi_3(1) = 3$$
$$\delta_3(2) = \max\left[\delta_2(1)\,a_{12}, \delta_2(2)\,a_{22}, \underline{\delta_2(3)\,a_{32}}\right] b_2(v_1) = (0,25)^2\,(0,50)^2\,(0,75)^2 \quad \psi_3(2) = 3$$
$$\delta_3(3) = \max\left[\underline{\delta_2(1)\,a_{13}}, \delta_2(2)\,a_{23}, \delta_2(3)\,a_{33}\right] b_3(v_1) = (0,25)\,(0,50)^3\,(0,75)^2 \quad \psi_3(3) = 1$$

$$\delta_4(1) = \max\left[\delta_3(1)\,a_{11}, \delta_3(2)\,a_{21}, \underline{\delta_3(3)\,a_{31}}\right] b_1(v_1) = (0,25)\,(0,50)^5\,(0,75)^2 \quad \psi_4(1) = 3$$
$$\delta_4(2) = \max\left[\delta_3(1)\,a_{12}, \delta_3(2)\,a_{22}, \underline{\delta_3(3)\,a_{32}}\right] b_2(v_1) = (0,25)^2\,(0,50)^4\,(0,75)^2 \quad \psi_4(2) = 3$$
$$\delta_4(3) = \max\left[\underline{\delta_3(1)\,a_{13}}, \delta_3(2)\,a_{23}, \delta_3(3)\,a_{33}\right] b_3(v_1) = (0,25)\,(0,50)^4\,(0,75)^3 \quad \psi_4(3) = 1$$

$$\delta_5(1) = \max\left[\delta_4(1)\,a_{11}, \delta_4(2)\,a_{21}, \underline{\delta_4(3)\,a_{31}}\right] b_1(v_2) = (0,25)\,(0,50)^6\,(0,75)^3 \quad \psi_5(1) = 3$$
$$\delta_5(2) = \max\left[\delta_4(1)\,a_{12}, \delta_4(2)\,a_{22}, \underline{\delta_4(3)\,a_{32}}\right] b_2(v_2) = (0,25)\,(0,50)^5\,(0,75)^4 \quad \psi_5(2) = 3$$
$$\delta_5(3) = \max\left[\underline{\delta_4(1)\,a_{13}}, \delta_4(2)\,a_{23}, \delta_4(3)\,a_{33}\right] b_3(v_2) = (0,25)^2\,(0,50)^6\,(0,75)^2 \quad \psi_5(3) = 1$$

$$\delta_6(1) = \max\left[\underline{\delta_5(1)\,a_{11}}, \delta_5(2)\,a_{21}, \delta_5(3)\,a_{31}\right] b_1(v_2) = (0,25)^2\,(0,50)^7\,(0,75)^3 \quad \psi_6(1) = 1$$
$$\delta_6(2) = \max\left[\delta_5(1)\,a_{12}, \underline{\delta_5(2)\,a_{22}}, \delta_5(3)\,a_{32}\right] b_2(v_2) = (0,25)^2\,(0,50)^5\,(0,75)^5 \quad \psi_6(2) = 2$$
$$\delta_6(3) = \max\left[\delta_5(1)\,a_{13}, \underline{\delta_5(2)\,a_{23}}, \delta_5(3)\,a_{33}\right] b_3(v_2) = (0,25)^2\,(0,50)^5\,(0,75)^5 \quad \psi_6(3) = 2$$

$$\delta_7(1) = \max\left[\delta_6(1)\,a_{11}, \delta_6(2)\,a_{21}, \underline{\delta_6(3)\,a_{31}}\right] b_1(v_1) = (0,25)^2\,(0,50)^7\,(0,75)^5 \quad \psi_7(1) = 3$$
$$\delta_7(2) = \max\left[\delta_6(1)\,a_{12}, \delta_6(2)\,a_{22}, \underline{\delta_6(3)\,a_{32}}\right] b_2(v_1) = (0,25)^3\,(0,50)^6\,(0,75)^5 \quad \psi_7(2) = 3$$
$$\delta_7(3) = \max\left[\delta_6(1)\,a_{13}, \underline{\delta_6(2)\,a_{23}}, \delta_6(3)\,a_{33}\right] b_3(v_1) = (0,25)^2\,(0,50)^5\,(0,75)^7 \quad \psi_7(3) = 2$$

$$\delta_8(1) = \max\left[\delta_7(1)\,a_{11}, \delta_7(2)\,a_{21}, \underline{\delta_7(3)\,a_{31}}\right] b_1(v_2) = (0,25)^2\,(0,50)^7\,(0,75)^7 \quad \psi_8(1) = 3$$
$$\delta_8(2) = \max\left[\delta_7(1)\,a_{12}, \delta_7(2)\,a_{22}, \underline{\delta_7(3)\,a_{32}}\right] b_2(v_2) = (0,25)^2\,(0,50)^6\,(0,75)^8 \quad \psi_8(2) = 3$$
$$\delta_8(3) = \max\left[\underline{\delta_7(1)\,a_{13}}, \delta_7(2)\,a_{23}, \delta_7(3)\,a_{33}\right] b_3(v_2) = (0,25)^3\,(0,50)^8\,(0,75)^5 \quad \psi_8(3) = 1$$

En cada paso aparecen subrayados los términos máximos. El valor de i en cada uno de esos términos es el valor que se asigna a cada $\psi_t(j)$. La probabilidad máxima para la secuencia de observaciones completa se alcanza en $\delta_8(2)$, lo cual implica que $q_8^* = 2$, y al reconstruir hacia atrás la secuencia de estados obtenemos

$$S = (2, 3, 1, 3, 2, 2, 3, 2)$$

tal y como se puede observar también en el enrejado de la figura 4.6. En dicha figura, se han representado todos los caminos posibles mediante líneas de puntos. Posteriormente, para todos los instantes de tiempo t, excepto el primero, cada estado j se une con una línea continua al estado del instante anterior que indique $\psi_t(j)$. Por ejemplo, el estado 1 del instante de tiempo 4 aparece unido con una línea continua con el estado 3 del instante 3, ya que $\psi_4(1) = 3$. La explicación intuitiva de esta línea es la siguiente: aún no sabemos si el camino más probable pasará por el estado 1 del instante 4, pero si lo hace, entonces sabemos que pasará también por el estado 3 del instante 3. Esta *traza* se mantiene hasta el final para todos los estados. Y por último, cuando vemos que la probabilidad máxima en el instante 8 correponde al estado 2, reconstruimos el camino hacia atrás desde ese punto para obtener la secuencia de estados más probable, que es la que aparece marcada con una línea continua más gruesa. □

Para el caso que nos ocupa, que es el de la etiquetación de palabras, los cálculos involucrados en el algoritmo de Viterbi se realizan frase por frase sobre enrejados simplificados como el de la figura 4.7, donde en cada posición no se consideran todos los estados posibles, es decir, todas la etiquetas del juego de etiquetas utilizado, sino sólo las etiquetas candidatas que proponga el diccionario para cada palabra.

No obstante, como se puede observar, hemos añadido un estado especial, que denominaremos estado 0 ó `etiqueta 0`, para marcar el comienzo y el fin de frase. Conceptualmente, el propósito real de este nuevo estado es garantizar que el etiquetador simula un sistema que funciona indefinidamente en el tiempo, sin detenerse, tal y como exige el paradigma de los modelos de Markov. En la práctica, una vez que nuestro etiquetador está funcionando, la justificación coloquial es que si tenemos que etiquetar un conjunto de frases, podemos procesar unas cuantas, detener el proceso, apagar el ordenador, encenderlo al día siguiente, arrancar de nuevo el proceso, etiquetar el resto de frases, y el resultado obtenido será el mismo que si las frases hubieran sido etiquetadas todas juntas en bloque.

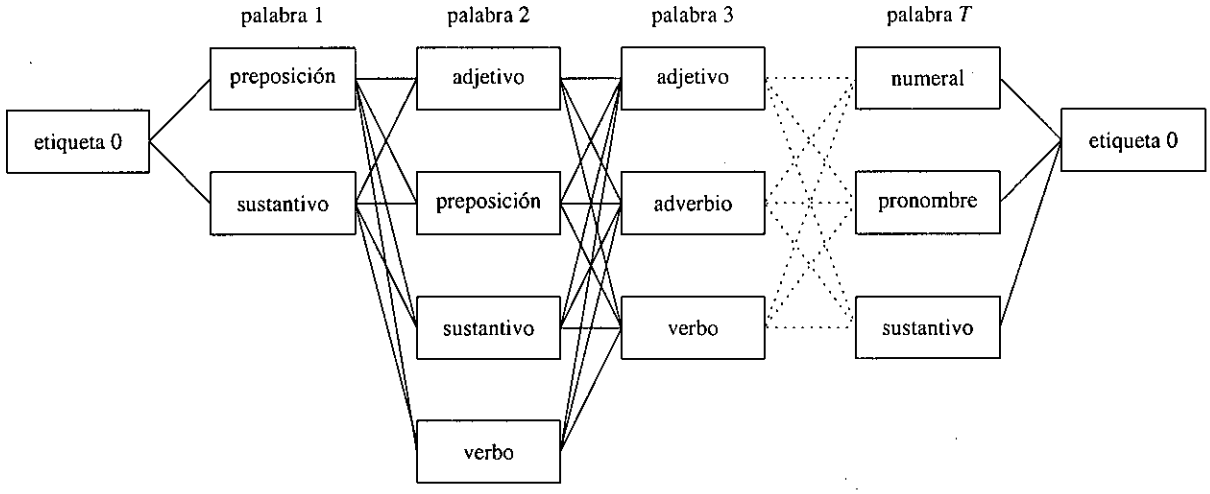


Figura 4.7: Enrejado simplificado para la etiquetación de una frase de T palabras

Los experimentos realizados por Merialdo sugieren que no existe una gran diferencia de éxito entre maximizar la probabilidad de cada etiqueta individualmente y maximizar la probabilidad de la secuencia completa, tal y como hace el algoritmo de Viterbi [Merialdo 1994]. Intuitivamente, esto es sencillo de comprender. Con el algoritmo de Viterbi, las transiciones entre estados o etiquetas son más sensibles, pero si algo va mal, se pueden obtener secuencias con varias etiquetas incorrectas. Maximizando etiqueta por etiqueta, esto no ocurre. Un fallo no desencadena otros fallos, de manera que lo que se obtiene son secuencias con errores ocasionales dispersos. No obstante, en la práctica, el método preferido para los etiquetadores basados en HMM,s sigue siendo el algoritmo de Viterbi.

Otra cuestión que podemos considerar es si el proceso de etiquetación resulta ser reversible o no, es decir, si etiquetando las frases de izquierda a derecha se obtienen los mismos resultados que etiquetando de derecha a izquierda. Si revisamos la implementación de los algoritmos de cálculo hacia adelante y hacia atrás de la probabilidad de una secuencia de observaciones dada (algoritmos 4.1 y 4.2), veremos que ambos obtienen la misma probabilidad. Así que, en definitiva, el proceso de etiquetación no es dependiente del sentido elegido. El algoritmo de Viterbi que hemos descrito aquí se basa en el algoritmo hacia adelante y por tanto etiqueta de izquierda a derecha, mientras que el etiquetador de Church, por ejemplo, lo hace en sentido contrario [Church 1988].

Sin embargo, lo que sí es importante es que hasta ahora hemos modelizado el proceso de la

etiquetación del lenguaje natural utilizando siempre HMM,s de orden 1, es decir, modelos en los que la dependencia contextual de una etiqueta se establece solamente en relación a la etiqueta anterior. A este tipo de modelos se les denomina también modelos de *bigramas* de etiquetas. Si se quiere trabajar con un HMM de orden superior, por ejemplo de orden 2, sería necesario extender los cálculos del algoritmo de Viterbi para considerar las transiciones de estados desde dos posiciones antes de la etiqueta actual, tal y como podemos ver a continuación.

Algoritmo 4.4 Cálculo de la secuencia de estados más probable para una secuencia de observaciones dada (algoritmo de Viterbi para HMM,s de orden 2):

1. Inicialización:

$$\delta_1(i) = \pi_i\, b_i(o_1), \qquad 1 \leq i \leq N.$$

$$\delta_2(i,j) = \delta_1(i)\, a_{ij}\, b_j(o_2), \qquad 1 \leq i,j \leq N. \tag{4.10}$$

2. Recurrencia:

$$\delta_{t+1}(j,k) = \left[\max_{1\leq i\leq N} \delta_t(i,j)\, a_{ijk}\right] b_k(o_{t+1}), \qquad t = 2,3,\ldots,T-1, \qquad 1 \leq j,k \leq N.$$

$$\psi_{t+1}(j,k) = \arg\max_{1\leq i\leq N} \delta_t(i,j)\, a_{ijk}, \qquad t = 2,3,\ldots,T-1, \qquad 1 \leq j,k \leq N.$$

3. Terminación:

$$(q^*_{T-1}, q^*_T) = \arg\max_{1\leq j,k\leq N} \delta_T(j,k).$$

4. Construcción hacia atrás de la secuencia de estados:

$$q^*_t = \psi_{t+2}(q^*_{t+1}, q^*_{t+2}), \qquad t = T-2, T-3, \ldots, 1.$$

Obsérvese que resulta necesario almacenar tanto $a_{ij} = P(j|i)$, la distribución de probabilidad de las transiciones entre cada posible par de estados, utilizada en la ecuación (4.10), segundo paso de inicialización, como $a_{ijk} = P(k|i,j)$, la distribución de probabilidad de las transiciones entre cada posible trío de estados, utilizada en el resto de pasos del algoritmo. □

Sin embargo, en lugar de realizar esta extensión, resulta mucho más sencillo cambiar el espacio de estados del modelo por la n-ésima potencia del conjunto de etiquetas, donde n es el orden del HMM. De esta manera, es posible realizar una única implementación del algoritmo de Viterbi que sirva para cualquier orden n, con sólo introducir dicho orden como un parámetro del algoritmo. Por ejemplo, en el caso de los HMM,s de orden 2, el conjunto de estados del modelo se define como el producto cartesiano del conjunto de etiquetas, siendo válidas las transiciones entre un estado (t^i, t^l) y otro (t^m, t^k), donde todas las t^j son etiquetas, sólo cuando $l = m$. En cualquier caso, esto es lo que se conoce como modelo de *trigramas* de etiquetas, ampliamente referenciado y utilizado por la mayoría de los etiquetadores estocásticos.

Por supuesto, cabría pensar que el hecho de aumentar el orden de un HMM podría desembocar en una mejora de la representación del contexto necesario para la correcta etiquetación de las palabras. Es decir, ¿por qué considerar sólo 2 etiquetas hacia atrás, en lugar de 3, ó 4, ó incluso más? Esto no se lleva a cabo en la práctica porque las unidades de información mayores que el trigrama generalmente modelizan fenómenos lingüísticos muy complejos y que afectan a muy pocos idiomas, resultando por ello suficientemente adecuado el uso de los trigramas. Por otra parte, el gran problema de aumentar el orden de un HMM es que crece desorbitadamente el número de parámetros que es necesario estimar para hacer operativo el modelo, tal y como veremos más adelante.

4.4.2.2 Implementaciones alternativas del algoritmo de Viterbi

Dado que el algoritmo de Viterbi no calcula una probabilidad exacta, sino que construye una secuencia de estados, podemos tomar logaritmos sobre los parámetros del modelo e implementarlo sólo con sumas, sin necesidad de ninguna multiplicación. De esta manera, no sólo se evita el problema de la rápida pérdida de precisión debida a las multiplicaciones de números muy pequeños, sino que además la velocidad de ejecución aumenta ya que las sumas se realizan más rápido que las multiplicaciones. En la práctica, es particularmente importante disponer de una implementación muy eficiente de este algoritmo, porque es el que realmente etiqueta las palabras que aparecen en los textos, mientras que el proceso de estimación de los parámetros del modelo, que veremos en la siguiente sección, puede realizarse de manera previa y separada, y por tanto su velocidad no es tan crítica. Los pasos del nuevo algoritmo son los siguientes.

Algoritmo 4.5 Cálculo de la secuencia de estados más probable para una secuencia de observaciones dada (algoritmo de Viterbi con logaritmos y sumas):

0. Preproceso:
$$\begin{aligned} \tilde{\pi}_i &= \log(\pi_i), && 1 \le i \le N. \\ \tilde{a}_{ij} &= \log(a_{ij}), && 1 \le i, j \le N. \\ \tilde{b}_i(o_t) &= \log[b_i(o_t)], && 1 \le i \le N,\ 1 \le t \le T. \end{aligned}$$

1. Inicialización:
$$\tilde{\delta}_1(i) = \log[\delta_1(i)] = \tilde{\pi}_i + \tilde{b}_i(o_1), \qquad 1 \le i \le N.$$

2. Recurrencia:
$$\tilde{\delta}_{t+1}(j) = \log[\delta_{t+1}(j)] = \left[\max_{1 \le i \le N}[\tilde{\delta}_t(i) + \tilde{a}_{ij}]\right] + \tilde{b}_j(o_{t+1}), \quad t = 1, 2, \ldots, T-1, \quad 1 \le j \le N.$$

$$\psi_{t+1}(j) = \arg\max_{1 \le i \le N}[\tilde{\delta}_t(i) + \tilde{a}_{ij}], \qquad t = 1, 2, \ldots, T-1, \qquad 1 \le j \le N.$$

3. Terminación:
$$q_T^* = \arg\max_{1 \le i \le N} \tilde{\delta}_T(i).$$

4. Construcción hacia atrás de la secuencia de estados:
$$q_t^* = \psi_{t+1}(q_{t+1}^*), \qquad t = T-1, T-2, \ldots, 1.$$

Esta implementación alternativa requiere del orden de N^2T sumas, más los cálculos necesarios para el paso de preproceso. No obstante, dado que el paso de preproceso normalmente se realiza una sola vez y se almacenan los resultados, su coste es despreciable en la mayoría de las aplicaciones. □

Por otra parte, el tiempo de procesamiento del algoritmo de Viterbi también se puede reducir introduciendo una búsqueda con corte[4]: cada estado que recibe un valor δ menor que el δ máximo hasta ese instante dividido por un cierto umbral θ queda excluido de los cálculos. Por supuesto, la incorporación de esta búsqueda con corte ya no garantiza que el algoritmo de Viterbi encuentre la secuencia de estados de máxima probabilidad. Sin embargo, para propósitos prácticos y con una buena elección del umbral θ, no existe virtualmente ninguna diferencia de precisión entre los algoritmos con y sin corte. Empíricamente, un valor de $\theta = 1.000$ puede aproximadamente doblar la velocidad de un etiquetador sin afectar a la precisión [Brants 2000].

[4] *Beam search* o *purging*.

4.4.3 Estimación de los parámetros del modelo

Dada una secuencia de observaciones O, el tercer problema fundamental relativo a los HMM,s consiste en determinar los parámetros del modelo que mejor *explican* los datos observados. Para el caso que nos ocupa, que es el de la etiquetación, dichas observaciones se corresponden con las palabras de un texto en lenguaje natural. En general, existen dos posibles métodos de estimación claramente diferenciados, en función de si el texto que observamos ha sido ya previamente etiquetado o no.

Con el fin de seguir el orden cronológico en el que han ido apareciendo las distintas técnicas, consideraremos primero el caso en el que el texto que observamos no está etiquetado. En este caso, es posible realizar un proceso de estimación, que denominaremos *no visible* o no supervisado, mediante el algoritmo de Baum-Welch. Sin embargo, veremos que un texto no etiquetado por sí solo no es suficiente para entrenar el modelo. Es necesario contar también con la ayuda de otro recurso lingüístico en forma de diccionario, que permita obtener un buen punto de inicialización para dicho algoritmo. Posteriormente, a medida que aparecieron disponibles los textos etiquetados, fue surgiendo también otro tipo de proceso de estimación que denominaremos *visible* o supervisado.

4.4.3.1 Estimación no supervisada: algoritmo de Baum-Welch

Cuando se dispone de una observación O, formada por un texto que no ha sido previamente etiquetado, este tercer problema de la estimación de los parámetros es el más complejo, ya que no se conoce ningún método analítico definitivo para encontrar un modelo $\mu = (\pi, A, B)$ que maximice $P(O|\mu)$. Sin embargo, podemos elegir un modelo que maximice localmente dicha probabilidad mediante un procedimiento iterativo tal como el algoritmo de Baum-Welch, que es un caso especial del algoritmo EM[5] [Dempster *et al.* 1977].

La idea intuitiva del algoritmo de Baum-Welch es la siguiente. En un primer momento, no sabemos cómo es el modelo, pero podemos trabajar sobre la probabilidad de la secuencia de observaciones utilizando algún modelo inicial, quizás preseleccionado o simplemente elegido de forma aleatoria. A partir de ese cálculo, identificamos qué transiciones y qué símbolos de emisión son los más probables. Incrementando la probabilidad de esas transiciones y de esos símbolos, construimos un modelo revisado, el cual obtendrá una probabilidad mayor que el modelo anterior para la secuencia de observaciones dada. Este proceso de maximización, denominado normalmente *proceso de entrenamiento*, se repite un cierto número de veces. Finalmente, el algoritmo se detiene cuando no consigue construir un modelo que mejore la probabilidad de la secuencia de observaciones dada.

Para describir formalmente este procedimiento, definimos primero $\xi_t(i,j)$, la probabilidad de estar en el estado i en el instante t y en el estado j en el instante $t+1$, dada la observación y dado el modelo, es decir,

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j | O, \mu). \tag{4.11}$$

Los caminos que satisfacen las condiciones requeridas por la ecuación (4.11) son los que se muestran en la figura 4.8. Por tanto, a partir de las definiciones de las variables hacia adelante y hacia atrás, podemos escribir $\xi_t(i,j)$ de la forma

$$\xi_t(i,j) = \frac{P(q_t = i, q_{t+1} = j, O|\mu)}{P(O|\mu)} = \frac{\alpha_t(i)\, a_{ij}\, b_j(o_{t+1})\, \beta_{t+1}(j)}{P(O|\mu)} = \frac{\alpha_t(i)\, a_{ij}\, b_j(o_{t+1})\, \beta_{t+1}(j)}{\sum_{k=1}^{N}\sum_{l=1}^{N} \alpha_t(k)\, a_{kl}\, b_l(o_{t+1})\, \beta_{t+1}(l)}.$$

[5] *Expectation-Maximization* (maximización de la esperanza); la parte E del algoritmo de Baum-Welch se conoce también como algoritmo *forward-backward* (hacia adelante y hacia atrás) [Baum 1972].

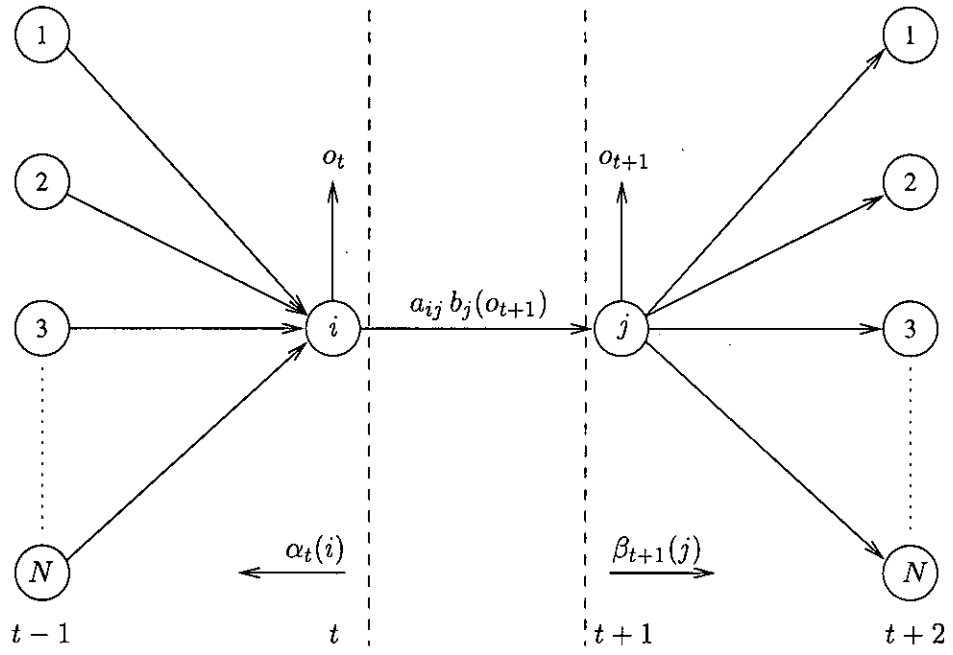


Figura 4.8: Detalle de la secuencia de operaciones necesarias para el cálculo de la probabilidad conjunta de que el sistema esté en el estado i en el instante t y en el estado j en el instante $t+1$

Previamente, en las ecuaciones (4.5) y (4.6), habíamos definido también $\gamma_t(i)$ como la probabilidad de estar en el estado i en el instante t, dada la secuencia de obsevaciones y dado el modelo. Por tanto, podemos relacionar $\gamma_t(i)$ con $\xi_t(i,j)$ mediante un sumatorio que recorre todo el espacio de estados sobre el índice j, dejando fijo el estado i, es decir,

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j).$$

Si sumamos $\gamma_t(i)$ a lo largo del tiempo, obtenemos un valor que se puede interpretar como el número esperado de veces que se visita el estado i, o lo que es lo mismo, el número esperado de transiciones hechas desde el estado i, si eliminamos del sumatorio el instante de tiempo $t = T$. De manera similar, el sumatorio de $\xi_t(i,j)$ sobre t, también desde $t = 1$ hasta $t = T-1$, se puede interpretar como el número esperado de transiciones desde el estado i al estado j. Es decir,

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{número esperado de transiciones desde el estado } i \text{ en } O,$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{número esperado de transiciones desde el estado } i \text{ al estado } j \text{ en } O.$$

Utilizando estas fórmulas, se puede dar un método general para reestimar los parámetros de un HMM.

Algoritmo 4.6 Reestimación de los parámetros de un HMM (algoritmo de Baum-Welch). El conjunto de ecuaciones para la reestimación de π, A y B es el siguiente:

$$\bar{\pi}_i = \text{frecuencia esperada de estar en el estado } i \text{ en el primer instante } = \gamma_1(i),$$

$$\bar{a}_{ij} = \frac{\text{número esperado de transiciones desde el estado } i \text{ al estado } j}{\text{número esperado de transiciones desde el estado } i} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)},$$

$$\bar{b}_j(v_k) = \frac{\text{número esperado de veces en el estado } j \text{ observando el símbolo } v_k}{\text{número esperado de veces en el estado } j} = \frac{\sum_{t=1}^{T} \gamma_t(j) \text{ tal que } o_t = v_k}{\sum_{t=1}^{T} \gamma_t(j)}.$$

Si definimos el modelo actual como $\mu = (\pi, A, B)$ y lo utilizamos para calcular los términos que aparecen en las partes derechas de las tres ecuaciones anteriores, y definimos el modelo reestimado como $\bar{\mu} = (\bar{\pi}, \bar{A}, \bar{B})$ a partir de los valores obtenidos para las partes izquierdas de dichas ecuaciones, está demostrado por Baum que: o bien (1) el modelo inicial μ define un punto crítico para la función de probabilidad, en cuyo caso $\bar{\mu} = \mu$; o bien (2) el modelo $\bar{\mu}$ es más probable que el modelo μ en el sentido de que $P(O|\bar{\mu}) > P(O|\mu)$, esto es, hemos encontrado un nuevo modelo $\bar{\mu}$ para el cual la probabilidad de generar la secuencia de observaciones es mayor. Así pues, reemplazando μ por $\bar{\mu}$ y repitiendo la reestimación de los parámetros un cierto número de veces, podemos mejorar la probabilidad de O en cada iteración, hasta que no se aprecie ninguna ganancia significativa. □

Sin embargo, este método para la reestimación de los parámetros es muy sensible a las condiciones de inicialización del modelo. Es decir, el método no garantiza que encontremos el mejor modelo, ya que, debido a una inicialización incorrecta, el proceso podría detenerse en un máximo local, y en la mayoría de los problemas reales la función de probabilidad suele ser muy compleja, no lineal y con numerosos máximos locales. Por tanto, para encontrar el máximo global, una posible aproximación es intentar inicializar el modelo en una región del espacio de parámetros cercana a ese máximo global. En definitiva, una inicialización aleatoria podría dejar el problema de la reestimación de parámetros demasiado abierto y la manera de restringir el problema es elegir unos parámetros adecuados, en lugar de fijarlos aleatoriamente.

Algunos autores consideran que en la práctica, una estimación inicial de los parámetros π y A con números aproximadamente iguales, ligeramente perturbados para evitar los máximos locales, pero que respeten las restricciones estocásticas estándar y que sean siempre distintos de cero, suele ser normalmente satisfactoria para que posteriormente el algoritmo de Baum-Welch aprenda por sí solo la regularidad de las etiquetas, es decir, todo lo relativo a los transiciones entre los estados del modelo. Sin embargo esto no es más que una confusión que parte de la creencia general de que cuando un sistema tiene la capacidad de aprender, debe aprenderlo todo por sí mismo. Esto es efectivamente falso. Cuanta más información se le proporcione inicialmente al sistema, mejor[6]. Es decir, en nuestro caso, se ha demostrado que siempre se van a obtener mejores resultados cuando se inicializan los parámetros π y A a partir de un

[6]Por supuesto, esta idea de considerar como modelo más preciso el que más se acerca a los datos disponibles, llevada al extremo, degenera en un tratamiento manual específico para cualquier nuevo corpus, lo cual no es en absoluto realista. Ésta es la razón por la cual se desarrollan modelos abstractos y técnicas de aprendizaje: para que los sistemas se adapten rápidamente a los nuevos datos y sean capaces de manejarlos adecuadamente. Por tanto, se debe encontrar un equilibrio entre: por un lado, la descripción del modelo (y la dificultad de parametrizarlo), y por otro, su posibilidad de adaptación (es decir, su aprendizaje) con respecto al volumen de datos. Desde el punto de vista de la estadística y del aprendizaje, esto se conoce como el *Bias-Variance balance* (balance del sesgo y la varianza).

texto etiquetado, por pequeño que sea. Dicho texto podría ser construido manualmente si no se dispone de ninguno. Incluso repetir unas cuantas veces la estrategia de construir un modelo inicial, etiquetar otra pequeña porción de texto, corregirla manualmente, y utilizarla para estimar de nuevo los parámetros del modelo, ofrecería mejores resultados que la aplicación a ciegas del método de Baum-Welch. En definitiva, lo que queremos señalar es que la división entre métodos de estimación visible y no visible no es en nuestro caso tan estricta como la estamos presentando aquí. O si se prefiere, podríamos decir que, en sentido estricto, la estimación totalmente no visible no existe.

En cualquier caso, cuando no se dispone de textos etiquetados, o cuando estos son muy pequeños, resulta importante obtener un buen valor inicial para el parámetro B, es decir, para las probabilidades de emisión de las palabras, lo cual puede hacerse mediante el uso de un diccionario. Existen dos métodos generales para realizar esta inicialización: el método de Jelinek y el método de Kupiec. A continuación esbozamos cada uno de ellos:

- **Método de Jelinek**. Si definimos $Q(v_k)$ como el número de etiquetas permitidas para la palabra v_k en el diccionario, $C(v_k)$ como el número de apariciones de la palabra v_k en el corpus de entrenamiento, y $b_j^*(v_k)$ como

$$b_j^*(v_k) = \begin{cases} 0 & \text{si } j \text{ no es una etiqueta permitida para la palabra } v_k \\ \frac{1}{Q(v_k)} & \text{en caso contrario,} \end{cases}$$

  entonces se inicializa $b_j(v_k)$ con el valor

$$b_j(v_k) = \frac{b_j^*(v_k)\, C(v_k)}{\sum_m b_j^*(v_m)\, C(v_m)}.$$

  Es decir, el método de Jelinek inicializa las probabilidades de emisión del modelo mediante el uso de la regla de Bayes, estimando mediante la frecuencia observada la probabilidad de aparición de una palabra y suponiendo que todas las etiquetas que aparecen en el diccionario para una palabra dada son igualmente probables [Jelinek 1985].

- **Método de Kupiec**. La otra alternativa es agrupar las palabras en clases de ambigüedad, de tal manera que todas aquellas palabras que tengan el mismo conjunto de etiquetas permitidas en el diccionario pertenezcan a la misma clase o *metapalabra* u_L, donde L es un subconjunto de Q, el conjunto de etiquetas utilizado. Es decir, si $L(v_k)$ es el conjunto de todas las etiquetas posibles para la palabra v_k, entonces

$$u_L = \{v_k \mid L = L(v_k)\}, \qquad \forall L \subseteq Q.$$

  Por ejemplo, la metapalabra $u_{\{\texttt{Scms,P}\}}$ contendrá todas las palabras para las cuales el diccionario permite las etiquetas `Scms`, es decir, sustantivo común masculino singular, y `P`, es decir, preposición, y ninguna otra etiqueta más. Entonces, a cada una de estas metapalabras se le da un tratamiento similar al del método de Jelinek:

$$b_j^*(u_L) = \begin{cases} 0 & \text{si } j \notin L \\ \frac{1}{|L|} & \text{en caso contrario,} \end{cases}$$

  donde $|L|$ es el cardinal del conjunto L, y

$$b_j(u_L) = \frac{b_j^*(u_L)\, C(u_L)}{\sum_{L'} b_j^*(u_{L'})\, C(u_{L'})}$$

donde $C(u_L)$ (respectivamente $C(u_{L'})$) es el número de palabras pertenecientes a u_L (respectivamente $u_{L'}$) que aparecen en el corpus de entrenamiento. La implementación real utilizada por Kupiec es una variante de la presentada aquí, pero se ha expuesto de esta manera para hacer más transparente la similitud entre ambos métodos. Por ejemplo, Kupiec no incluye las 100 palabras más frecuentes dentro de las clases de ambigüedad, sino que las trata separadamente como clases de una sola palabra, con el fin de no introducir errores y mejorar así la estimación inicial [Kupiec 1992].

La ventaja del método de Kupiec sobre el de Jelinek es que no necesita afinar una probabilidad de emisión diferente para cada palabra. Mediante la definición de estas clases de ambigüedad, el número total de parámetros se reduce substancialmente y éstos se pueden estimar de una manera más precisa. Por supuesto, esta ventaja podría convertirse en desventaja si existe la suficiente cantidad de material de entrenamiento como para estimar los parámetros palabra por palabra tal y como hace el método de Jelinek.

En el algoritmo de Baum-Welch también es necesario hacer algo para evitar el problema de la pérdida de precisión en punto flotante. Sin embargo, la necesidad de realizar sumas dificulta el uso de logaritmos. Una solución bastante común es utilizar unos coeficientes auxiliares de escalado, cuyos valores crecen con el tiempo de manera que si se multiplican las probabilidades por esos coeficientes, éstas siempre se mantienen dentro del rango de punto flotante del ordenador. Al final de cada iteración, cuando efectivamente se reestiman los parámetros, se cancela el uso de esos coeficientes [Rabiner y Juang 1993, pp. 365-368]. Otra alternativa es utilizar igualmente logaritmos y utilizar la función que mostramos a continuación para sumarlos.

Algoritmo 4.7 Función para la suma de logaritmos. Dados $x = \log(x')$ e $y = \log(y')$, la función calcula $\log(x' + y')$:

```
function Sumar_Logaritmos (x, y) =
  begin
    if (y − x > log C) then
      return y
    else
      if (x − y > log C) then
        return x
      else
        return min (x, y) + log (exp (x − min (x, y)) + exp (y − min (x, y)))
  end;
```

En esta porción de pseudo-código, C representa una constante grande, del orden de 10^{30}. De esta manera, lo que estamos haciendo es calcular un factor de escalado apropiado en el momento de realizar cada suma, y lo único que resta es tener cuidado con los errores de redondeo. □

4.4.3.2 Estimación supervisada: métodos de suavización

Cuando se dispone de un texto etiquetado, la primera idea que acude a nuestra mente para abordar el proceso de estimación de los parámetros del modelo es quizás la de diseñar un sencillo mecanismo basado en el uso de frecuencias relativas. Pero antes de comentar dicho mecanismo, vamos a introducir un cambio en la notación que hace referencia a la definición formal de los HMM,s.

Hasta aquí, la notación que hemos utilizado estaba inspirada en la excelente presentación que Rabiner llevó a cabo en relación con este tipo de procesos estocásticos [Rabiner 1989].

Efectivamente, se trata de una notación genérica sobre HMM,s especialmente orientada a describir con el máximo detalle la implementación de determinados algoritmos, como pueden ser el de Viterbi o el de Baum-Welch. Sin embargo, una vez que hemos centrado la discusión sobre el tema concreto de la etiquetación, algunos aspectos de esa notación no se identifican con nuestro problema todo lo bien que nos gustaría. Por poner un ejemplo, los estados de un HMM habían sido denotados simplemente con los números enteros del conjunto $\{1, 2, \ldots, N\}$. Esto facilita mucho la expresión de operaciones tales como la indexación de las celdas de la matriz A de transiciones entre estados, pero no nos permite hacer referencia a las etiquetas concretas que se están utilizando, si no es a través de la consideración de una función correspondencia entre ambos conjuntos.

La notación que proponemos ahora está más acorde con la que se adopta en la mayoría de las referencias bibliográficas dedicadas específicamente al tema de la etiquetación.

Definición 4.2 Básicamente, la nueva notación describe cada uno de los elementos de un HMM como sigue:

1. Considerando la inicial de la palabra inglesa *tag* (etiqueta), denotaremos mediante t^i la i-ésima etiqueta del conjunto de etiquetas utilizado, y mediante t_i la etiqueta de la palabra que ocupa la posición i dentro de la frase que se está tratando. De esta manera, en el modelo de bigramas, los estados se denotan también mediante t^i, es decir, mediante el nombre de las propias etiquetas. Y si el modelo está basado en trigramas, los estados serán de la forma (t^i, t^j), donde cada uno de estos pares proviene del producto cartesiano del conjunto de etiquetas utilizado, tal y como ya habíamos esbozado.

2. Anteriormente, habíamos denotado mediante v_k cada uno de los símbolos de observación correspondientes a la parte *visible* del modelo, y habíamos visto que, en el caso de la etiquetación, dichos sucesos observables se corresponden con las palabras. Pues bien, de igual manera, considerando la inicial del vocablo inglés *word* (palabra), denotaremos ahora mediante w^i la i-ésima palabra del diccionario, y mediante w_i la palabra que ocupa la posición i dentro de la frase.

3. Habíamos indicado también que trabajaremos siempre en el nivel de frase, y que consideraremos un estado o etiqueta especial t^0 para marcar el inicio y el final de cada una de las frases. Por tanto, respecto al parámetro π, la distribución de probabilidad del estado inicial, tendremos que $\pi(t^0)$ será igual a 1, y $\pi(t^i)$ será igual a 0, para todo $i = 1, 2, \ldots, N$.

4. Por lo que respecta al parámetro A, la distribución de probabilidad de las transiciones entre estados, basta aclarar que, de acuerdo con la nueva notación, la información que realmente está almacenada en cada una de las celdas de esta matriz es $P(t^j|t^i)$ en el caso de los modelos basados en bigramas, y $P(t^k|t^i\, t^j)$ en el caso de los modelos basados en trigramas.

5. Por último, respecto a B, las probabilidades de emisión de las palabras, simplemente indicamos que la información que antes se denotaba mediante $b_j(v_k)$ se escribe ahora como $P(w^k|t^j)$.

Para terminar, completamos esta notación con algunos aspectos que nos permitirán avanzar más cómodamente con la exposición. Denotaremos mediante $C(x)$ el número de veces que el suceso x aparece en el corpus de entrenamiento. Por ejemplo, $C(t^i, t^j, t^k)$ representa el número de veces que aparece el trigrama $t^i\, t^j\, t^k$. De igual manera, $C(w^k|t^j)$ representa el número de veces que aparece la palabra w^k etiquetada como t^j. Denotaremos también mediante $f(x)$ la frecuencia del suceso x en el corpus de entrenamiento. Y finalmente denotaremos mediante $\hat{p}(x)$ nuestra estimación de la probabilidad real $P(x)$. □

Nuevo modelo probabilístico. La etiquetación se describe entonces, a través de esta nueva notación, como el proceso de encontrar la mejor secuencia de etiquetas $t_{1,n}$ para una frase dada de n palabras $w_{1,n}$. Aplicando la regla de Bayes, podemos escribir:

$$\arg\max_{t_{1,n}} P(t_{1,n}|w_{1,n}) = \arg\max_{t_{1,n}} \frac{P(w_{1,n}|t_{1,n})P(t_{1,n})}{P(w_{1,n})} = \arg\max_{t_{1,n}} P(w_{1,n}|t_{1,n})P(t_{1,n}).$$

Ahora, trabajamos sobre esta expresión para conseguir que utilice los parámetros que podemos estimar directamente desde el corpus de entrenamiento. Para ello, además de la propiedad del horizonte limitado (4.14), utilizamos dos suposiciones más acerca de las palabras:

1. Las etiquetas no son independientes unas de otras, pero las palabras sí (4.12).
2. La identidad de una palabra depende sólo de su propia etiqueta (4.13).

Así pues, tenemos que:

$$P(w_{1,n}|t_{1,n})P(t_{1,n}) =$$

$$= \left(\prod_{i=1}^{n} P(w_i|t_{1,n})\right) \times P(t_n|t_{1,n-1}) \times P(t_{n-1}|t_{1,n-2}) \times \ldots \times P(t_2|t_1) = \quad (4.12)$$

$$= \left(\prod_{i=1}^{n} P(w_i|t_i)\right) \times P(t_n|t_{1,n-1}) \times P(t_{n-1}|t_{1,n-2}) \times \ldots \times P(t_2|t_1) = \quad (4.13)$$

$$= \left(\prod_{i=1}^{n} P(w_i|t_i)\right) \times P(t_n|t_{n-1}) \times P(t_{n-1}|t_{n-2}) \times \ldots \times P(t_2|t_1) = \quad (4.14)$$

$$= \prod_{i=1}^{n} [P(w_i|t_i) \times P(t_i|t_{i-1})].$$

Por tanto, la ecuación final para determinar la secuencia de etiquetas óptima para una frase dada es

$$\hat{t}_{1,n} = \arg\max_{t_{1,n}} P(t_{1,n}|w_{1,n}) = \arg\max_{t_{1,n}} \prod_{i=1}^{n} [P(w_i|t_i) \times P(t_i|t_{i-1})]$$

para los etiquetadores basados en bigramas, y

$$\hat{t}_{1,n} = \arg\max_{t_{1,n}} P(t_{1,n}|w_{1,n}) = \arg\max_{t_{1,n}} \prod_{i=1}^{n} [P(w_i|t_i) \times P(t_i|t_{i-2}, t_{i-1})]$$

para los etiquetadores basados en trigramas. Estos cálculos son precisamente los que realiza el algoritmo de Viterbi, de tal manera que para disponer de un etiquetador totalmente operativo sólo resta especificar de qué manera se pueden obtener los parámetros de funcionamiento del HMM subyacente. La sección anterior presentaba un método para la estimación de dichos parámetros a partir de textos no etiquetados. Ahora nos ocuparemos del diseño de métodos de estimación de parámetros a partir de textos etiquetados.

Retomemos entonces la idea inicial de diseñar un mecanismo intuitivo para estimar los parámetros de nuestro modelo. Dado que en este caso partimos de un texto previamente etiquetado, podemos sugerir que el proceso de estimación de las transiciones entre estados podría realizarse en base a unos sencillos cálculos de frecuencias relativas. En el caso de un modelo basado en bigramas, tendríamos que

$$\hat{p}(t^j|t^i) = f(t^j|t^i) = \frac{C(t^i, t^j)}{C(t^i)}.$$

En el caso de un modelo basado en trigramas, tendríamos que

$$\hat{p}(t^k|t^i\,t^j) = f(t^k|t^i\,t^j) = \frac{C(t^i,t^j,t^k)}{C(t^i,t^j)}.$$

De igual manera, para la estimación de las probabilidades de emisión de las palabras, tendríamos que

$$\hat{p}(w^k|t^j) = f(w^k|t^j) = \frac{C(w^k|t^j)}{C(t^j)}.$$

Hasta aquí, lo que hacemos no es realmente estimar, sino construir un modelo de manera visible. Una vez que este modelo está preparado, podemos utilizarlo para etiquetar nuevos textos mediante la aplicación del algoritmo de Viterbi a cada una de sus frases, y la idea original de modelo oculto vuelve a tener sentido.

Sin embargo, antes de utilizar directamente un modelo construido a partir de frecuencias relativas, es necesario hacer la siguiente reflexión. Pensemos en un caso práctico real, como por ejemplo el experimento realizado con el corpus ITU[7]. El cardinal del juego de etiquetas utilizado en dicho experimento es de 373 etiquetas, tal y como puede verse en la sección A.2. Esto no quiere decir que una frase de longitud n palabras deba ser etiquetada aplicando el algoritmo de Viterbi sobre un enrejado de $373 \times n$ estados, sino más bien sobre un enrejado simplificado como el que veíamos en la figura 4.7, donde se consideran sólo las posibles etiquetas de cada palabra. Lo que sí quiere decir es que, en el caso de un modelo basado en bigramas, la matriz A contendría $373 \times 373 = 139.129$ celdas, y sin embargo, en el mayor de los corpus de entrenamiento utilizados en dicho experimento, tan solo se pueden ver 4.037 de esas transiciones (es decir, el $2,90\%$), y las restantes celdas quedarían a cero. En el caso de un modelo basado en trigramas, la matriz A contendría $373 \times 373 \times 373 = 51.895.117$ celdas, sólo se pueden llegar a ver 23.119 (el $0,04\%$), y las restantes quedarían también a cero. Esto es debido al fenómeno de *dispersión de los datos*[8], el cual puede provocar también que algunas de esas transiciones aparezcan como candidatas a la hora de etiquetar nuevas frases.

Así pues, teniendo en cuenta que los enrejados simplificados ya contemplan sólo un subconjunto reducido de todos los caminos posibles, parece muy arriesgado trabajar con transiciones nulas, cuyo uso implica multiplicaciones por cero que anularían completamente todos los caminos que pasen por ellas. Para evitar el problema de las transiciones nulas, se suelen utilizar *métodos de suavización*[9]. Estos métodos permiten que a partir de muestras pequeñas se puedan estimar unas probabilidades más representativas del comportamiento real de la población que estamos estudiando.

Una forma de implementar la suavización es mediante técnicas de *interpolación lineal*. En general, el esquema de suavizado mediante interpolación lineal funciona como sigue. La distribución $f(x)$ observada en un conjunto de E posibles sucesos se modifica añadiendo un valor muy pequeño procedente de una distribución menos específica $q(x)$ a la cual damos un peso o confianza α. Entonces, la distribución de probabilidad que nos interesa se aproxima de la siguiente forma:

$$\hat{p}(x) \approx \frac{f(x) + \alpha\, q(x)}{E + \alpha}.$$

Y si utilizamos el parámetro tradicional de interpolación $\lambda = \frac{\alpha}{E+\alpha}$, entonces también se verifica la siguiente aproximación:

$$\hat{p}(x) \approx (1-\lambda)\,\frac{f(x)}{E} + \lambda\, q(x).$$

[7] *International Telecommunications Union CCITT Handbook* (véase la sección 2.1).

[8] También denominado fenómeno de *sparse data*.

[9] También denominados métodos de *smoothing*.

Es decir, si lo que queremos estimar son las probabilidades de transición entre estados de un modelo basado en bigramas, esa distribución menos específica ponderada con α al interpolar puede ser la distribución de probabilidad de los *unigramas*, esto es, la distribución de probabilidad de cada etiqueta individualmente[10]. Por tanto, las probabilidades de los bigramas se pueden aproximar mediante

$$\hat{p}(t_i|t_{i-1}) = (1-\lambda)\, f(t_i|t_{i-1}) + \lambda\, f(t_i)$$

y las de los trigramas mediante

$$\hat{p}(t_i|t_{i-2}\, t_{i-1}) = \lambda_3\, f(t_i|t_{i-2}\, t_{i-1}) + \lambda_2\, f(t_i|t_{i-1}) + \lambda_1\, f(t_i) \tag{4.15}$$

donde todos los *pesos* λ_i deben ser no negativos y deben satisfacer la restricción $\lambda_1 + \lambda_2 + \lambda_3 = 1$. A continuación, y centrando ya toda nuestra atención en el modelo de trigramas, discutiremos cómo elegir los pesos λ_i de manera óptima.

Suavizado lineal óptimo. Un modelo de lenguaje que opere de acuerdo con la ecuación (4.15) se puede ver realmente como un HMM. La figura 4.9 muestra un fragmento de este modelo. En este punto es importante aclarar la siguiente cuestión. Existe una variante de representación de los HMM,s que lleva asociadas las probabilidades de emisión no a un único estado, sino tanto al estado origen como al estado destino de las transiciones. Es decir, dichas probabilidades de emisión están asociadas realmente a los arcos o transiciones, no a los estados individuales. Algunos autores prefieren enfocar la introducción a los HMM,s comenzando con este tipo de representación, ya que el paso de generalización desde los HMM,s con probabilidades de emisión en los arcos a los HMM,s con probabilidades de emisión en los estados resulta más natural que el paso inverso. A pesar de esto, nosotros hemos preferido manejar desde el principio HMM,s con probabilidades de emisión en los estados, porque este es el tipo de HMM,s que efectivamente se utiliza en la práctica para la etiquetación de textos, y por tanto es posible establecer analogías entre ambos conceptos desde el primer momento.

No obstante, el HMM de la figura 4.9 es un HMM con probabilidades de emisión en los arcos. Ocurre además que en este tipo de HMM,s está permitido el uso de transiciones que no emiten ningún símbolo, sino que simplemente se utilizan para cambiar de estado. Este tipo de transiciones se denominan *transiciones épsilon* o *transiciones vacías*. En la figura, no aparecen representados los símbolos de salida, sino simplemente las probabilidades de cada arco. Por tanto, para distinguir las transiciones normales de las transiciones épsilon, hemos representado estas últimas mediante líneas discontinuas. Así pues, saliendo del estado más a la izquierda (t^i, t^j) tenemos tres transiciones vacías que van a los pseudo-estados $s_1(t^i, t^j)$, $s_2(t^i, t^j)$ y $s_3(t^i, t^j)$. Las probabilidades de estas transiciones son λ_1, λ_2 y λ_3, respectivamente. A continuación, saliendo de cada uno de los tres pseudo-estados aparecen N transiciones. Cada una de ellas conduce a un estado (t^j, t^k), $k = 1, 2, \ldots, N$, para generar la tercera etiqueta del trigrama. Las probabilidades de estas transiciones son $f(t^k)$, $f(t^k|t^j)$ y $f(t^k|t^i\, t^j)$, $k = 1, 2, \ldots, N$, respectivamente. En la figura, hemos etiquetado sólo las transiciones que entran en los estados (t^j, t^1) y (t^j, t^N). Nótese que desde el estado más a la izquierda (t^i, t^j) se puede alcanzar cada uno de estos estados a través de tres caminos posibles, y que por tanto la probabilidad total de llegar a cada estado coincide con el cálculo de la ecuación (4.15).

[10] Esta fórmula viene del hecho de que $P(X|Y) = P(X)$, si X e Y son independientes. Por eso decimos que $P(X|Y)$ se estima mediante $f(X|Y)$, o en todo caso mediante una combinación lineal de $f(X|Y)$ y $f(X)$, si el bigrama $Y\,X$ aparece en el corpus de entrenamiento, y mediante $f(X)$ si no aparece, asumiendo la hipótesis de independencia para esos sucesos perdidos. Rigurosamente hablando esto no es correcto, pero constituye una buena aproximación de primer orden (considerando la hipótesis de Markov como una hipótesis de segundo orden).

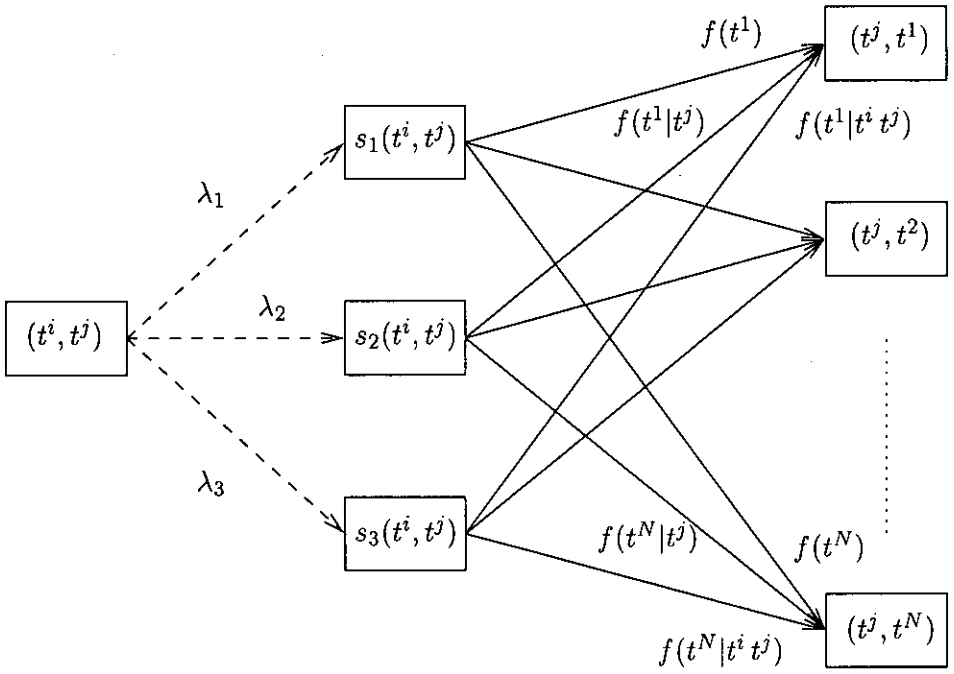


Figura 4.9: Fragmento del suavizado lineal de un HMM basado en trigramas

Por supuesto, otra forma de ver el problema podría ser considerando el HMM de la figura 4.9 todavía como un HMM con probabilidades de emisión en los estados, salvo que ahora es a los pseudo-estados $s_1(t^i, t^j)$, $s_2(t^i, t^j)$ y $s_3(t^i, t^j)$ a los que no se les permite generar ninguna palabra. En cualquier caso, tenemos que las probabilidades de las transiciones normales son conocidas, mientras que las de las transiciones épsilon, λ_1, λ_2 y λ_3, deben ser determinadas mediante algún procedimiento. El HMM completo es desde luego enorme: consta de $4 \times N^2$ estados, donde N es el tamaño del conjunto de etiquetas utilizado. Pero de acuerdo con la ecuación (4.15), para un i dado, $i \in \{1, 2, 3\}$, todas las probabilidades λ_i de las transiciones que van desde el estado más a la izquierda (t^i, t^j) hasta uno de los estados más a la derecha (t^j, t^k) tienen el mismo valor, sea cual sea la combinación de etiquetas que forma el trigrama $t^i\,t^j\,t^k$. Se dice entonces que estas probabilidades toman *valores empatados*[11].

Pues bien, dado que hemos visto que el modelo de lenguaje al que obedece la ecuación (4.15) es un HMM, se puede utilizar el algoritmo de Baum-Welch para estimar los valores λ_i óptimos. Además, como una consecuencia favorable de que el algoritmo deba *empatar* o igualar las probabilidades λ_i, independientemente de a qué subparte del HMM pertenezcan, el proceso de estimación sólo necesita manejar tres contadores, uno para cada tipo de transición épsilon, del número esperado de veces que se cruza por una transición. Para el resto de transiciones, no se necesita establecer ningún otro tipo de contador porque de hecho las probabilidades de esas transiciones están ya fijadas de antemano. Debido a todo esto, se pueden sugerir ideas prácticas que dan lugar a cálculos en este caso más sencillos que los del algoritmo de Baum-Welch, y que por tanto simplifican aún más la obtención de los λ_i óptimos [Jelinek 1997, pp. 66-69].

Pero ahora preferimos centrar nuestro interés en responder a una importante pregunta: ¿qué tipo de datos de entrenamiento deberíamos usar para determinar los pesos λ_i? Está claro que no pueden ser los mismos datos sobre los que se calculan las frecuencias $f(\ |\ )$, porque en ese caso las estimaciones darían como resultado $\lambda_3 = 1$ y $\lambda_1 = \lambda_2 = 0$. De hecho, $f(t^k|t^i\,t^j)$ es la estimación

[11] O también *tied values*.

de máxima verosimilitud de $P(t^k|t^i\,t^j)$ para los datos de entrenamiento en los que se basa dicha frecuencia $f(t^k|t^i\,t^j)$. Sin embargo, mirando de nuevo la figura 4.9, se intuye que, de los tres caminos posibles que conducen a cada uno de los estados más a la derecha, el HMM ha incluido el camino superior para poder generar la tercera etiqueta t^k, incluso aunque el bigrama $t^j\,t^k$ no haya aparecido en los datos de entrenamiento. Y de manera similar, mediante ese mismo camino superior y mediante el camino central, se hace posible la tercera etiqueta t^k, incluso aunque no exista ninguna ocurrencia del trigrama $t^i\,t^j\,t^k$. Por tanto se concluye que el conjunto total de los datos de entrenamiento debe ser dividido en dos porciones:

1. La primera, mucho más grande, denominada *datos de desarrollo*[12], se utiliza para estimar las frecuencias relativas $f(\ |\ )$.

2. Una vez que éstas están fijadas, la segunda porción de datos, mucho más pequeña ya que quedan por estimar muchos menos parámetros, se usa para estimar los pesos λ_i. Esta porción se denomina *datos extendidos* u *ocultos*[13].

Por supuesto, una vez que se ha realizado este proceso, el modelo obtenido se puede mejorar combinando ambas porciones de datos y reestimando las frecuencias $f(\ |\ )$. Esta técnica de suavizado lineal se denomina *interpolación de borrado*[14].

La interpolación de borrado se puede realizar también mediante una eliminación sucesiva de cada trigrama del corpus de entrenamiento, y una posterior estimación de los λ_i óptimos a partir del resto de n-gramas del corpus. Conocidos los contadores de frecuencias para unigramas, bigramas y trigramas, los pesos λ_i se pueden determinar eficientemente a través del siguiente algoritmo.

Algoritmo 4.8 Cálculo de los parámetros λ_1, λ_2 y λ_3 de un esquema de interpolación lineal, conocidas las frecuencias de unigramas, bigramas y trigramas, y conocido N, el tamaño del corpus de entrenamiento [Brants 2000]:

1. Inicializar $\lambda_1 = \lambda_2 = \lambda_3 = 0$.

2. Para cada trigrama $t_1\,t_2\,t_3$ con $C(t_1,t_2,t_3) > 0$, localizar el máximo de los tres valores siguientes y realizar la acción correspondiente:

   - $\frac{C(t_1,t_2,t_3)-1}{C(t_1,t_2)-1}$: incrementar λ_3 en $C(t_1,t_2,t_3)$ unidades.
   - $\frac{C(t_2,t_3)-1}{C(t_2)-1}$: incrementar λ_2 en $C(t_1,t_2,t_3)$ unidades.
   - $\frac{C(t_3)-1}{N-1}$: incrementar λ_1 en $C(t_1,t_2,t_3)$ unidades.

3. Normalizar los valores λ_1, λ_2 y λ_3.

Abusando un poco de la notación, con el fin de no complicar en exceso las ecuaciones con la presencia de tantos subíndices, hemos denotado el trigrama $t_{i-2}\,t_{i-1}\,t_i$ mediante $t_1\,t_2\,t_3$, donde de manera obvia los números 1, 2 y 3 hacen referencia a la posición de cada etiqueta dentro del trigrama. Si el denominador de alguna de las expresiones es 0, se define el resultado de esa expresión como 0. Restar 1 en este algoritmo es la manera de tener en cuenta los datos no observados. Sin esta resta, el modelo efectivamente sobreestimaría los datos de entrenamiento, generando $\lambda_3 = 1$ y $\lambda_1 = \lambda_2 = 0$ y produciendo peores resultados. □

[12] *Development data.*
[13] *Held-out data.*
[14] O también *deleted interpolation.*

Hemos visto que la ecuación (4.15) propone valores constantes para los parámetros de interpolación λ_1, λ_2 y λ_3. Es cierto que quizás no es una buena idea utilizar siempre los mismos valores, pero es cierto también que la consideración de un conjunto de valores λ_i, $i \in \{1,2,3\}$, para cada posible par de etiquetas eleva muchísimo el número de parámetros del modelo, lo cual no sólo no introduce ninguna mejora en relación con el fenómeno de los datos dispersos, sino que empeora el problema.

No obstante, algunos autores han sugerido que al menos sí sería conveniente una agrupación de los bigramas en un número moderado de clases y una posterior estimación de un conjunto de valores λ_i distinto para cada clase, aunque no queda claro cuál sería un buen criterio para la definición de esas clases.

Brants es quizás el único autor que proporciona datos concretos sobre diferentes experimentos de agrupación. Uno de ellos incluía un conjunto de valores λ_i distinto para cada frecuencia. Otra de las posibilidades estudiadas fue la definición de dos clases, frecuencias altas y frecuencias bajas, a ambos extremos de la escala, y la posterior aplicación de diferentes criterios de agrupación, también basados en frecuencias, para definir las clases intermedias. Pero finalmente observó que la mayoría de los resultados eran equivalentes, obteniendo incluso peores cifras con algunas de las agrupaciones consideradas. Por esta razón, Brants utiliza en su etiquetador interpolación lineal independiente del contexto, es decir, valores constantes para los parámetros λ_i [Brants 2000].

Otros autores, sin embargo, proponen una reducción del número de parámetros a estimar mediante la especificación de que ciertos sucesos son absolutamente improbables, es decir, tienen probabilidad 0, lo cual permite introducir *ceros estructurales* en el modelo. Efectivamente, el hecho de hacer que algunos sucesos no sean posibles, por ejemplo un trigrama formado por tres preposiciones seguidas, añade gran cantidad de estructura al modelo, mejora el rendimiento del proceso de entrenamiento, y por tanto facilita en gran medida la labor de estimación de parámetros. Pero esto resulta apropiado sólo en algunas circunstancias y no con espacios de estados tan grandes.

En cualquier caso, la interpolación lineal no es la única manera de enfrentarse al problema de la poca frecuencia de determinados sucesos en los datos de entrenamiento. Existen otros métodos de estimación que, aunque son también dependientes del número de veces que aparecen los sucesos en el corpus de entrenamiento, no se basan en absoluto en frecuencias relativas. Como consecuencia, se han sugerido muchas fórmulas que son capaces de asignar probabilidades distintas de cero a los sucesos no observados, permitiendo así hacer una importante distinción entre este tipo de *ceros no observados*, y los *ceros reales* que serían los correspondientes a los sucesos efectivamente no posibles.

Entre todas ellas, la más conocida es la fórmula de Good-Turing[15]. La sección 4.5 está especialmente dedicada al estudio detallado de ésta y otras aproximaciones, incluida la que actualmente comparte el estado del arte con la interpolación lineal, en lo que se refiere al tratamiento de datos dispersos dentro del ámbito de los sistemas de etiquetación: el método de *marcha atrás* o *back-off* [16].

[15]La idea intuitiva del método de estimación de Good-Turing es que, en lugar de estimar $P(X)$, intentamos estimar $P(X|C)$, donde C es una clasificación de sucesos definida *a priori* (en nuestro caso, los sucesos X que aparecen en el corpus de entrenamiento exactamente C veces). De esta manera, es más sencillo estimar $P(X)$, y en concreto se puede estimar $P(X|0)$, que es precisamente la probabilidad de los sucesos que no aparecen en el corpus. En resumen, el método de Good-Turing corrige la estimación de máxima verosimilitud en base al número de veces que algo ocurre en el corpus.

[16]La idea general del método de estimación de marcha atrás o *back-off* es confiar en las frecuencias si hay un número suficiente de apariciones, utilizar Good-Turing si no lo hay, pero el suceso está todavía presente, y utilizar una hipótesis de menor nivel (como el suavizado lineal) si no está presente en absoluto. La estimación de menor nivel esta basada en la suposición que hemos visto anteriormente, de que $P(X|Y) = P(X)$ cuando X e Y son independientes.

4.4.3.3 Estimación combinada mediante métodos híbridos

Los etiquetadores basados en HMM,s trabajan bien cuando se dispone de un corpus de entrenamiento suficientemente grande. A menudo éste no es el caso. Suele ocurrir con frecuencia que queremos etiquetar un texto de un dominio especializado donde las probabilidades de emisión de las palabras son diferentes de las que se pueden observar en los textos de entrenamiento. Otras veces necesitamos etiquetar textos de idiomas para los cuales simplemente no existen *corpora* etiquetados. En estos casos, hemos visto anteriormente que se puede utilizar un procedimiento de estimación no visible o no supervisado a partir de una inicialización pseudo-aleatoria de los parámetros del modelo y de la posterior aplicación del algoritmo de Baum-Welch.

Los fundamentos teóricos sugieren que el algoritmo de Baum-Welch debe detenerse cuando no se puede construir un modelo que mejore la probabilidad de la secuencia de entrenamiento dada. Sin embargo, ha sido demostrado que, para el caso de la etiquetación, este criterio a menudo produce como resultado un sobreentrenamiento del modelo. Este fenómeno ha sido estudiado en profundidad por Elworthy, quien entrenó diferentes HMM,s considerando una gran variedad de condiciones de inicialización y de distintos números de iteraciones [Elworthy 1994]:

- En ocasiones, el proceso de entrenamiento se mostraba estable y siempre producía mejoras del rendimiento después de cada iteración, demostrando que el criterio probabilístico era apropiado para esos casos.

- Pero otras veces, la mejora aparecía sólo durante algunas iteraciones, normalmente 2 ó 3, y después el proceso de entrenamiento no hacía más que degradar el rendimiento.

Esto último se producía en lo que inicialmente se consideraban como los escenarios típicos de aplicación de los HMM,s: cuando se dispone de un diccionario, pero no se dispone de ningún corpus etiquetado. Por tanto, en este tipo de situaciones, los cuidados se deben extremar al máximo con el fin de no sobreentrenar el modelo. Una forma de conseguir esto es validar dicho modelo después de cada iteración sobre un conjunto de datos separado (*held-out data*), y parar el entrenamiento cuando el rendimiento empieza a decrecer.

Elworthy también confirmó los resultados encontrados por Merialdo, que demuestran que si se dipone inicialmente de un corpus etiquetado, la aplicación del algoritmo de Baum-Welch puede degradar el rendimiento ya desde la primera iteración [Merialdo 1994]. Sin embargo, una peculiaridad interesante es el hecho de que si el texto de entrenamiento y el de validación son muy diferentes, unas pocas iteraciones podrían producir mejoras. Además, esto suele ocurrir con frecuencia en la práctica, ya que a menudo tenemos que enfrentarnos con tipos de textos para los cuales no se dispone de *corpora* de entrenamiento etiquetados similares. En resumen:

- Si existe un texto de entrenamiento suficientemente grande y similar a los textos con los que se va a trabajar, entonces se debería utilizar un procedimiento de estimacion totalmente visible o supervisado.

- Si no existe ningún texto de entrenamiento disponible, o si los textos de entrenamiento y validación son muy diferentes, entonces se debería aplicar el algoritmo de Baum-Welch durante unas pocas iteraciones.

- Sólo se debería de aplicar durante un número grande de iteraciones, 10 ó más, cuando no hay ningún tipo de información léxica disponible.

Y en este último caso, no cabe esperar un buen rendimiento. Esto no se debe a ningún defecto del algoritmo de Baum-Welch, sino al hecho de que dicho algoritmo tan sólo realiza ajustes de los parámetros del HMM con el fin de maximizar la probabilidad de los datos de entrenamiento.

Los cambios que realiza para reducir la entropía cruzada de esos datos podrían no estar de acuerdo con nuestro objetivo real, que es el de asignar a las palabras etiquetas pertenecientes a un conjunto predefinido. Por tanto, esta técnica no siempre es capaz de optimizar el rendimiento para la tarea particular de la etiquetación.

4.4.3.4 Integración de diccionarios

Un problema similar al de las transiciones entre estados ocurre también con las palabras. En un primer momento podemos pensar que si estamos hablando de entrenamiento puramente supervisado, las probabilidades de generación de las palabras son lo único que sí se puede estimar perfectamente. Sin embargo, en la práctica, podemos encontrarnos con palabras que no aparecen en los textos de entrenamiento, pero que quizás sí las tenemos presentes en un diccionario. Es decir, se trata de palabras para las cuales conocemos sus posibles etiquetas, pero que al no haber sido vistas durante el entrenamiento, no tienen definidas sus probabilidades de emisión. Una vez más, no resulta conveniente dejar a cero esas probabilidades.

Por tanto, se hace necesario el uso de métodos que permitan la correcta integración de la información aportada por el diccionario con los datos proporcionados por el texto de entrenamiento. Existen dos métodos generales para realizar esta integración:

- **Método *Adding One* de Church.** El método *Adding One* o método *de sumar uno* utiliza el diccionario como si se tratara de un corpus etiquetado, de manera que cada uno de los pares palabra-etiqueta presentes en el diccionario aparece una sola vez en dicho corpus [Church 1988]. Por supuesto, este nuevo corpus se utilizará sólo para estimar las probabilidades de emisión de las palabras, pero no las probabilidades de transición entre estados, ya que el orden que siguen las palabras y las etiquetas dentro del diccionario no tiene nada que ver con el orden que siguen en las frases reales. De esta manera, para todo par (w^k, t^j) presente en el diccionario, la probabilidad de emisión se estima mediante

$$\hat{p}(w^k|t^j) = \frac{C(w^k|t^j) + 1}{C(t^j) + K_j}$$

  donde K_j es el número de palabras etiquetadas con t^j que aparecen en el diccionario, mientras que para el resto de pares palabra-etiqueta que aparezcan en el corpus de entrenamiento y no en el diccionario la estimación se realiza mediante la misma ecuación, pero sin sumar 1 en el numerador. Intuitivamente, volviendo al ejemplo de las urnas y las bolas, el método opera como si todas y cada unas de las bolas o palabras del diccionario fueran colocadas en sus urnas o etiquetas correspondientes una sola vez, como paso previo a la estimación de las probabilidades de emisión. Con esto se produce el efecto deseado de que las probabilidades de emisión de las palabras del diccionario no queden a cero, incluso aunque no hayan aparecido en el corpus de entrenamiento.

- **Método de Good-Turing.** Como ya hemos comentado anteriormente, éste es un método de estimación que no está basado en frecuencias relativas, pero que también es capaz de asignar probabilidades distintas de cero a los sucesos no observados. En este contexto, un suceso posible pero no observado es cualquier par palabra-etiqueta que aparece en el diccionario y no aparece en el corpus de entrenamiento (cero no observado), mientras que un suceso no posible es cualquier par palabra-etiqueta que no aparece ni en el diccionario ni en el corpus de entrenamiento (cero real). La sección 4.5 describe detalladamente la obtención de las fórmulas de Good-Turing.

Es importante recordar que uno de nuestros principales objetivos es comprobar de qué manera el uso de un diccionario externo puede ayudar a incrementar el rendimiento del proceso de

etiquetación. En el capítulo 7 se detallan las cifras de rendimiento para todos los etiquetadores que se han evaluado en el presente trabajo. Siempre que ha sido posible, dicha evaluación se ha realizado con y sin el uso de diccionarios externos.

4.4.3.5 Tratamiento de palabras desconocidas

Hemos visto anteriormente cómo estimar las probabilidades de emisión para las palabras que aparecen o bien en el corpus, o bien en nuestro diccionario, o bien en ambos. Pero es seguro que al intentar etiquetar nuevas frases encontraremos multitud de palabras que no han aparecido previamente en ninguno de esos recursos. Hemos visto también que un conocimiento *a priori* de la distribución de etiquetas de una palabra, o al menos de la proporción para la etiqueta más probable, supone una gran ayuda para la etiquetación. Esto significa que las palabras desconocidas son el mayor problema de los etiquetadores, y en la práctica la diferencia de rendimiento entre unos y otros puede ser debida a la proporción de palabras desconocidas y al procedimiento de adivinación y tratamiento de las mismas que cada etiquetador tenga integrado.

La manera más simple de enfrentarse a este problema es suponer que toda palabra desconocida puede pertenecer a cualquier categoría gramatical, o quizás a una clase de categorías *abiertas*, por ejemplo, sustantivos, adjetivos, verbos, etc., pero no preposiciones, ni artículos, ni pronombres, que suponemos que pertenecen a otra clase de categorías *cerradas* y que todas las palabras correspondientes a ellas son conocidas y están ya en el diccionario. Aunque esta aproximación podría ser válida en algunos casos, en general, el no hacer uso de la información léxica de las palabras (prefijos, sufijos, etc.) para proponer un conjunto más limitado de posibles etiquetas degrada mucho el rendimiento de los etiquetadores. Es por ello que existen numerosos trabajos orientados a explorar este tipo de características, y mejorar así la estimación de las probabilidades de emisión para las palabras desconocidas:

- Por ejemplo, Weischedel estima las probabilidades de generación de las palabras en base a tres tipos de información: la probabilidad de que una categoría genere una palabra desconocida[17], la probabilidad de que una categoría genere una palabra cuya inicial es mayúscula o minúscula, y la probabilidad de generación de sufijos particulares:

$$\hat{p}(w^k|t^j) = \frac{1}{Z}\, P(\text{desconocida}|t^j)\; P(\text{mayúscula}|t^j)\; P(\text{sufijo}|t^j)$$

  donde Z es una constante de normalización. Este modelo reduce la proporción de error de palabaras desconocidas de más de un 40% a menos de un 20% [Weischedel *et al.* 1993]. Charniak propuso un modelo alternativo que utiliza tanto las raíces como los sufijos [Charniak *et al.* 1993].

- La mayor parte del trabajo que se realiza en relación con las palabras desconocidas asume que las propiedades consideradas son independientes. Dicha independencia no siempre es una buena suposición. Por ejemplo, las palabras en mayúsculas tienen más probabilidad de ser desconocidas, y por tanto las propiedades *mayúscula* y *desconocida* del modelo de Weischedel no son realmente independientes. Franz desarrolló un modelo que tiene en cuenta este tipo de dependencias [Franz 1996, Franz 1997].

- Actualmente, el método de manejo de palabras desconocidas que parece ofrecer mejores resultados para los lenguajes flexivos es el análisis de sufijos mediante aproximaciones basadas en inferencias Bayesianas [Samuelsson 1993]. En este método, las probabilidades de las etiquetas propuestas para las palabras no presentes en el diccionario se eligen en

[17] Esta probabilidad puede ser cero para algunas categorías, como por ejemplo las preposiciones.

función de las terminaciones de dichas palabras. La distribución de probabilidad para un sufijo particular se genera a partir de todas las palabras del corpus de entrenamiento que comparten ese mismo sufijo. El término sufijo tal y como se utiliza aquí significa *secuencia final de caracteres de una palabra*, lo cual no coincide necesariamente con el significado lingüístico de sufijo. Por esta razón, el método requiere una definición previa de la longitud máxima de sufijos que se va a considerar.

Las probabilidades se suavizan mediante un procedimiento de abstracción sucesiva. Esto permite calcular $P(t|l_{n-m+1},\ldots,l_n)$, la probabilidad de una etiqueta t dadas las últimas m letras l_i de una palabra, a través de una secuencia de contextos más generales que omite un caracter del sufijo en cada iteración, de forma que la suavización se lleva a cabo en base a $P(t|l_{n-m+2},\ldots,l_n)$, $P(t|l_{n-m+3},\ldots,l_n)$, $\ldots$, $P(t)$. La fórmula de la recursión es:

$$P(t|l_{n-i+1},\ldots,l_n) = \frac{\hat{p}(t|l_{n-i+1},\ldots,l_n) + \theta_i\, P(t|l_{n-i+2},\ldots,l_n)}{1+\theta_i}$$

para $i = m,\ldots,1$, utilizando la estimación de máxima verosimilitud para un sufijo de longitud i calculada a partir de las frecuencias del corpus de entrenamiento como

$$\hat{p}(t|l_{n-i+1},\ldots,l_n) = \frac{C(t,l_{n-i+1},\ldots,l_n)}{C(l_{n-i+1},\ldots,l_n)},$$

utilizando también unos determinados pesos de suavización θ_i, y utilizando como caso base $P(t) = \hat{p}(t)$. Por supuesto, para el modelo de Markov, necesitamos las probabilidades condicionadas inversas $P(l_{n-i+1},\ldots,l_n|t)$, las cuales se obtienen por la regla de Bayes.

Como se puede ver, la definición de este método no es rigurosa en todos y cada uno de sus aspectos, lo cual da lugar a diferentes interpretaciones a la hora de aplicarlo. Por ejemplo:

- Es necesario identificar un buen valor para m, la longitud máxima utilizada para los sufijos. Brants elige para su etiquetador el criterio de que m depende de cada palabra en cuestión [Brants 2000], y utiliza para dicha palabra el sufijo más largo de entre todos los que hayan sido observados al menos una vez en el corpus de entrenamiento, pero como máximo $m = 10$ caracteres.
- Brants utiliza también una elección independiente del contexto para los pesos de suavización de sufijos θ_i, al igual que hacía con los parámetros de interpolación lineal de trigramas λ_i. Es decir, en lugar de calcular los θ_i tal y como propuso Samuelsson, a partir de la desviación estándar de las probabilidades de máxima verosimilitud de los sufijos, Brants asigna a todos los θ_i la desviación estándar de las probabilidades de máxima verosimilitud no condicionadas de las etiquetas en el corpus de entrenamiento:

  $$\theta_i = \frac{1}{s-1}\sum_{j=1}^{s}(\hat{p}(t^j) - \bar{P})^2$$

  para todo $i = m,\ldots,1$, donde s es el cardinal del conjunto de etiquetas utilizado, y la media $\bar{P}$ se calcula como

  $$\bar{P} = \frac{1}{s}\sum_{j=1}^{s}\hat{p}(t^j).$$

- Otro grado de libertad del método es el que concierne a la elección de las palabras del corpus de entrenamiento que se utilizan para la extracción de los sufijos. ¿Deberíamos utilizar todas las palabras, o algunas son más adecuadas que otras? Dado que las palabras desconocidas son probablemente las más infrecuentes, se puede argumentar

que el uso de los sufijos de las palabras menos frecuentes del corpus constituye una aproximación mejor para las palabras desconocidas que los sufijos de las palabras frecuentes. Brants, por tanto, realiza la extracción de sufijos sólo sobre aquellas palabras cuya frecuencia sea menor o igual que un cierto umbral de corte, y fija dicho umbral empíricamente en 10 apariciones. Además, mantiene dos pruebas de sufijos separadas, en función de si la inicial de la palabra es mayúscula o minúscula.

En general, para las palabras desconocidas, $P(w|t) = 0$ estrictamente hablando. En caso contrario, la palabra no sería desconocida. Pero tal y como hemos dicho, una de las principales habilidades que debe incluir un buen etiquetador es asignar una probabilidad $P(w|t)$ distinta de cero para las palabras no presentes en el diccionario. La única restricción que hay que respetar es que, para toda etiqueta t del conjunto de etiquetas, debería cumplirse

$$\sum_{w} P(w|t) = 1, \tag{4.16}$$

sobre todas las palabras w: las conocidas y las, en teoría, posiblemente desconocidas. Ésta es la clave del problema, porque implica que deberíamos conocer *a priori* todas las palabras desconocidas. En la práctica, esto no es así, y por tanto no conocemos completamente el conjunto de las probabilidades que estamos sumando. Una forma de afrontar el problema es considerar que no todas las palabras desconocidas juegan el mismo papel. Algunas de ellas serán efectivamente palabras nuevas para las cuales queremos probabilidades de emisión distintas de cero, pero otras podrían ser palabras con errores ortográficos para las cuales queremos probabilidades igual a cero o, en todo caso, las probabilidades de las correspondientes palabras corregidas.

Así pues, si el conjunto de palabras que van a ser aceptadas como desconocidas es *de alguna manera* conocido, entonces se puede dar un significado a la probabilidad $P(w|t)$, ya que la restricción (4.16) podría ser calculada y verificada. Si esta restricción se respeta, entonces podemos elegir la definición de $P(w|t)$ que queramos, y una definición basada en la morfología es, por supuesto, una buena elección. En otras palabras, definir $P(w|t)$ mediante $P(\mathit{sufijo}(w)|t)$ para las palabras desconocidas es una buena idea siempre que se cumpla la restricción (4.16).

Para ello, se podrían combinar las probabilidades de los sufijos con las probabilidades de emisión de las palabras conocidas, reservando una parte de la masa de probabilidad de éstas últimas. Es decir, si conocida la etiqueta t y para toda palabra w del diccionario,

$$S = \sum_{w} P(w|t),$$

donde $S < 1$, entonces $1 - S$ sería la masa de probabilidad dedicada a las palabras desconocidas, conocida t. $P(\mathit{sufijo}|t)$ debería ser una distribución de probabilidad, de tal forma que

$$\sum_{\mathit{sufijo}} P(\mathit{sufijo}|t) = 1, \tag{4.17}$$

para toda etiqueta t del conjunto de etiquetas utilizado. Y los sufijos deberían estar correctamente definidos, de tal manera que la función $\mathit{sufijo}(w)$ fuera una aplicación. Es decir, el sufijo de w debería ser único para cada palabra w. De esta forma, $P(w|t)$ se define para las palabras desconocidas como

$$P(w|t) = (1 - S)\, P(\mathit{sufijo}(w)|t),$$

lo cual no sólo es una definición formal, sino también sensible a las propiedades morfológicas de cada palabra. Las dificultades, por supuesto, radican en la identificación de los sufijos válidos para las palabras y en la estimación de un buen valor para S.

Tal y como vimos antes, la implementación que hace Brants del método de Samuelsson no mezcla las probabilidades de los sufijos con las probabilidades de las palabras conocidas, y por tanto no respeta la restricción (4.16). Sin embargo, dicho método construye un conjunto de probabilidades de emisión distinto para cada longitud de sufijo, y cada uno de esos conjuntos verifica la restricción (4.17). Como se puede apreciar, el enfoque es totalmente distinto, pero el método también genera probabilidades directamente utilizables por los modelos de Markov.

En definitiva, el tratamiento de palabras desconocidas[18] representa el punto más débil de la aplicación de los HMM,s al proceso de etiquetación del lenguaje natural, y no se incluye normalmente en el estado del arte de los HMM,s. En esta sección hemos esbozado sólo algunos de los métodos de tratamiento de palabras desconocidas que han sido desarrollados e integrados formalmente dentro del marco de los HMM,s.

4.5 Estimación de probabilidades a partir de contadores

En las secciones previas hemos asumido tácitamente que las frecuencias relativas de los sucesos son una buena estimación de sus probabilidades. La conocida ley de los grandes números nos asegura que hasta cierto punto esto es cierto. En particular, si $\mathcal{X}$ es el conjunto de posibles sucesos al repetir N experimentos distribuidos idénticamente, y si $C_N(x)$ es el número de veces que el suceso $x \in \mathcal{X}$ fue observado, entonces

$$P\left[\lim_{N\to\infty} \frac{C_N(x)}{N} = P(x)\right] = 1, \qquad \forall x \in \mathcal{X},$$

donde $P(x)$ denota la probabilidad de que un experimento dado dé como resultado el suceso x.

Ya que nuestra definición de frecuencia relativa $f(x)$ es la proporción $\frac{C_N(x)}{N}$, podría parecer que nuestro método de estimación de $P(x)$ es el mejor posible. Pero tenemos al menos tres problemas:

1. La estimación $\frac{C_N(x)}{N}$ para una determinada cantidad finita N de datos de entrenamiento puede no ser suficiente.

2. En muchas situaciones de interés, el espacio de sucesos $\mathcal{X}$ no se conoce en toda su totalidad. Es decir, el conjunto de sucesos que tienen lugar en los datos de entrenamiento puede ser sólo un subconjunto de $\mathcal{X}$.

3. Incluso cuando $\mathcal{X}$ es conocido, puede ser tan grande[19] que para muchos de los $x \in \mathcal{X}$ la probabilidad verdadera satisface $0 < P(x) \ll \frac{1}{N}$, de manera que el hecho de que x no haya sido observado en los datos de entrenamiento no implica en absoluto que su probabilidad deba ser 0.

Y todavía podrían surgir otras muchas cuestiones. Por ejemplo, el tercero de los problemas anteriores nos lleva a preguntarnos no sólo si un suceso no observado debería tener probabilidad 0 o no, sino también cuánto más grande debería ser la probabilidad de los sucesos observados una vez respecto a los no observados ninguna vez, o, en general, si la proporción de las probabilidades de dos sucesos observados n y m veces, respectivamente, debería ser realmente n/m.

En esta sección introduciremos primero un método de estimación de las probabilidades deseadas que emplea una aproximación basada en *datos extendidos* o *held-out data.* Posteriormente, una variante de esta aproximación nos llevará a las conocidas fórmulas de Good-Turing [Church y Gale 1991]. Y finalizaremos con una explicación detallada del método de

[18] También denominado *word guessing.*

[19] Por ejemplo, el conjunto de los posibles trigramas de etiquetas $t^i t^j t^k$ de un juego de etiquetas dado.

estimación de *marcha atrás* o *back-off* [Katz 1987], el cual constituye la base del funcionamiento de muchos de los modelos actuales de procesamiento automático del lenguaje.

4.5.1 Método de estimación mediante *held-out data*

La idea básica. En este método de estimación, la estrategia a seguir consiste en dividir los datos de entrenamiento $\mathcal{X}$ en dos partes [Jelinek y Mercer 1985]. La primera parte, llamada $\mathcal{D}$ (*datos de desarrollo* o *development data*), se usa para recolectar los contadores $C_d(x)$ de los sucesos x. La segunda parte, llamada $\mathcal{H}$ (*datos extendidos* o *held-out data*), se utilizará para estimar los parámetros adicionales que determinarán el valor final de los $\hat{p}(x)$, nuestras estimaciones de las probabilidades $P(x)$. Dichas estimaciones tendrán la siguiente estructura:

$$\hat{p}(x) = \begin{cases} q_i & \text{para todo } x \text{ tal que } C_d(x) = i, \text{donde } i = 0, 1, \ldots, M, \\ \alpha f_d(x) & \text{para todo } x \text{ tal que } C_d(x) > M, \text{donde } f_d(x) = \frac{C_d(x)}{N_d}, \end{cases} \tag{4.18}$$

y donde N_d es el tamaño del conjunto de datos $\mathcal{D}$. Los datos de $\mathcal{H}$ se utilizarán para encontrar los valores óptimos de los parámetros q_i y α que satisfagan la normalización

$$\sum_{x \in \mathcal{X}} \hat{p}(x) = 1. \tag{4.19}$$

La idea intuitiva que subyace en la ecuación (4.18) es que todos los sucesos observados el mismo número de veces i, con $i \in \{0, 1, \ldots, M\}$, deberían tener la misma probabilidad. El segundo caso, cuando $C_d(x) > M$, representa la suavización. Más adelante discutiremos cómo elegir un valor apropiado para el umbral de confianza M.

Por el momento, denotemos mediante n_i el número de símbolos diferentes $x \in \mathcal{X}$ para los cuales $C_d(x) = i$. Es decir,

$$n_i = \sum_{x \in \mathcal{X}} \delta(C_d(x), i)$$

donde la expresión $\delta(a, b)$ es la función de Kronecker definida como:

$$\delta(a, b) = \begin{cases} 1 & \text{si } a = b, \\ 0 & \text{en caso contrario.} \end{cases}$$

Entonces podemos dar una interpretación basada en clases a la ecuación (4.18). Un suceso x pertenecerá a la clase Φ_i si $C_d(x) = i$, para $i \in \{0, 1, \ldots, M\}$, y pertenecerá a la clase Φ_{M+1} si $C_d(x) > M$. Entonces

$$\hat{p}(x) = P(\Phi(x))\, P(x|\Phi(x)) \tag{4.20}$$

donde

$$P(x|\Phi(x)) = \begin{cases} \frac{1}{n_i} & \text{si } \Phi(x) = \Phi_i, \text{ donde } i = 0, 1, \ldots, M, \\ \frac{C_d(x)}{\sum_{j>M} j\, n_j} & \text{si } \Phi(x) = \Phi_{M+1}, \end{cases} \tag{4.21}$$

y utilizaremos los datos de $\mathcal{H}$ para estimar las probabilidades de ocupación $P(\Phi_i)$. Por supuesto, $q_i = P(\Phi_i)/n_i$ y $\alpha = P(\Phi_{M+1}) N_d / \sum_{j>M} j\, n_j$.

La estimación. Para determinar los valores de máxima verosimilitud de los q_i y de α, necesitamos notaciones adicionales. Denotaremos mediante:

- $C_h(x)$, el número de veces que el símbolo x aparece en $\mathcal{H}$,

- r_i, el número de veces que se han visto en $\mathcal{H}$ símbolos x tales que $C_d(x) = i$, es decir,

$$r_i = \sum_{x \in \mathcal{X}} C_h(x)\, \delta(C_d(x), i),$$

- r^*, el número de veces que $\mathcal{H}$ contiene símbolos x tales que $C_d(x) > M$, es decir,

$$r^* = \sum_{i>M} r_i,$$

- $|\mathcal{H}|$, el tamaño total del conjunto $\mathcal{H}$, es decir,

$$|\mathcal{H}| = \left(\sum_{i=0}^{M} r_i\right) + r^*. \tag{4.22}$$

La probabilidad de los datos de $\mathcal{H}$ calculada según la ecuación (4.18) es entonces igual a

$$P(\mathcal{H}) = \left(\prod_{i=0}^{M} (q_i)^{r_i}\right) \alpha^{r^*} K \tag{4.23}$$

donde

$$K = \prod_{x:C_d(x)>M} f_d(x)^{C_h(x)}.$$

Ahora podemos elegir los valores de q_i y α que maximizan la ecuación (4.23) y mantienen la restricción de la ecuación (4.19) mediante el método de los multiplicadores indeterminados de Lagrange. Esto es, lo que buscamos son las soluciones del siguiente conjunto de ecuaciones:

$$\frac{\partial}{\partial q_j}\left[P(\mathcal{H}) - \lambda\left(\sum_{i=1}^{M} n_i q_i + \alpha P_M\right)\right] = 0 \qquad j = 0, 1, \ldots, M$$

$$\frac{\partial}{\partial \alpha}\left[P(\mathcal{H}) - \lambda\left(\sum_{i=1}^{M} n_i q_i + \alpha P_M\right)\right] = 0$$

$$\sum_{i=1}^{M} n_i q_i + \alpha P_M = 1 \tag{4.24}$$

donde

$$P_M = \sum_{x:C_d(x)>M} f_d(x) = \frac{1}{N_d} \sum_{i>M} i\, n_i.$$

Calculando las derivadas parciales se sigue que

$$\frac{r_j}{q_j} P(\mathcal{H}) = \lambda n_j, \qquad j = 0, 1, \ldots, M,$$

y

$$\frac{r^*}{\alpha} P(\mathcal{H}) = \lambda P_M$$

y substituyéndolas en la ecuacion (4.24), se obtiene

$$\frac{P(\mathcal{H})}{\lambda}\left(\left(\sum_{i=0}^{M} r_i\right) + r^*\right) = 1.$$

Utilizando la definición (4.22), la solución final es

$$q_j = \frac{1}{n_j}\frac{r_j}{|\mathcal{H}|}, \qquad j = 0, 1, \ldots, M, \tag{4.25}$$

y

$$\alpha = \frac{1}{P_M}\frac{r^*}{|\mathcal{H}|}. \tag{4.26}$$

Es interesante resaltar en particular que

$$q_0 = \frac{1}{n_0}\frac{r_0}{|\mathcal{H}|} > 0$$

y que

$$\frac{q_{j+1}}{q_j} = \frac{r_{j+1}}{r_j}\frac{n_j}{n_{j+1}}, \qquad j = 0, 1, \ldots, M,$$

lo cual no es igual a $(j+1)/j$, el valor que habríamos obtenido si las probabilidades hubieran sido estimadas directamente mediante frecuencias relativas.

Elección del valor de M. Claramente, lo que queremos es que $\hat{p}(x)$ sea una función creciente de los contadores $C_d(x)$. Por tanto, debemos elegir M de manera que $q_j < q_{j+1}$ para $j = 0, 1, \ldots, M$. Esto quiere decir que debe verificarse

$$r_j\, n_{j+1} < r_{j+1}\, n_j, \qquad j = 0, 1, \ldots, M-1,$$

y

$$\frac{r_M}{n_M} < \frac{\sum_{j>M} r_j}{\sum_{j>M} j\, n_j}(M+1).$$

Utilizando los valores de (4.25) y (4.26), M debería ser elegido de forma que

$$\frac{r_M}{M\, n_M} \cong \frac{\sum_{j>M} r_j}{\sum_{j>M} j\, n_j}.$$

Universalidad de la estimación *held-out*. Para que la estimación de (4.18) con valores (4.25) y (4.26) que acabamos de obtener tenga una aplicabilidad universal sobre el tipo de datos en los que está basado el conjunto de entrenamiento, sólo necesitamos hacer uso de la interpretación de clases de las ecuaciones (4.20) y (4.21) y reescribir la fórmula (4.18) como sigue:

$$\hat{p}(x) = \begin{cases} \lambda_i \frac{1}{n_i} & \text{para todo } x \text{ tal que } C(x) = i, \text{ donde } i = 0, 1, \ldots, M, \\ \lambda_{M+1} \frac{C(x)}{\sum_{i>M} i\, n_i} & \text{para todo } x \text{ tal que } C(x) > M. \end{cases} \tag{4.27}$$

De esta manera, los coeficientes λ_i vienen a ser los pesos de un esquema de interpolación de borrado[20].

Comparando (4.18) y (4.27) obtenemos las siguientes relaciones:

$$\lambda_i = q_i n_i^d = \frac{r_i}{|\mathcal{H}|}, \qquad i = 0, 1, \ldots, M, \tag{4.28}$$

y

$$\lambda_{M+1} = \alpha \frac{\sum_{i>M} i\, n_i^d}{N_d} = \frac{r^*}{|\mathcal{H}|}$$

[20] *Deleted interpolation.*

donde se ha utilizado una d diacrítica en n_i^d y N_d para indicar que estamos tratando con las cantidades correspondientes a un conjunto particular $\mathcal{D}$. Por tanto, primero utilizamos los contadores de la parte $\mathcal{D}$ para calcular los λ_i, y una vez que estos valores están fijados mejoramos las estimaciones finales $\hat{p}(x)$ mediante los cálculos de la fórmula (4.27), la cual utiliza los contadores $C(x)$ obtenidos no sólo de la parte $\mathcal{D}$, sino a lo largo de todo el corpus de entrenamiento disponible ($\mathcal{D} \cup \mathcal{H}$).

4.5.2 Método de estimación de Good-Turing

Existe otra fórmula particular para estimar probabilidades a partir de contadores, que no depende directamente de una división del corpus de entrenamiento en los conjuntos de datos $\mathcal{D}$ (*development*) y $\mathcal{H}$ (*held-out*). Se trata de la estimación de Good-Turing, la cual se puede derivar mediante varios métodos. Una de las posibles derivaciones, la que veremos a continuación, está fuertemente relacionada con el concepto de universalidad que hemos visto anteriormente [Nadas 1991].

Primero creamos una serie de conjuntos pseudo-*held-out* mediante el método de *dejar uno fuera*[21] [Quenouille 1949, Ney *et al.* 1995]. Esto es, creamos N pares de conjuntos diferentes $\mathcal{D}_j$ y $\mathcal{H}_j$ a partir del conjunto de entrenamiento $\mathcal{T}$ de tamaño N, simplemente omitiendo cada uno de los elementos x_j de $\mathcal{T} = \{x_1, x_2, \ldots, x_N\}$. Por tanto, nuestros pares de conjuntos *development/held-out* serán de la forma:

$$\mathcal{D}_j = \mathcal{T} - x_j, \qquad \mathcal{H}_j = \{x_j\}, \qquad j = 1, 2, \ldots, N.$$

Lo que haremos ahora será predecir $\mathcal{H}_j$ en base a las estimaciones obtenidas a partir de $\mathcal{D}_j$, y elegiremos nuestros parámetros de manera que se maximice el producto

$$\prod_{j=1}^{N} P_j(\mathcal{H}_j) = \prod_{j=1}^{N} P_j(x_j)$$

donde $P_j(x_j)$ denota la probabilidad de x_j en base a los contadores obtenidos a partir del conjunto $\mathcal{D}_j$.

Sean $C(x)$ y n_i los contadores obtenidos a partir del conjunto completo de entrenamiento $\mathcal{T}$. Es decir, n_i es el número de símbolos diferentes x para los cuales $C(x) = i$. Denotemos también mediante $C_j(x)$ el contador de x en $\mathcal{D}_j$, y mediante r_i el número de veces que se han visto en la totalidad de los conjuntos *held-out* $\mathcal{H}_j$, $j = 1, 2, \ldots, N$, símbolos x tales que $C_j(x) = i$. Entonces,

$$r_i = (i+1)\, n_{i+1}. \tag{4.29}$$

En efecto, si $\mathcal{H}_j = x$, entonces $C_j(x) = C(x) - 1$. Por tanto, si $C_j(x) = i$ entonces $C(x) = i+1$. Y además, en $\mathcal{T}$ existen n_{i+1} de esos elementos x, y por tanto existen $(i+1)\, n_{i+1}$ valores diferentes de $j \in \{1, 2, \ldots, N\}$ para los cuales siendo $\mathcal{H}_j = x$ se verifica que $C_j(x) = i$, lo cual demuestra la certeza de la ecuación (4.29).

Dado que en este caso $|\mathcal{H}| = N$, obtenemos a partir de (4.25) que

$$q_j = \frac{n_{j+1}}{n_j} \frac{j+1}{N}, \qquad j = 0, 1, \ldots, M. \tag{4.30}$$

El valor de α se obtiene a partir de la normalización $\sum \hat{p}(x) = 1$. Esto es, a partir de

$$\sum_{j=0}^{M} q_j\, n_j + \alpha \sum_{j>M} \frac{j}{N}\, n_j = 1$$

[21] También denominado método *leave-one-out*.

y de (4.30) obtenemos

$$\alpha = \frac{\sum_{j>M+1} j\, n_j}{\sum_{j>M} j\, n_j}. \tag{4.31}$$

De esta manera, hemos derivado en (4.30) y (4.31) las conocidas fórmulas de Good-Turing, y la restricción de monotonía natural $q_{j-1} < q_j$ requiere ahora la elección de un umbral de confianza M que satisfaga:

$$(n_j)^2 < \frac{j+1}{j}\, n_{j-1}\, n_{j+1}, \qquad j = 1, 2, \ldots, M, \tag{4.32}$$

y

$$\frac{n_{M+1}}{n_M} < \frac{\sum_{j>M+1} j\, n_j}{\sum_{j>M} j\, n_j}. \tag{4.33}$$

Queremos llamar ahora la atención sobre el hecho de que aunque los números de aparición n_j para $j = 1, 2, \ldots, M+1$ son los que realmente se observan en el conjunto de datos $\mathcal{T}$, el número n_0 es diferente. Este número se infiere como el tamaño total del espacio de sucesos $\mathcal{X}$, que presumiblemente conocemos, menos el tamaño del subespacio $\mathcal{X}_\mathcal{T}$ observado realmente en $\mathcal{T}$. Es decir,

$$n_0 = |\mathcal{X}| - \sum_{i=1}^{\infty} n_i.$$

Por ejemplo, si $\mathcal{X}$ es el conjunto de todos los trigramas de $t^i t^j t^k$ pertenecientes a un juego de etiquetas dado de cardinal t, entonces $|\mathcal{X}| = t^3$, mientras que $\mathcal{X}_\mathcal{T}$ consistiría simplemente en el conjunto de todos los trigramas diferentes observados realmente en los datos de entrenamiento $\mathcal{T}$. Es interesante entonces destacar a partir de la ecuación (4.30) que la probabilidad total $q_0\, n_0$ asignada por las fórmulas de Good-Turing a los sucesos no observados es igual a n_1/N, esto es, la probabilidad total que sería asignada a los sucesos observados una sola vez mediante una fórmula de frecuencias relativas.

Observamos también a partir de la ecuación (4.31) que, en comparación con la fórmula de frecuencias relativas, el método de Good-Turing descuenta a los sucesos de frecuencia alta la cantidad

$$\frac{(M+1)\, n_{M+1}}{\sum_{j>M} j\, n_j}.$$

De hecho, la manera en que se dota de probabilidad a los sucesos no observados es a través de una secuencia de frecuencias relativas desplazadas desde los contadores más altos a los más bajos.

Finalmente, dado que M es normalmente un número entero pequeño, es importante introducir la fórmula de optimización

$$\sum_{j>M} j\, n_j = N - \sum_{i=1}^{M} i\, n_i$$

que indica que el sumatorio de la parte izquierda de la igualdad se puede calcular rápidamente mediante un sumatorio mucho más sencillo, el de la parte derecha, y una resta posterior.

4.5.3 Aplicabilidad de las estimaciones *held-out* y Good-Turing

Está claro que el método de estimación de Good-Turing fue diseñado para situaciones en las cuales $n_0 \gg n_1 \gg n_2$, es decir, para casos en los que hay muchos sucesos no observados y los datos aparecen realmente dispersos. De hecho, la fórmula (4.30) no es aplicable para $j = 0$ cuando todos los elementos de $\mathcal{X}$ han sido observados en los datos de entrenamiento, es decir, cuando $n_0 = 0$. Por otra parte, el estimador *held-out* siempre es aplicable: si $n_0 = 0$, entonces necesariamente $r_0 = 0$, pero la fórmula (4.28) permanece válida. No obstante, es fácil incurrir en usos abusivos de este método, por ejemplo cuando se considera un conjunto de datos *held-out* excesivamente pequeño, o simplemente cuando la partición entre los datos de desarrollo y los datos extendidos no está correctamente balanceada.

En todo caso, si el método de estimación mediante *held-out data* es siempre válido, ¿por qué no lo es el método de Good-Turing, que ha sido derivado a partir del estimador *held-out*? La razón es que hemos asumido desde el principio una cierta regularidad en nuestras observaciones de sucesos. Suponemos que los sucesos que fueron observados una sola vez son muy raros y perfectamente podían no haber sido observados en absoluto. O, en general, que los sucesos que fueron observados $j+1$ veces también podían haber sido observados sólo j veces, y esta situación queda perfectamente representada por los cálculos del método de *dejar uno fuera*, más robustos en este sentido. Por supuesto, esto es asumible sólo para grandes espacios de sucesos, donde además los sucesos son, por así decirlo, anónimos. Es decir, la frecuencia de aparición de un suceso no tiene nada que ver con la frecuencia de aparición de otro suceso diferente.

Cuando trabajamos con modelos de lenguaje basados en trigramas, estamos interesados en probabilidades de la forma $P(t_i|t_{i-2}, t_{i-1})$, que, una vez más, abusando un poco de la notación, y para no complicar en exceso las ecuaciones con la presencia de tantos subíndices, denotaremos mediante $P(t_3|t_1, t_2)$, donde de manera obvia los números 1, 2 y 3 hacen referencia a la posición de cada etiqueta dentro del trigrama. En este caso, existe una tentación obvia de utilizar el método de estimación de Good-Turing directamente, es decir, de usar contadores $n_i(t_1, t_2)$ del número de etiquetas diferentes t_3 que han seguido al bigrama de etiquetas $t_1\, t_2$ exactamente i veces en los datos de entrenamiento. Sin embargo, debemos resistir esta tentación porque en la mayoría de los casos esos contadores podrían estar basados en datos muy reducidos. Para modelos de lenguaje basados en trigramas lo que realmente necesitaríamos calcular es

$$P(t_3|t_1, t_2) = \frac{P_T(t_1, t_2, t_3)}{\sum_{t'_3} P_T(t_1, t_2, t'_3)} \tag{4.34}$$

donde sólo las probabilidades P_T serían el resultado de una estimación Good-Turing. La fórmula (4.34) sería computacionalmente bastante factible, ya que el denominador se puede calcular de manera previa:

$$\sum_{t'_3} P_T(t_1, t_2, t'_3) = \sum_{i=0}^{M} q_i\, n_i(t_1, t_2) + \frac{\alpha}{N} \sum_{i=M+1}^{\infty} i\, n_i(t_1, t_2)$$

donde q_i y α vendrían dados por las ecuaciones (4.30) y (4.31), respectivamente.

4.5.4 Mejora de los métodos de estimación

Ambos métodos de estimación, *held-out* y Good-Turing, están basados en la estructura (4.18), la cual dice que la probabilidad de cada elemento de $\mathcal{X}$ debería determinarse mediante su frecuencia en el conjunto de datos de desarrollo. Pero podría ocurrir que tuviéramos algún tipo de información disponible que nos permitiera hacer distinciones entre elementos con la misma frecuencia de aparición. Church y Gale apuntaron ideas sobre cómo se puede hacer uso de esta información adicional [Church y Gale 1991].

Supongamos que el espacio de sucesos $\mathcal{X}$ es el conjunto de los bigramas de $t^i\,t^j$ pertenecientes a un juego de etiquetas dado. ¿Existe alguna razón para pensar que dos bigramas diferentes, ninguno de los cuales ha sido observado, puedan tener probabilidades diferentes? He aquí un posible argumento: si tanto t^i como t^j son frecuentes, entonces es muy probable que no haya sido simplemente mala suerte el hecho de que por ejemplo el bigrama $t^i\,t^j$ no aparezca; por otra parte, si t^i y t^j no son frecuentes, su no aparición no debería implicar automáticamente una *alergia* entre ellos.

Por tanto, una manera de obtener una estimación mejor es categorizar o clasificar cada bigrama $t^i\,t^j$, no sólo por su frecuencia de aparición, sino también por su producto de probabilidad $\hat{p}(t^i)\,\hat{p}(t^j)$, donde normalmente la estimación $\hat{p}(t) = f(t)$, porque dado que todos los unigramas aparecen, la frecuencia $f(t)$ sí estará basada en datos suficientes como para constituir una estimación precisa de esa probabilidad. Sin embargo, no será posible basar las categorías de sucesos en el valor exacto $\hat{p}(t^i)\,\hat{p}(t^j)$. Es cierto que se podría realizar una ponderación utilizando una función continua sobre esos valores. Pero en este caso nos interesa establecer una clasificación, de manera que habrá que particionar el intervalo $[0, 1]$ en subintervalos y clasificar $t^i\,t^j$ mediante el subintervalo en el que cae $\hat{p}(t^i)\,\hat{p}(t^j)$. Church y Gale recomiendan que esos subintervalos sean logarítmicos y que su tamaño exacto se determine de manera que la clase o subintervalo correspondiente contenga un número suficiente de bigramas diferentes. Resulta intuitivamente obvio que los contadores $C(t^i, t^j)$ bajos deberían dividirse en más subclases que los contadores altos.

Para encontrar las fórmulas de esta estimación mejorada, basta pensar en una interpretación basada en clases de la estructura (4.18), que ya habíamos presentado en las ecuaciones (4.20) y (4.21). En esta interpretación, dos sucesos x e y pertenecían a una misma clase Φ_i, si $C(x) = C(y) = i$. Pues bien, considerando de nuevo esta interpretación, encontramos que todas las ecuaciones finales, (4.25) y (4.26) para la estimación *held-out*, y (4.30) y (4.31) para la estimación Good-Turing, pueden permanecer invariables con sólo definir una partición de clases Φ basada en otro criterio de clasificación diferente [Jelinek 1997, pp. 268-270]. Es decir, considerando una partición que clasifique un suceso x mediante un criterio más acorde con las ideas expuestas en el párrafo anterior, y no basado simplemente en los contadores de aparición $C(x)$, las fórmulas de estimación ya vistas continuarán siendo válidas y dichas estimaciones mejorarán, en la medida en que Φ agrupe mejor los sucesos.

Debido a esto, basta cambiar Φ para aplicar diferentes criterios de clasificación, y por tanto existirán muchas posibles mejoras. La siguiente sección presta atención a la principal de todas ellas: el método de estimación de *marcha atrás* o *back-off*.

4.5.5 Método de estimación *back-off*

Tal y como hemos apuntado ya desde las primeras secciones del presente capítulo, cualquier estimación de las probabilidades de un modelo de lenguaje que dependan de una cierta *historia*, es decir, no basados simplemente en los unigramas, necesariamente sufre el problema de los datos dispersos. Hasta ahora, habíamos intentado aliviar este problema mediante interpolaciones lineales. Por ejemplo, para los trigramas de etiquetas, definíamos las probabilidades del modelo mediante

$$P(t_3|t_1, t_2) = \lambda_3\, f(t_3|t_1, t_2) + \lambda_2\, f(t_3|t_2) + \lambda_1\, f(t_3) \tag{4.35}$$

donde los pesos λ_i se estimaban mediante datos *held-out*. Sin embargo, Katz argumentó que si el contador $C(t_1, t_2, t_3)$ es suficientemente grande, entonces $f(t_3|t_1, t_2)$ es en sí mismo un criterio de categorización que porporciona una estimación mucho mejor de $P(t_3|t_1, t_2)$ que la mostrada

anteriormente en la ecuación (4.35) [Katz 1987], de manera que el esquema

$$\hat{p}(t_3|t_1,t_2) = \begin{cases} f(t_3|t_1,t_2) & \text{si } C(t_1,t_2,t_3) \geq K \\ \alpha\, Q_T(t_3|t_1,t_2) & \text{si } 1 \leq C(t_1,t_2,t_3) < K \\ \beta(t_1,t_2)\,\hat{p}(t_3|t_2) & \text{en caso contrario} \end{cases} \tag{4.36}$$

constituye una aproximación superior, una vez que los coeficientes α y $\beta(t_1,t_2)$ hayan sido convenientemente elegidos[22]. En el esquema (4.36), $Q_T(t_3|t_1,t_2)$ es una función de tipo Good-Turing, tal y como veremos más adelante, y $\hat{p}(t_3|t_2)$ es la probabilidad del bigrama estimada mediante un esquema similar:

$$\hat{p}(t_3|t_2) = \begin{cases} f(t_3|t_2) & \text{si } C(t_2,t_3) \geq K \\ \alpha'\, Q_T(t_3|t_2) & \text{si } 1 \leq C(t_2,t_3) < K \\ \beta(t_2)\, f(t_3) & \text{en caso contrario.} \end{cases} \tag{4.37}$$

Por lo tanto, la definición de la ecuación (4.36) es una definición recursiva[23].

¿Cómo determinamos las funciones Q_T? Si el umbral de confianza K fuera infinito, Katz sugiere que $\alpha = 1$, y tenemos por definición que

$$Q_T(t_3|t_1,t_2) = \frac{P_T(t_1,t_2,t_3)}{f(t_1,t_2)} \tag{4.38}$$

donde el numerador es la probabilidad Good-Turing obtenida a partir de la colección de contadores $C(t_1,t_2,t_3)$ mediante las fórmulas de la sección 4.5.2. La definición (4.38) de Q_T no constituye una probabilidad porque no está normalizada adecuadamente, lo cual ocurriría sólo si $f(t_1,t_2)$ fuera reemplazado por $\sum_{t'_3} P_T(t_1,t_2,t'_3)$. No obstante, la fórmula (4.38) es simple y permite una elección sencilla de los coeficientes α y β. Nos ocuparemos de la normalización de los $\hat{p}(t_3|t_1,t_2)$ resultantes al final de nuestro desarrollo.

Sin embargo, dado que K es finito (Katz recomienda $K \cong 6$), los cálculos son un poco más complejos. Nos interesa determinar α de manera que satisfaga el siguiente principio básico: la probabilidad total reservada a todos los trigramas no observados[24] debería ser n_1/N, tal y como prescribe el método de Good-Turing cuando $K = \infty$. Entonces tenemos que

$$n_1 = \sum_{t_1,t_2,t_3} \delta(C(t_1,t_2,t_3),1)$$

y

$$N = \sum_{t_1,t_2,t_3} C(t_1,t_2,t_3).$$

Ahora, mediante la fórmula (4.30),

$$P_T(t_1,t_2,t_3) = q_i = \frac{n_{i+1}}{n_i}\frac{i+1}{N}, \qquad \text{donde } i = C(t_1,t_2,t_3),$$

[22] Entre otras restricciones, se les exige que aseguren la normalización estándar de las probabilidades $\hat{p}(t_3|t_1,t_2)$; obsérvese también que el coeficiente α es constante, pero el coeficiente β es variable y depende del bigrama t_1t_2 que constituye la *historia* del trigrama $t_1t_2t_3$ que se está tratando.

[23] En ambos esquemas, (4.35) y (4.36), se asume que $f(t_1,t_2) > 0$; si esto no es así, entonces en (4.35) el primer sumando es 0, mientras que (4.36) simplemente no es aplicable y por definición tenemos que $\hat{p}(t_3|t_1,t_2) = \hat{p}(t_3|t_2)$, estando este último término definido por (4.37); nótese finalmente que, dado que utilizaremos siempre textos de entrenamiento en los cuales cada etiqueta estará siempre presente al menos una vez, $f(t_3) > 0$ para todo t_3 perteneciente al juego de etiquetas utilizado.

[24] Nos referimos realmente a la probabilidad de los trigramas, no a la probabilidad condicionada de la tercera etiqueta dadas las dos primeras.

y

$$f(t_1,t_2,t_3) = f_i = \frac{i}{N}, \qquad \text{donde } i = C(t_1,t_2,t_3),$$

y, por tanto, considerando la restricción $\sum \hat{p}(t_1,t_2,t_3) = 1$ y el principio citado anteriormente, la obtención de α es como sigue:

$$\sum_{i=K}^{\infty} f_i\, n_i + \alpha \sum_{i=1}^{K-1} q_i\, n_i + \frac{n_1}{N} = 1 \;\Rightarrow\; \sum_{i=K}^{\infty} \frac{i}{N}\, n_i + \alpha \sum_{i=1}^{K-1} \frac{n_{i+1}}{n_i}\frac{i+1}{N}\, n_i + \frac{n_1}{N} = 1 \;\Rightarrow$$

$$\Rightarrow\; \sum_{i=K}^{\infty} i\, n_i + \alpha \sum_{i=1}^{K-1} n_{i+1}\,(i+1) + n_1 = N \;\Rightarrow\; \sum_{i=K}^{\infty} i\, n_i + \alpha \underbrace{\sum_{i=1}^{K-1} (i+1)\, n_{i+1}}_{\sum_{i=2}^{K} i\, n_i} = \underbrace{N - n_1}_{\sum_{i=2}^{\infty} i\, n_i} \;\Rightarrow$$

$$\Rightarrow\; \alpha = \frac{\sum_{i=2}^{\infty} i\, n_i - \sum_{i=K}^{\infty} i\, n_i}{\sum_{i=2}^{K} i\, n_i} \;\Rightarrow\; \alpha = \frac{\sum_{i=2}^{K-1} i\, n_i}{\sum_{i=2}^{K} i\, n_i}.$$

Finalmente, para que los $\hat{p}(t_3|t_1,t_2)$ estén normalizados, es decir, para que $\sum_{t_3} \hat{p}(t_3|t_1,t_2) = 1$, $\beta(t_1,t_2)$ ha de satisfacer

$$\beta(t_1,t_2) \sum_{t_3 \in \varphi_0} \hat{p}(t_3|t_2) = 1 - \sum_{t_3 \in \varphi_K} f(t_3|t_1,t_2) - \alpha \sum_{t_3 \in \varphi_*} Q_T(t_3|t_1,t_2)$$

donde $\varphi_0 = \{t_3 : C(t_1,t_2,t_3) = 0\}$, $\varphi_K = \{t_3 : C(t_1,t_2,t_3) \geq K\}$, y $\varphi_* = \{t_3 : 1 \leq C(t_1,t_2,t_3) < K\}$.

De manera similar, a partir de las restricciones $\sum \hat{p}(t_2,t_3) = 1$ y $\sum_{t_3} \hat{p}(t_3|t_2) = 1$, es posible derivar las fórmulas para calcular el valor de α' y de $\beta(t_2)$, respectivamente, para el esquema (4.37).

Dichas fórmulas aparecen en el resumen que mostramos a continuación, el cual describe de nuevo los cálculos involucrados en los esquemas (4.36) y (4.37), pero esta vez con todos los detalles que resultan necesarios para poder realizar una correcta implementación del método de estimación *back-off* a partir de los contadores observados en los datos de entrenamiento. Así pues, de acuerdo con el esquema (4.36), $\hat{p}(t_3|t_1,t_2)$ se define como sigue:

- Si $C(t_1,t_2,t_3) \geq K$, $\hat{p}(t_3|t_1,t_2) = f(t_3|t_1,t_2)$, donde

$$f(t_3|t_1,t_2) = \frac{C(t_1,t_2,t_3)}{C(t_1,t_2)}.$$

- Si $1 \leq C(t_1,t_2,t_3) = i < K$, $\hat{p}(t_3|t_1,t_2) = \alpha\, Q_T(t_3|t_1,t_2)$, donde

$$\alpha = \frac{\sum_{j=2}^{K-1} j\, n_j}{\sum_{j=2}^{K} j\, n_j}$$

siendo n_j el número de trigramas diferentes cualesquiera que han aparecido j veces en los datos de entrenamiento, y donde

$$Q_T(t_3|t_1,t_2) = \frac{P_T(t_1,t_2,t_3)}{f(t_1,t_2)} = \frac{q_i}{f(t_1,t_2)} = \frac{\frac{n_{i+1}}{n_i}\frac{i+1}{N_3}}{\frac{C(t_1,t_2)}{N_2}}$$

siendo N_3 y N_2 el número total de trigramas y bigramas, respectivamente, que han sido observados en los datos de entrenamiento.

- Si $C(t_1,t_2,t_3) = 0$, $\hat{p}(t_3|t_1,t_2) = \beta(t_1,t_2)\,\hat{p}(t_3|t_2)$, donde

$$\beta(t_1,t_2) = \frac{1 - \sum_{t_3\in\varphi_K} f(t_3|t_1,t_2) - \alpha \sum_{t_3\in\varphi_*} Q_T(t_3|t_1,t_2)}{\sum_{t_3\in\varphi_0} \hat{p}(t_3|t_2)}$$

siendo $\varphi_K = \{t_3 : C(t_1,t_2,t_3) \geq K\}$, $\varphi_* = \{t_3 : 1 \leq C(t_1,t_2,t_3) < K\}$ y $\varphi_0 = \{t_3 : C(t_1,t_2,t_3) = 0\}$, y donde $\hat{p}(t_3|t_2)$ viene dado por los cálculos del esquema (4.37).

De acuerdo con el esquema (4.37), $\hat{p}(t_3|t_2)$ se define como sigue:

- Si $C(t_2,t_3) \geq K$, $\hat{p}(t_3|t_2) = f(t_3|t_2)$, donde

$$f(t_3|t_2) = \frac{C(t_2,t_3)}{C(t_2)}.$$

- Si $1 \leq C(t_2,t_3) = i < K$, $\hat{p}(t_3|t_2) = \alpha'\,Q_T(t_3|t_2)$, donde

$$\alpha' = \frac{\sum_{j=2}^{K-1} j\,n'_j}{\sum_{j=2}^{K} j\,n'_j}$$

siendo n'_j el número de bigramas diferentes cualesquiera que han aparecido j veces en los datos de entrenamiento, y donde

$$Q_T(t_3|t_2) = \frac{P_T(t_2,t_3)}{f(t_2)} = \frac{q'_i}{f(t_2)} = \frac{\frac{n'_{i+1}}{n'_i}\frac{i+1}{N_2}}{\frac{C(t_2)}{N_1}}$$

siendo N_2 y N_1 el número total de bigramas y unigramas, respectivamente, que han sido observados en los datos de entrenamiento.

- Si $C(t_2,t_3) = 0$, $\hat{p}(t_3|t_2) = \beta(t_2)\,f(t_3)$, donde

$$\beta(t_2) = \frac{1 - \sum_{t_3\in\varphi'_K} f(t_3|t_2) - \alpha' \sum_{t_3\in\varphi'_*} Q_T(t_3|t_2)}{\sum_{t_3\in\varphi'_0} f(t_3)}$$

siendo $\varphi'_K = \{t_3 : C(t_2,t_3) \geq K\}$, $\varphi'_* = \{t_3 : 1 \leq C(t_2,t_3) < K\}$ y $\varphi'_0 = \{t_3 : C(t_2,t_3) = 0\}$, y donde $f(t_3) = \frac{C(t_3)}{N_1}$.

Finalmente, queremos destacar que aunque el valor del coeficiente K es constante, no tiene porqué ser el mismo en ambos esquemas de estimación. Es decir, resulta perfectamente factible la consideración de un coeficiente K para el esquema (4.36), y de otro coeficiente K' distinto para el esquema (4.37). Es más, al igual que en el método de estimación Good-Turing calculábamos el valor del umbral de confianza M, ambos coeficientes K y K' también deberían ser convenientemente estimados directamente sobre los datos de entrenamiento mediante las restricciones (4.32) y (4.33), para poder efectuar una aplicación totalmente rigurosa del método de estimación *back-off*.

4.6 Variantes de implementación de los HMM,s

Las predicciones condicionadas sobre una historia o contexto de gran longitud no son siempre una buena idea. Por ejemplo, normalmente no existen dependencias sintácticas entre las palabras que aparecen antes y después de las comas. Por tanto, el conocimiento de la etiqueta que aparece antes de una coma no siempre ayuda a determinar correctamente la que aparece después. De hecho, ante este tipo de casos, un etiquetador basado en trigramas podría hacer predicciones peores que otro basado en bigramas, debido una vez más al fenómeno de los datos dispersos[25]. La técnica de interpolación lineal de las probabilidades de los unigramas, bigramas y trigramas presentada en las secciones anteriores, y el método de *back-off* que ya hemos descrito también detalladamente, son algunas de las aproximaciones que tratan de paliar este problema, pero no son las únicas.

Algunos autores han optado por aumentar de manera selectiva un modelo de Markov de orden bajo, tomando como base para ello el análisis de los errores cometidos y algún otro tipo de conocimiento lingüístico previo. Por ejemplo, Kupiec observó que un HMM de orden 1 se equivocaba sistemáticamente al etiquetar la secuencia de palabras `the bottom of` como artículo-adjetivo-preposición. Entonces extendió el modelo con una red especial para esta construcción, con el propósito de que el etiquetador pudiera aprender la imposibilidad de que una preposición siga a la secuencia artículo-adjetivo [Kupiec 1992]. En general, este método selecciona manualmente determinados estados del modelo y aumenta su orden, para casos donde la *memoria* de orden 1 no es suficiente.

Otro método relacionado es el de los Modelos de Markov de memoria variable o VMMM,s[26] [Schütze y Singer 1994, Triviño y Morales 2000]. Los VMMM,s presentan estados de diferentes *longitudes*, en lugar de los estados de *longitud fija* de los etiquetadores basados en bigramas o en trigramas. Un VMMM podría transitar desde un estado que recuerda las dos últimas etiquetas (correspondiente, por tanto, a un trigrama) a un estado que recuerda las tres últimas etiquetas (correspondiente a un cuatrigrama) y después a otro estado sin memoria (correspondiente a un unigrama). El número de símbolos a recordar para una determinada secuencia se determina durante el proceso de entrenamiento, en base a criterios teóricos. En contraste con la interpolación lineal, los VMMM,s condicionan la longitud de memoria utilizada durante la predicción a la secuencia actual, en lugar de considerar una suma ponderada fija para todas las secuencias. Los VMMM,s se construyen con estrategias descendentes mediante métodos de división de estados[27]. Una posible alternativa es construirlos con estrategias ascendentes mediante métodos de fusión de modelos[28] [Stolcke y Omohundro 1994, Brants 1998].

Otra aproximación que resulta todavía más potente es la constituida por los modelos

[25] Muchas veces, las probabilidades de transición de los trigramas se estiman en base a sucesos de rara aparición, y por tanto la posibilidad de obtener estimaciones malas es mayor.

[26] *Variable Memory Markov Models.*

[27] *State splitting.*

[28] *Model merging.*

de Markov jerárquicos no emisores [Ristad y Thomas 1997]. Mediante la introducción de transiciones sin emisiones[29], estos modelos pueden almacenar también dependencias de longitud arbitraria entre los estados.

4.7 Otras aplicaciones de los HMM,s

La teoría matemática relativa a los HMM,s fue desarrollada por Baum y sus colaboradores a finales de los años sesenta y principios de los setenta [Baum *et al.* 1970]. Los HMM,s se aplicaron al procesamiento de la voz o reconocimiento del discurso hablado[30] en los años setenta por Baker [Baker 1975], y por Jelinek y sus colegas de IBM [Jelinek *et al.* 1975, Jelinek 1976]. Fue posteriormente cuando los HMM,s se utilizaron para modelizar otros aspectos de los lenguajes humanos, tales como el proceso de etiquetación.

Dentro del contexto del reconocimiento de voz, existen muy buenas referencias sobre los HMM,s y sus algoritmos [Levinson *et al.* 1983, Charniak 1993, Jelinek 1997]. Particularmente conocidas son también [Rabiner 1989, Rabiner y Juang 1993]. Todas ellas estudian en detalle tanto los HMM,s continuos, donde la salida es un valor real, como los HMM,s discretos que nosotros hemos considerado aquí. Aunque se centran en la aplicación de los HMM,s al reconocimiento de voz, resulta también muy recomendable su consulta para la obtención de directrices y de otras muchas referencias acerca del desarrollo y uso general de los HMM,s.

En este capítulo, hemos asumido en todo momento que la arquitectura de nuestro HMM era fija, y hemos centrado la discusión en torno al problema de la obtención de los parámetros óptimos para esa arquitectura. Sin embargo, ¿cuál es la forma y el tamaño que deberíamos elegir para nuestros HMM,s cuando nos enfrentamos a un nuevo problema? Normalmente, como en el caso de la etiquetación, es la naturaleza del problema la que determina de alguna manera la arquitectura del modelo. Para circunstancias en las que no es ese el caso, existen trabajos que estudian la construcción automática de la estructura de los HMM,s. Dichos trabajos se basan en el principio de intentar encontrar el HMM más compacto posible que describa adecuadamente los datos [Stolcke y Omohundro 1993].

Los HMM,s han sido también ampliamente utilizados en aplicaciones bioinformáticas para el análisis de secuencias de genes [Baldi y Brunak 1998, Durbin *et al.* 1998]. Es cierto que puede parecer increíble que el manejo de un alfabeto de sólo cuatro símbolos, las bases nitrogenadas del ADN, entrañe grandes dificultades, pero la bioinformática es un dominio perfectamente establecido, y en el que efectivamente se plantean problemas muy serios que pueden requerir el uso de modelizaciones complejas.

Y quizás la aplicación más reciente de los HMM,s dentro del marco del procesamiento de lenguaje natural es su uso directo como motor de búsqueda en sistemas de recuperación de información [Miller *et al.* 1999]. La idea básica de esta aproximación consiste una vez más en considerar la unión de todas las palabras que aparecen en el corpus como el conjunto de símbolos de salida del modelo, e identificar un estado distinto, en este caso no para cada etiqueta, sino para cada uno de los posibles mecanismos de generación de palabras en las consultas: términos temáticos, términos dependientes del documento, palabras que aparecen frecuentemente en las consultas, mecanismos de sinonimia, etc. Posteriormente, se construye un sencillo HMM individual para cada documento, a través del cual se puede calcular la probabilidad de que dicho documento genere las palabras involucradas en la consulta que efectúa el usuario. Esa probabilidad indica si el documento es relevante o no, y en función de ella se establece la lista ordenada de documentos que el sistema muestra al usuario como respuesta a su consulta. Los

[29] Transiciones entre estados que no emiten ninguna palabra o que, de manera equivalente, emiten la palabra vacía ϵ.

[30] *Speech recognition.*

resultados obtenidos con esta técnica son muy prometedores, y refuerzan la perspectiva de que los HMM,s continúan siendo aplicables incluso en temas de verdadera actualidad como es el de la recuperación de información.

Capítulo 5

Aprendizaje de etiquetas basado en transformaciones

Tradicionalmente, en lo que se refiere a la etiquetación de textos en lenguaje natural, y tal y como hemos visto en el capítulo anterior, se han preferido las aproximaciones puramente estocásticas frente a las aproximaciones basadas en reglas, debido a su buen rendimiento y sobre todo a sus facilidades de entrenamiento automático. Sin embargo, hemos visto también que algunas de las hipótesis de funcionamiento de los modelos de Markov no se adaptan del todo bien a las propiedades sintácticas de los lenguajes naturales. Debido a esto, inmediatamente surge la idea de utilizar modelos más sofisticados. Podríamos pensar por ejemplo en establecer condiciones que relacionen las nuevas etiquetas, no sólo con las etiquetas precedentes, sino también con las palabras precedentes. Podríamos pensar también en utilizar un contexto mayor que el de los etiquetadores basados en trigramas. Pero la mayoría de estas aproximaciones no tienen cabida dentro de los modelos de Markov, debido a la carga computacional que implican y a la gran cantidad de nuevos parámetros que necesitaríamos estimar. Incluso con los etiquetadores basados en trigramas hemos visto que es necesario aplicar técnicas de suavización e interpolación, ya que la estimación de máxima verosimilitud por sí sola no es lo suficientemente robusta.

Eric Brill presentó un sistema de etiquetación basado en reglas, el cual, a partir de un corpus de entrenamiento, infiere automáticamente las reglas de transformación [Brill 1993b], salvando así la principal limitación de este tipo de técnica que es precisamente el problema de cómo obtener dichas reglas. El etiquetador de Brill alcanza un rendimiento comparable al de los etiquetadores estocásticos y, a diferencia de éstos, la información lingüística no se captura de manera indirecta a través de grandes tablas de probabilidades, sino que se codifica directamente bajo la forma de un pequeño conjunto de reglas no estocásticas muy simples, pero capaces de representar interdependencias muy complejas entre palabras y etiquetas. Este capítulo describe las características principales del etiquetador de Brill y de su principio de funcionamiento: un paradigma de aprendizaje basado en transformaciones y dirigido por el error[1]. Veremos que este método es capaz de explorar un abanico mayor de propiedades, tanto léxicas como sintácticas, de los lenguajes naturales. En particular, se pueden relacionar etiquetas con palabras concretas, se puede ampliar el contexto precedente, e incluso se puede utilizar el contexto posterior.

5.1 Arquitectura interna del etiquetador de Brill

El etiquetador de Brill consta de tres partes, que se infieren automáticamente a partir de un corpus de entrenamiento: un etiquetador léxico, un etiquetador de palabras desconocidas, y un

[1] *Transformation-based error-driven learning.*

etiquetador contextual. A continuación se describe detalladamente cada una de ellas.

5.1.1 El etiquetador léxico

El etiquetador léxico etiqueta inicialmente cada palabra con su etiqueta más probable, sin tener en cuenta el contexto en el que dicha palabra aparece. Dicha etiqueta más probable se estima previamente mediante el estudio del corpus de entrenamiento. A las palabras desconocidas se les asigna en un primer momento la etiqueta correspondiente a sustantivo propio si la primera letra es mayúscula, o la correspondiente a sustantivo común en otro caso. Posteriormente, el etiquetador de palabras desconocidas aplica en orden una serie de reglas de transformación léxicas.

Si se dispone de un diccionario previamente construido, es posible utilizarlo junto con el que el etiquetador de Brill genera automáticamente. Más adelante veremos cómo se realiza esta integración y en qué circunstancias concretas este proceso ayuda a mejorar el rendimiento del etiquetador.

5.1.2 El etiquetador de palabras desconocidas

El etiquetador de palabras desconocidas opera justo después de que el etiquetador léxico haya etiquetado todas las palabras presentes en el diccionario, y justo antes de que se apliquen las reglas contextuales. Este módulo intenta *adivinar* una etiqueta para una palabra desconocida en función de su sufijo[2], de su prefijo, y de otras propiedades relevantes similares.

Básicamente, cada transformación consta de dos partes: una descripción del contexto de aplicación, y una regla de reescritura que reemplaza una etiqueta por otra. La plantilla genérica de transformaciones léxicas es la siguiente:

- `x haspref l A`: si los primeros `l` caracteres de la palabra son `x`, se asigna a la palabra desconocida la etiqueta `A`
- `A x fhaspref l B`: si la etiqueta actual de la palabra es `A` y sus primeros `l` caracteres son `x`, se cambia dicha etiqueta por `B`
- `x deletepref l A`: si borrando el prefijo `x` de longitud `l` obtenemos una palabra conocida, se asigna a la palabra desconocida la etiqueta `A`
- `A x fdeletepref l B`: si la etiqueta actual de la palabra es `A` y borrando el prefijo `x` de longitud `l` obtenemos una palabra conocida, se cambia dicha etiqueta por `B`
- `x addpref l A`: si añadiendo el prefijo `x` de longitud `l` obtenemos una palabra conocida, se asigna a la palabra desconocida la etiqueta `A`
- `A x faddpref l B`: si la etiqueta actual de la palabra es `A` y añadiendo el prefijo `x` de longitud `l` obtenemos una palabra conocida, se cambia dicha etiqueta por `B`
- `x hassuf l A`: si los últimos `l` caracteres de la palabra son `x`, se asigna a la palabra desconocida la etiqueta `A`
- `A x fhassuf l B`: si la etiqueta actual de la palabra es `A` y sus últimos `l` caracteres son `x`, se cambia dicha etiqueta por `B`
- `x deletesuf l A`: si borrando el sufijo `x` de longitud `l` obtenemos una palabra conocida, se asigna a la palabra desconocida la etiqueta `A`
- `A x fdeletesuf l B`: si la etiqueta actual de la palabra es `A` y borrando el sufijo `x` de longitud `l` obtenemos una palabra conocida, se cambia dicha etiqueta por `B`

[2]Por ejemplo, en inglés, una palabra terminada en ing podría etiquetarse como un verbo en gerundio.

`x addsuf l A`: si añadiendo el sufijo `x` de longitud `l` obtenemos una palabra conocida, se asigna a la palabra desconocida la etiqueta `A`

`A x faddsuf l B`: si la etiqueta actual de la palabra es `A` y añadiendo el sufijo `x` de longitud `l` obtenemos una palabra conocida, se cambia dicha etiqueta por `B`

`w goodright A`: si la palabra aparece inmediatamente a la derecha de la palabra `w`, se asigna a la palabra desconocida la etiqueta `A`

`A w fgoodright B`: si la etiqueta actual de la palabra es `A` y aparece inmediatamente a la derecha de la palabra `w`, se cambia dicha etiqueta por `B`

`w goodleft A`: si la palabra aparece inmediatamente a la izquierda de la palabra `w`, se asigna a la palabra desconocida la etiqueta `A`

`A w fgoodleft B`: si la etiqueta actual de la palabra es `A` y aparece inmediatamente a la izquierda de la palabra `w`, se cambia dicha etiqueta por `B`

`z char A`: si el caracter `z` aparece en la palabra, se asigna a la palabra desconocida la etiqueta `A`

`A z fchar B`: si la etiqueta actual de la palabra es `A` y el caracter `z` aparece en la palabra, se cambia dicha etiqueta por `B`

donde `A` y `B` son variables sobre el conjunto de todas las etiquetas, `x` es cualquier cadena de caracteres de longitud 1, 2, 3 o 4, `l` es la longitud de dicha cadena, `w` es cualquier palabra, y `z` es cualquier caracter.

Ejemplo 5.1 A continuación se muestran algunas de las reglas de transformación léxicas más comunes que el etiquetador de Brill encontró para el español:

`rse hassuf 3 VOOOfOPE1`: si los últimos 3 caracteres de la palabra son `rse`, se asigna a la palabra desconocida la etiqueta `VOOOfOPE1`, es decir, verbo infinitivo con un pronombre enclítico

`r hassuf 1 VOOOfO`: si el último caracter de la palabra es `r`, se asigna a la palabra desconocida la etiqueta `VOOOfO`, es decir, verbo infinitivo

`VOOOfO or fhassuf 2 Scms`: si la etiqueta actual de la palabra es `VOOOfO`, es decir, verbo infinitivo, y sus últimos 2 caracteres son `or`, se cambia dicha etiqueta por la etiqueta `Scms`, es decir, sustantivo común, masculino, singular

`ría deletesuf 3 VysciO`: si borrando el sufijo `ría` de longitud 3 obtenemos una palabra conocida, se asigna a la palabra desconocida la etiqueta `VysciO`, es decir, verbo, primera y tercera personas del singular, postpretérito de indicativo

`Scfs r faddsuf 1 V3spiO`: si la etiqueta actual de la palabra es `Scfs`, es decir, sustantivo, común, femenino, singular, y añadiendo el sufijo `r` de longitud 1 obtenemos una palabra conocida, se cambia dicha etiqueta por la etiqueta `V3spiO`, es decir, verbo, tercera persona del singular, presente de indicativo

`el goodright Scms`: si la palabra aparece inmediatamente a la derecha de la palabra `el`, se asigna a la palabra desconocida la etiqueta `Scms`, es decir, sustantivo común, masculino, singular

`Scmp las fgoodright Scfp`: si la etiqueta actual de la palabra es `Scmp`, es decir, sustantivo común, masculino, plural, y aparece inmediatamente a la derecha de la palabra `las`, se cambia dicha etiqueta por la etiqueta `Scfp`, es decir, sustantivo común femenino, plural

`%` `goodleft` `Ncyyp`: si la palabra aparece inmediatamente a la izquierda de la palabra `%`, se asigna a la palabra desconocida la etiqueta `Ncyyp`, es decir, numeral cardinal, determinante y no determinante, masculino y femenino, plural

`w` `char` `Ze00`: si el caracter `w` aparece en la palabra, se asigna a la palabra desconocida la etiqueta `Ze00`, es decir, palabra extranjera

Esas reglas fueron generadas por el etiquetador de Brill después de ser entrenado con una porción de texto en español procedente del corpus ITU[3]. □

Por debajo del formato general de las reglas léxicas propuestas por Brill subyace un estudio lingüístico muy importante. Es por ello que esta plantilla genérica de transformaciones léxicas representa una manera muy elegante de integrar el manejo de palabras desconocidas dentro de una herramienta general de etiquetación, y la hace independiente del idioma a tratar. Pero considerando el caso concreto del español, se echan de menos fenómenos muy comunes tales como el sufijo `mente` para formar adverbios a partir de los adjetivos. En el corpus ITU existe un buen número de palabras que presentan dicha característica, pero ésta nunca podrá aparecer reflejada en una regla debido a la limitación de 4 caracteres de longitud en los prefijos y sufijos de las reglas extraídas automáticamente. No obstante, el etiquetador de Brill proporciona al usuario la posibilidad de añadir manualmente nuevas reglas después del entrenamiento.

Otras características relevantes para el tratamiento de palabras desconocidas en modelos de etiquetación basados en reglas fueron estudiadas por Mikheev, quien no sólo amplía el formalismo de reglas léxicas del etiquetador de Brill, sino que también propone su propio algoritmo para la inducción automática de dichas reglas [Mikheev 1997].

5.1.3 El etiquetador contextual

El etiquetador contextual actúa justo después del etiquetador de palabras desconocidas, aplicando en orden una secuencia de reglas contextuales que, al igual que las léxicas, también han sido previamente inferidas de manera automática a partir del corpus de entrenamiento. La plantilla genérica de transformaciones contextuales es la siguiente:

`A B prevtag C`: cambiar la etiqueta `A` por `B` si la palabra anterior aparece etiquetada con la etiqueta `C`

`A B prev1or2tag C`: cambiar la etiqueta `A` por `B` si una de las dos palabras anteriores aparece etiquetada con la etiqueta `C`

`A B prev1or2or3tag C`: cambiar la etiqueta `A` por `B` si una de las tres palabras anteriores aparece etiquetada con la etiqueta `C`

`A B prev2tag C`: cambiar la etiqueta `A` por `B` si la segunda palabra anterior aparece etiquetada con la etiqueta `C`

`A B nexttag C`: cambiar la etiqueta `A` por `B` si la palabra siguiente aparece etiquetada con la etiqueta `C`

`A B next1or2tag C`: cambiar la etiqueta `A` por `B` si una de las dos palabras siguientes aparece etiquetada con la etiqueta `C`

`A B next1or2or3tag C`: cambiar la etiqueta `A` por `B` si una de las tres palabras siguientes aparece etiquetada con la etiqueta `C`

`A B next2tag C`: cambiar la etiqueta `A` por `B` si la segunda palabra siguiente aparece etiquetada con la etiqueta `C`

[3] *International Telecommunications Union CCITT Handbook* (véase la sección 2.1).

- `A B prevbigram C D`: cambiar la etiqueta `A` por `B` si la palabra anterior aparece etiquetada con la etiqueta `C` y la segunda palabra anterior con `D`
- `A B nextbigram C D`: cambiar la etiqueta `A` por `B` si la palabra siguiente aparece etiquetada con la etiqueta `C` y la segunda palabra siguiente con `D`
- `A B surroundtag C D`: cambiar la etiqueta `A` por `B` si la palabra anterior aparece etiquetada con la etiqueta `C` y la siguiente con `D`
- `A B curwd w`: cambiar la etiqueta `A` por `B` si la palabra actual es `w`
- `A B prevwd w`: cambiar la etiqueta `A` por `B` si la palabra anterior es `w`
- `A B prev1or2wd w`: cambiar la etiqueta `A` por `B` si una de las dos palabra anteriores es `w`
- `A B prev2wd w`: cambiar la etiqueta `A` por `B` si la segunda palabra anterior es `w`
- `A B nextwd w`: cambiar la etiqueta `A` por `B` si la palabra siguiente es `w`
- `A B next1or2wd w`: cambiar la etiqueta `A` por la etiqueta `B` si una de las dos palabras siguientes es `w`
- `A B next2wd w`: cambiar la etiqueta `A` por `B` si la segunda palabra siguiente es `w`
- `A B lbigram w x`: cambiar la etiqueta `A` por `B` si las dos palabras anteriores son `w y x`
- `A B rbigram w x`: cambiar la etiqueta `A` por `B` si las dos palabras siguientes son `w y x`
- `A B wdand2bfr x w`: cambiar la etiqueta `A` por `B` si la palabra actual es `w` y la segunda palabra anterior es `x`
- `A B wdand2aft w x`: cambiar la etiqueta `A` por `B` si la palabra actual es `w` y la segunda palabra siguiente es `x`
- `A B wdprevtag C w`: cambiar la etiqueta `A` por `B` si la palabra actual es `w` y la anterior aparece etiquetada con la etiqueta `C`
- `A B wdnexttag w C`: cambiar la etiqueta `A` por `B` si la palabra actual es `w` y la siguiente aparece etiquetada con la etiqueta `C`
- `A B wdand2tagbfr C w`: cambiar la etiqueta `A` por `B` si la palabra actual es `w` y la segunda palabra anterior aparece etiquetada con la etiqueta `C`
- `A B wdand2tagaft w C`: cambiar la etiqueta `A` por `B` si la palabra actual es `w` y la segunda palabra siguiente aparece etiquetada con la etiqueta `C`

donde `A`, `B`, `C` y `D` son variables sobre el conjunto de todas las etiquetas, y `w` y `x` son cualquier palabra. La figura 5.1 resume gráficamente los posibles contextos que se pueden tener en cuenta a la hora de aplicar una transformación. La palabra a etiquetar es siempre la de la posición i, que aparece marcada con un asterisco. Los recuadros indican cuáles son las posiciones relevantes de cada esquema contextual. Si en una posición dada aparece un recuadro normal, el esquema considera la etiqueta de la palabra que está en esa posición. Si aparece un recuadro sombreado, el esquema considera la palabra concreta. Por ejemplo, la regla `A B prevbigram C D` responde al esquema 3, mientras que la regla `A B lbigram w x` utiliza el esquema 15.

Ejemplo 5.2 A continuación se muestran algunas de las reglas de transformación contextuales más comunes que el etiquetador de Brill encontró para el español, también a partir de una porción de texto del corpus ITU:

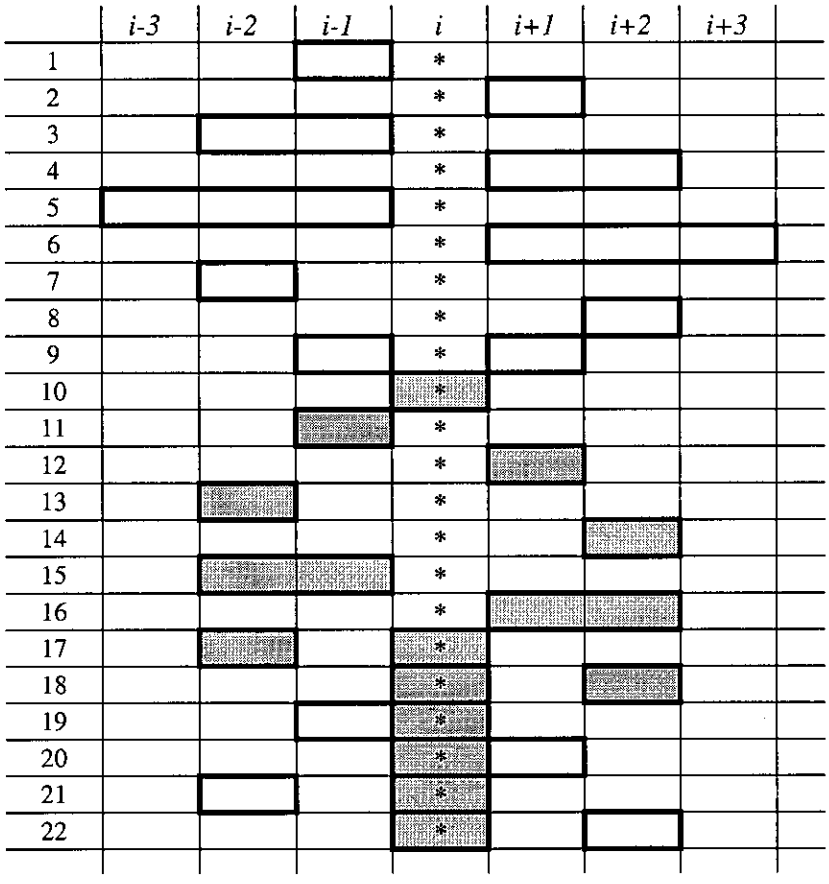


Figura 5.1: Esquemas contextuales de las reglas del etiquetador de Brill

`Afp0 Amp0 prevtag Scmp`: cambiar la etiqueta `Afp0`, es decir, adjetivo, femenino, plural, sin grado, por `Amp0`, es decir, adjetivo, masculino, plural, sin grado, si la palabra anterior aparece etiquetada con la etiqueta `Scmp`, es decir, sustantivo común, masculino, plural

`Scms Ams0 wdprevtag Scms receptor`: cambiar la etiqueta `Scms`, es decir, sustantivo común, masculino, singular, por `Ams0`, es decir, adjetivo, masculino, singular, sin grado, si la palabra actual es `receptor` y la anterior aparece etiquetada con la etiqueta `Scms`

`Scms Ams0 wdand2bfr el transmisor`: cambiar la etiqueta `Scms`, es decir, sustantivo común, masculino, singular, por `Ams0`, es decir, adjetivo, masculino, singular, sin grado, si la palabra actual es `transmisor` y la segunda palabra anterior es `el`

`P Scms nexttag P`: cambiar la etiqueta `P`, es decir, preposición, por `Scms`, es decir, sustantivo común, masculino, singular, si la palabra siguiente aparece etiquetada con la etiqueta `P`

A través de estos ejemplos, se puede apreciar que el formalismo de reglas contextuales del etiquetador de Brill constituye un sencillo pero potente mecanismo, que es capaz de resolver las concordancias de género y número dentro de una misma categoría, las ambigüedades entre distintas categorías, como por ejemplo, adjetivo/sustantivo, y hasta puede evitar que aparezcan dos preposiciones seguidas. □

5.2 Aprendizaje basado en transformaciones y dirigido por el error

El proceso de generación de las reglas, tanto las léxicas en el caso del etiquetador de palabras desconocidas, como las contextuales en el caso del etiquetador contextual, selecciona el mejor conjunto de transformaciones y determina su orden de aplicación. El algoritmo consta de los pasos que se describen a continuación. En primer lugar, se toma una porción de texto no etiquetado, se pasa a través de la fase o fases de etiquetación anteriores, se compara la salida con el texto correctamente etiquetado, y se genera una lista de errores de etiquetación con sus correspondientes contadores. Entonces, para cada error, se determina qué instancia concreta de la plantilla genérica de reglas produce la mayor reducción de errores. Se aplica la regla, se calcula el nuevo conjunto de errores producidos, y se repite el proceso hasta que la reducción de errores cae por debajo de un umbral dado.

La figura 5.2 ilustra gráficamente este procedimiento, que es el que da nombre a la técnica de entrenamiento desarrollada por Brill: aprendizaje basado en transformaciones y dirigido por el error. El usuario puede especificar los umbrales de error antes del entrenamiento, y puede también añadir manualmente nuevas reglas de transformación después del mismo.

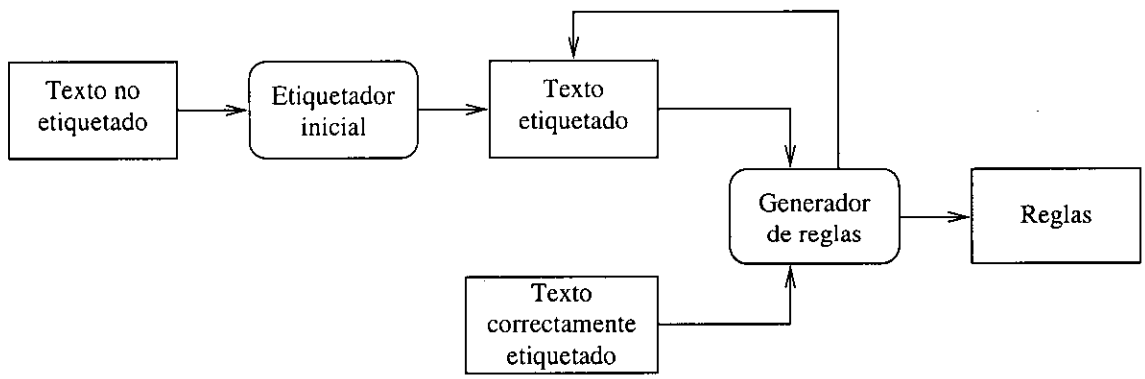


Figura 5.2: Aprendizaje basado en transformaciones y dirigido por el error

A las características anteriormente descritas podríamos añadir también una funcionalidad introducida posteriormente por el propio Brill, la cual permite obtener las k etiquetas más probables de una palabra[4] [Brill 1994], para ciertas aplicaciones en las que es posible relajar la restricción de una sola etiqueta por palabra. Brill implementa esta nueva funcionalidad mediante un sencillo cambio en el formato de las reglas:

> la acción *cambiar la etiqueta* `A` *por la etiqueta* `B` se transforma en *añadir la etiqueta* `A` *a la etiqueta* `B` o *añadir la etiqueta* `A` *a la palabra* `w`.

De esta manera, en lugar de reemplazar etiquetas, las reglas de transformación permiten ahora añadir etiquetas alternativas a una palabra. Sin embargo, el problema es que el etiquetador no nos proporciona información sobre la probabilidad de cada etiqueta. Es decir, si consideramos por ejemplo las dos mejores etiquetas para una palabra dada, la única conclusión que podemos extraer es que la que aparece en primer lugar es más probable que la que aparece en segundo lugar, pero bien podría ocurrir tanto que la primera fuera 100 veces más probable que la segunda, como que ambas fueran igualmente probables. La cuestión es que este tipo de información podría ser crucial para algunas aplicaciones, por ejemplo para la construcción de un análisis sintáctico. Los etiquetadores puramente estocásticos sí son capaces de proporcionar estas cifras y además lo hacen sin ningún esfuerzo computacional extra.

[4] *k-Best tags.*

5.3 Complejidad del etiquetador de Brill

La implementación original de Brill resulta considerablemente más lenta que las basadas en modelos probabilísticos. No sólo el proceso de entrenamiento consume muchísimo tiempo, tal y como veremos en el capítulo 7, sino que el proceso de etiquetación es también inherentemente lento. La principal razón de esta ineficiencia computacional es la potencial interacción entre las reglas, de manera que el algoritmo puede producir cálculos innecesarios.

Ejemplo 5.3 Si suponemos que `VBN` y `VBD` son las etiquetas más probables para las palabras `killed` y `shot`, respectivamente, el etiquetador léxico podría asignar las siguientes etiquetas[5]:

```
(1) Chapman/NP killed/VBN John/NP Lennon/NP
(2) John/NP Lennon/NP was/BEDZ shot/VBD by/BY Chapman/NP
(3) He/PPS witnessed/VBD Lennon/NP killed/VBN by/BY Chapman/NP
```

Dado que el etiquetador léxico no utiliza ninguna información contextual, muchas palabras pueden aparecer etiquetadas incorrectamente. Por ejemplo, en (1) la palabra `killed` aparece etiquetada incorrectamente como verbo en participio pasado, y en (2) `shot` aparece incorrectamente etiquetada como verbo en tiempo pasado.

Una vez obtenida la etiquetación inicial, el etiquetador contextual aplica en orden una secuencia de reglas e intenta remediar los errores cometidos. En un etiquetador contextual podríamos encontrar reglas como las siguientes:

```
VBN VBD prevtag NP
VBD VBN nexttag BY
```

La primera regla dice: *cambiar la etiqueta* `VBN` *por* `VBD` *si la etiqueta previa es* `NP`. La segunda regla dice: *cambiar* `VBD` *por* `VBN` *si la siguiente etiqueta es* `BY`. Una vez que aplicamos la primera regla, la palabra `killed` que aparece en las frases (1) y (3) cambia su etiqueta `VBN` por `VBD`, y obtenemos las siguientes etiquetaciones:

```
(4) Chapman/NP killed/VBD John/NP Lennon/NP
(5) John/NP Lennon/NP was/BEDZ shot/VBD by/BY Chapman/NP
(6) He/PPS witnessed/VBD Lennon/NP killed/VBD by/BY Chapman/NP
```

Una vez que aplicamos la segunda regla, la palabra `shot` de la frase (5) cambia su etiqueta `VBD` por `VBN`, generando la etiquetación (8), y la palabra `killed` de la frase (6) vuelve a cambiar su etiqueta `VBD` otra vez por `VBN`, y obtenemos la etiquetación (9):

```
(7) Chapman/NP killed/VBD John/NP Lennon/NP
(8) John/NP Lennon/NP was/BEDZ shot/VBN by/BY Chapman/NP
(9) He/PPS witnessed/VBD Lennon/NP killed/VBN by/BY Chapman/NP
```

Hemos visto que una regla `nextag` necesita mirar un *token* hacia adelante en la frase antes de poder ser aplicada, y hemos visto también que la aplicación de dos o más reglas puede producir una serie de operaciones que no se traducen en ningún cambio neto. Estos dos fenómenos constituyen la causa del no determinismo local del etiquetador de Brill. □

Este problema fue abordado por Roche y Schabes, quienes propusieron un sistema que codifica las reglas de transformación del etiquetador bajo la forma de un traductor de estado finito determinista [Roche y Schabes 1995]. El algoritmo de construcción de dicho traductor consta de cuatro pasos:

[5] La notación de las etiquetas es una adaptación del juego de etiquetas utilizado en el corpus BROWN [Francis y Kučera 1982]: `VBN` significa verbo en participio pasado, `VBD` es verbo en tiempo pasado, `NP` es sustantivo propio, `BEDZ` es la palabra `was`, `BY` es la palabra `by` y `PPS` es pronombre nominativo singular en tercera persona.

1. En primer lugar, cada transformación se convierte en un traductor de estado finito.

2. El segundo paso consiste en convertir cada traductor a su extensión local. La extensión local t_2 de un traductor t_1 se construye de tal manera que procesar una cadena de entrada a través de t_2 en un solo paso produce el mismo efecto que procesar cada posición de la cadena de entrada a través de t_1. Este paso se ocupa de casos como el siguiente. Supongamos un traductor que implementa la transformación *cambiar* `A` *por* `B` *si una de las dos etiquetas precedentes es* `C`. Este traductor contendrá un arco que transforma el símbolo de entrada `A` en el símbolo de salida `B`, de tal manera que dada la secuencia de entrada `CAA` tenemos que aplicarlo dos veces, en la segunda y en la tercera posiciones, para transformarla correctamente en `CBB`. La extensión local es capaz de realizar dicha conversión en un solo paso.

3. En el tercer paso, se genera un único traductor cuya aplicación tiene el mismo efecto que la aplicación de todos los traductores individuales en secuencia. En general, este traductor único es no determinista. Cuando necesita recordar un evento tal como `C` *apareció en la posición* i, lo hace lanzando dos caminos distintos: uno en el cual se supone que aparecerá una etiqueta que estará afectada por esa `C` precedente, y otro en el que se supone que tal etiqueta no aparecerá.

4. Este tipo de indeterminismo no es eficiente, de ahí que el cuarto paso se ocupe de transformar el traductor no determinista en uno determinista. Esto en general no es posible, ya que los traductores no deterministas pueden recordar eventos de longitud arbitraria y los deterministas no. Sin embargo, Roche y Schabes demuestran que las reglas que aparecen en los etiquetadores basados en transformaciones no generan traductores con esta propiedad. Por tanto, en la práctica siempre es posible transformar un etiquetador basado en transformaciones en un traductor de estado finito determinista.

Por tanto, el algoritmo de Brill podría necesitar RKn pasos elementales para etiquetar una cadena de entrada de n palabras, con R reglas aplicables en un contexto de hasta K *tokens*. Con el traductor de estado finito propuesto por Roche y Schabes, para etiquetar una frase de longitud n palabras, se necesitan sólo n pasos, independientemente del número de reglas y de la longitud del contexto que éstas utilizan. Esto significa que el proceso de etiquetación añade a la lectura del texto de entrada una carga computacional que es despreciable en comparación con tratamientos posteriores tales como los análisis sintáctico y semántico. Con los etiquetadores basados en traductores se pueden llegar a obtener velocidades de etiquetación de varias decenas de miles de palabras por segundo, mientras que con los etiquetadores basados en modelos de Markov esa velocidad puede ser de un orden de magnitud menos. Existen trabajos que estudian la transformación de modelos de Markov ocultos en traductores de estado finito [Kempe 1997], pero en este caso no se puede alcanzar una equivalencia completa ya que los autómatas no pueden simular de una manera exacta los cálculos de punto flotante involucrados en el algoritmo de Viterbi.

5.4 Relación con otros modelos de etiquetación

Se han esbozado ya algunas de las diferencias conceptuales más importantes que existen entre el etiquetador de Brill y los etiquetadores puramente estocásticos. Esta sección completa un poco más el estudio comparativo de los principios de funcionamiento de éstas aproximaciones y de otras relacionadas[6]. Finalmente, se citan otros posibles campos de aplicación en los que el

[6]El estudio analítico completo de los rendimientos de cada paradigma en el proceso de etiquetación será abordado en el capítulo 7.

aprendizaje basado en transformaciones ha funcionado también con éxito.

5.4.1 Árboles de decisión

El aprendizaje basado en transformaciones presenta algunas similitudes con los árboles de decisión[7]. Un árbol de decisión [Schmid 1994, Brown *et al.* 1991] se puede ver como un mecanismo que etiqueta todas las hojas dominadas por un nodo con la etiqueta de la clase mayoritaria de ese nodo. Posteriormente, a medida que descendemos por el árbol, reetiquetamos las hojas de los nodos hijos, si es que difieren de la etiqueta del nodo padre, en función de las respuestas a las cuestiones o decisiones que aparecen en cada nodo. Esta manera de ver los árboles de decisión es la que muestra el parecido con el aprendizaje basado en transformaciones, ya que ambos paradigmas realizan series de reetiquetados trabajando con subconjuntos de datos cada vez más pequeños.

En principio, el aprendizaje basado en transformaciones es más potente que los árboles de decisión [Brill 1995a]. Es decir, existen tareas de clasificación que se pueden resolver con el aprendizaje basado en transformaciones, pero no con los árboles de decisión. Sin embargo, no está muy claro si este tipo de potencia extra se utiliza o no en aplicaciones de procesamiento de lenguaje natural.

La principal diferencia entre estos dos modelos es que los datos de entrenamiento se dividen en cada nodo de un árbol de decisión, y que se aplica una secuencia de *transformaciones* distintas para cada nodo: la secuencia correspondiente a las decisiones del camino que va desde la raíz hasta ese nodo. Con el aprendizaje basado en transformaciones, cada transformación de la lista de transformaciones *aprendidas* se aplica a todo el texto, generando una reescritura cuando el contexto de los datos encaja con el de la regla. Como resultado, si minimizamos en función de los errores de etiquetación cometidos, en lugar de considerar otro tipo de medidas indirectas más comunes en el caso de los árboles de decisión, tales como la entropía, entonces sería relativamente sencillo alcanzar el 100% de precisión en cada nodo hoja. Sin embargo, el rendimiento sobre textos nuevos sería muy pobre debido a que cada nodo hoja estaría formado por un conjunto de propiedades totalmente arbitrarias, que aunque han sido extraídas de los datos de entrenamiento no son completamente generales.

Sorpredentemente, el aprendizaje basado en transformaciones parece ser inmune a este fenómeno [Ramshaw y Marcus 1994]. Esto se puede explicar parcialmente por el hecho de que el entrenamiento siempre se realiza sobre todo el conjunto de datos. Pero el precio que hay que pagar para obtener este tipo de robustez es que el espacio de secuencias de transformaciones debe ser grande. Una implementación *naive* del aprendizaje basado en transformaciones sería por tanto ineficiente. No obstante, existen maneras inteligentes de realizar las búsquedas en ese espacio [Brill 1995b].

5.4.2 Modelos probabilísticos en general

La gran ventaja de la etiquetación basada en transformaciones es que se pueden establecer decisiones sobre un conjunto de propiedades más rico que en el caso de los modelos puramente estocásticos. Por ejemplo, se puede utilizar simultáneamente información de los contextos izquierdo y derecho, y las palabras concretas, no sólo sus etiquetas, pueden influir en la etiquetación de las palabras vecinas.

Otro punto clave es que las reglas de transformación son más fáciles de entender y de modificar que las probabilidades de transición y de generación de palabras en los etiquetadores probabilísticos. Sin embargo, está claro también que es más difícil prever el efecto que puede

[7]También denominados *decision trees*.

llegar a tener la modificación de una regla dentro de una secuencia de aplicación, ya que el comportamiento de cada regla depende de la ejecución de las reglas previas y pueden surgir numerosas y complejas interacciones entre ellas.

Es importante también señalar que la etiquetación basada en transformaciones es claramente un método estadístico. Es decir, aunque no se hace un uso explícito de la teoría probabilística y aunque existe un componente basado en reglas, estas reglas se generan de una manera cuantitativa a partir de contadores calculados directamente sobre los textos. Por tanto, se trata de un método no supervisado[8] y de aplicación completamente automatizada. Desde este punto de vista, la única diferencia es que los cálculos con números se realizan sólo durante el proceso de aprendizaje, el cual además no presenta problemas de sobreentrenamiento, y una vez que el aprendizaje está hecho, el proceso de etiquetación resulta ser puramente simbólico y es por ello que se puede implementar en una estructura computacional muy eficiente.

Por último, dado que el rendimiento de los etiquetadores basados en transformaciones va a resultar muy similar al de los puramente estocásticos, la decisión final entre utilizar unos u otros dependerá casi exclusivamente de en qué tipo de sistema va a estar integrado el etiquetador y para qué tipo de aplicaciones se va a utilizar.

Además de al proceso de etiquetación, el aprendizaje basado en transformaciones se ha aplicado también al análisis sintáctico [Brill 1993a, Brill 1993c], al problema de la ligadura de la frase preposicional [Brill y Resnik 1994] y a la eliminación de ambigüedades semánticas [Dini *et al.* 1998].

[8] En el sentido de que lo que hay presente en los textos de entrenamiento son las etiquetas, pero lo que realmente se extrae de ellos son las reglas.

Capítulo 6

Modelos de máxima entropía

Una de las características de los modelos estocásticos basados en distribuciones de probabilidad es que existe una función de mérito que permite medir y comparar modelos entre sí. Dicha medida es la entropía. Este capítulo presenta otro modelo estadístico para la etiquetación, el sistema JMX [Ratnaparkhi 1996], el cual se entrena también a partir de un corpus previamente etiquetado. El modelo se puede clasificar como un Modelo de Máxima Entropía que utiliza simultáneamente multitud de *rasgos* contextuales para predecir las etiquetas, pero que no establece ninguna hipótesis fija respecto a las distribuciones de probabilidad de los sucesos del texto de entrenamiento. Dicho texto de entrenamiento es utilizado para confeccionar un modelo original, y para medir la diferencia existente entre éste y cualquier otro modelo generado *a posteriori* por el proceso de etiquetación. La idea básica de esta aproximación es entonces la de construir modelos a partir de los nuevos textos de entrada, evaluarlos mediante una función de entropía, y ajustarlos hasta obtener la proximidad máxima respecto al modelo original.

Algunos de los usos previos de este tipo de modelos incluyen la modelización y comprensión del lenguaje [Lau *et al.* 1993, Berger *et al.* 1996], la ligadura de la frase preposicional [Ratnaparkhi *et al.* 1994], y el análisis morfológico de las palabras [Della Prieta *et al.* 1995]. Comenzaremos recordando la noción clásica de entropía, y a continuación describiremos detalladamente las características del modelo, los *rasgos* específicos que utiliza para el proceso de etiquetación, su complejidad temporal, y su flexibilidad respecto a otros etiquetadores.

6.1 Definición clásica de entropía

La *entropía* es una magnitud termodinámica que mide la probabilidad de un estado. El *Principio de la Entropía*, llamado también *Segundo Principio de la Termodinámica* o *Principio de Carnot*, afirma que en toda transformación reversible elemental el cociente dQ/T del calor absorbido por la temperatura absoluta a la que se efectúa la transformación es diferencial exacto de una magnitud de estado S, denominada entropía:

$$\frac{dQ}{T} = dS.$$

En las transformaciones irreversibles se cumple que:

$$\frac{dQ}{T} \leq dS.$$

En resumen, la entropía es una función del estado de un sistema y su valor es independiente del camino seguido para llegar a ese estado. La unidad de entropía es el *clausius*, que equivale a $1\,julio/1^{\circ}\,kelvin$.

Puede demostrarse que la entropía es una magnitud aditiva, por lo que, para obtener el valor de entropía de un sistema, basta sumar las entropías de sus componentes. La dificultad surge para el cálculo de éstas, ya que las fórmulas sólo permiten determinar las diferencias de entropía entre dos estados. Para ello habría que recurrir al *Tercer Principio de la Termodinámica* enunciado por Nernst.

La interpretación microfísica y estadística de la evolución de un sistema lleva a la conclusión de que dicha evolución se efectúa espontáneamente hacia estados más desordenados, que son los más probables, y que precisamente la probabilidad de un estado viene medida logarítmicamente por su entropía.

6.1.1 Medida de la cantidad de información

Supongamos que tenemos un conjunto de q símbolos $S = \{s_1, s_2, \ldots, s_q\}$, y que cada uno tiene una probabilidad de aparición asociada en el conjunto $P = \{p_1, p_2, \ldots, p_q\}$. Cuando observamos uno de esos símbolos, ¿cuánta información obtenemos? Si por ejemplo $p_1 = 1$ y, por supuesto, todas las demás probabilidades son cero, entonces no hay ninguna sorpresa, ninguna información, ya que se conoce de antemano qué símbolo se va a observar. Sin embargo, si las probabilidades son muy diferentes, cuando llega un símbolo de probabilidad baja se produce más sorpresa y se obtiene más información que cuando llega un símbolo de probabilidad alta. Por tanto, la información está de alguna manera relacionada de forma inversa con la probabilidad de aparición de los símbolos.

Consideremos la construcción de una función $I(p)$ que mida la cantidad de información, sorpresa o incertidumbre en la ocurrencia de un suceso de probabilidad p, y que tenga las siguientes propiedades:

1. Debe ser una medida real no negativa: $I(p) \geq 0$.
2. Para sucesos independientes, debe verificar la propiedad aditiva: $I(p_1\, p_2) = I(p_1) + I(p_2)$.
3. Debe ser una función continua.

La segunda de estas condiciones es conocida como la *Ecuación Funcional de Cauchy* para nuestra función. Examinenos más detenidamente esta propiedad. Si p_1 y p_2 son el mismo número p, aunque no necesariamente correspondientes al mismo suceso, entonces

$$I(p^2) = I(p) + I(p) = 2\, I(p).$$

Si $p_1 = p$ y $p_2 = p^2$, entonces tenemos

$$I(p^3) = I(p) + I(p^2) = 3\, I(p).$$

Y en general tenemos que

$$I(p^n) = n\, I(p).$$

Utilizando fracciones como exponentes, tenemos que si $p^n = y$, entonces $p = y^{1/n}$ y por tanto

$$I(y) = n\, I(y^{1/n}).$$

O bien, en general,

$$I(y^{m/n}) = \frac{m}{n}\, I(y).$$

Por tanto, para los exponentes que son números racionales, la función $I(p)$ obedece a la misma fórmula que la función logaritmo.

Además, por la tercera restricción, la función es continua, lo que permite extender este razonamiento a todos los números $0 \leq p \leq 1$, racionales o irracionales. Por tanto, tenemos que

$$I(p) = k \log(p)$$

para alguna constante k y alguna base del sistema de logaritmos. El logaritmo de 1 es 0 (no hay información, no hay sorpresa), y los logaritmos de los números menores que 1 son negativos. Por tanto, resulta natural coger $k = -1$, ya que así podemos cuantificar la información en términos positivos. Finalmente, tenemos que

$$I(p) = -\log(p) = \log\left(\frac{1}{p}\right)$$

para alguna base del sistema de logaritmos.

Ejemplo 6.1 Para aclarar el significado de la propiedad aditiva descrita en la condición 2, consideremos simultánea e independientemente el lanzamiento de una moneda y de un dado. Los experimentos no tienen influencia uno en el otro. La moneda tiene 2 resultados equiprobables y el dado 6. Por tanto, el espacio de resultados del experimento global tiene 12 resultados igualmente probables.

Si m es el resultado de la moneda y d es el resultado del dado, entonces la cantidad de información obtenida será

$$I(m\,d) = \log\left(\frac{1}{12}\right),$$

o bien

$$I(m) + I(d) = \log\left(\frac{1}{2}\right) + \log\left(\frac{1}{6}\right).$$

Ambas cantidades son la misma. □

Las palabras incertidumbre, sorpresa e información están relacionadas. Antes del suceso (del experimento, de la observación de un símbolo, etc.) hay una cierta cantidad de incertidumbre; cuando ocurre el suceso hay una cierta cantidad de sorpresa; y después del suceso hay una ganancia de información. Todas esas cantidades son la misma.

¿Qué base del sistema de logaritmos debemos usar? Esto es simplemente una cuestión de convenio, ya que un conjunto de logaritmos es proporcional a cualquier otro. Esto se sigue directamente de la relación

$$\log_a(x) = \frac{\log_b(x)}{\log_b(a)} = \log_a(b)\,\log_b(x).$$

Si se utiliza la base 2, la unidad de información se llama *bit*. Si se usa base e, la unidad de información es el *nat*. Y si la base utilizada es 10, la unidad de información se denomina *Hartley*, ya que fue Hartley el primero en proponer el uso de logaritmos para medir la información. Por tanto, hay que señalar que la palabra *bit* se usa de dos maneras diferentes: para referirnos a un número en base 2, o como unidad de información. Estas dos cosas no son lo mismo, así que debemos tener cuidado a la hora de utilizar esta palabra. Cuando queramos referirnos al segundo concepto sin llevar a confusión, debemos decir *bit de información*.

6.1.2 Entropía de una variable aleatoria

Si cuando observamos el símbolo s_i obtenemos $I(p_i)$ unidades de información, ¿cuánta información obtenemos de media? La respuesta es que dado que p_i es la probabilidad de observar dicha información, entonces por cada símbolo s_i obtenemos una media de

$$p_i\, I(p_i) = p_i \log\left(\frac{1}{p_i}\right)$$

unidades de información. De esto se sigue que la información media que podemos obtener con el conjunto de símbolos entero es

$$H(S) = \sum_{i=1}^{q} p_i \log\left(\frac{1}{p_i}\right) = -\sum_{i=1}^{q} p_i \log(p_i).$$

Ésta es la entropía de un conjunto S, con q símbolos s_i de probabilidades de observación respectivas p_i. Por tanto, para nosotros la entropía es simplemente función de una distribución de probabilidad. Es decir, para cada distribución de probabilidad $P = \{p_1, p_2, \ldots, p_q\}$ siempre existe un número asociado que se denomina *entropía*.

Esto es análogo a la idea usual de la media de una distribución. La media es también un número que de algún modo resume la distribución. La entropía es la media ponderada de los logaritmos de los recíprocos de las probabilidades de la distribución y, en nuestro caso, es una medida de la información media del conjunto de símbolos S.

Para formalizar la noción de entropía, podemos pensar en la incertidumbre asociada a una variable aleatoria discreta que mide los posibles valores de un suceso (el lanzamiento de una moneda o un dado, la observación de un símbolo, etc.). En la siguiente definición se expresa esta idea y se generaliza para cualquier base r.

Definición 6.1 Sea X una variable aleatoria discreta con q resultados posibles, donde p_i es la probabilidad de obtener el resultado x_i. Definimos la *entropía* de X como la cantidad

$$H_r(X) = \sum_{i=1}^{q} p_i \log_r\left(\frac{1}{p_i}\right) = -\sum_{i=1}^{q} p_i \log_r(p_i)$$

donde, por convenio, si $p_i = 0$ entonces

$$p_i \log_r\left(\frac{1}{p_i}\right) = -p_i \log_r(p_i) = 0.$$

La entropía de X es conocida también como la *incertidumbre* de X. □

La entropía satisface las siguientes propiedades, que como se puede observar, están íntimamente relacionadas con la noción intuitiva de incertidumbre:

1. Se cumple que $H(X) \geq 0$. $H(X) = 0$ si y sólo si una de las probabilidades es 1 y las demás son 0.

2. La máxima entropía se obtiene cuando todas las probabilidades son iguales, esto es, cuando todos los resultados posibles de la variable aleatoria son igualmente probables.

3. Si dos variables aleatorias X e Y poseen n y m resultados equiprobables respectivamente, con $n < m$, entonces $H(X) < H(Y)$. Es decir, la entropía máxima crece cuando el número de resultados posibles crece.

Ejemplo 6.2 Supongamos que la variable aleatoria discreta X tiene 4 resultados posibles, con la siguiente distribución de probabilidad: $\{p_1 = 1/8, p_2 = 1/4, p_3 = 1/8, p_4 = 1/2\}$. Entonces,

$$H_2(X) = \frac{1}{8}\log_2(8) + \frac{1}{4}\log_2(4) + \frac{1}{8}\log_2(8) + \frac{1}{2}\log_2(2) = \frac{7}{4}.$$

Si consideramos una variable aleatoria Y con los mismos resultados posibles, pero siendo éstos equiprobables, la entropía quedaría maximizada para el caso de 4 resultados y tendríamos que $H_2(Y) = 2$. □

Nuestro interés inmediato se centra en el manejo de textos en lenguaje natural. Las variables aleatorias a tratar aquí serían frases, es decir, secuencias de palabras de la forma $W_{1,n} = w_1, w_2, \ldots, w_n$. Debido a esta descomposición del mensaje observado, aparece el concepto de *entropía por palabra*:

$$\frac{1}{n}H(W_{1,n}) = -\frac{1}{n}\sum_{w_{1,n}} p(w_{1,n})\,\log(p(w_{1,n})).$$

Con esto, se podría definir la *entropía de un lenguaje* L como la entropía por palabra de la frase más larga, es decir, aquella cuya longitud tiende a infinito:

$$H(L) = -\lim_{n\to\infty}\left(\frac{1}{n}\sum_{w_{1,n}} p(w_{1,n})\,\log(p(w_{1,n}))\right).$$

Sin embargo, para el caso particular de la etiquetación, es necesario considerar y formalizar el contexto en el que aparecen las palabras. La incorporación de dicho contexto al concepto de entropía visto anteriormente es la idea general que se desarrolla a lo largo de las siguientes secciones.

6.2 El modelo probabilístico

El modelo probabilístico propuesto por Ratnaparkhi en la implementación del etiquetador JMX [Ratnaparkhi 1996] se define sobre $\mathcal{H} \times \mathcal{T}$, donde $\mathcal{H}$ es el conjunto de posibles contextos palabra-etiqueta o sucesos *históricos*, y $\mathcal{T}$ es el conjunto de etiquetas permitidas. La probabilidad en el modelo para un suceso histórico h junto con una etiqueta t se define como:

$$p(h,t) = \pi\,\mu \prod_{j=1}^{k} \alpha_j^{f_j(h,t)} \tag{6.1}$$

donde: π es una constante de normalización; $\{\mu, \alpha_1, \alpha_2, \ldots, \alpha_k\}$ son los parámetros positivos del modelo; y $\{f_1, f_2, \ldots, f_k\}$ se conocen como *funciones de rasgo*[1], y son de la forma $f_j(h,t) \in \{0,1\}$. Cada parámetro α_j se corresponde con una función de rasgo f_j.

Dado un conjunto de datos de entrenamiento formado por una secuencia de palabras $\{w_1, w_2, \ldots, w_n\}$ y una secuencia de etiquetas $\{t_1, t_2, \ldots, t_n\}$, se define h_i como la historia disponible cuando se predice t_i. Entonces, los parámetros $\{\mu, \alpha_1, \alpha_2, \ldots, \alpha_k\}$ se eligen de manera que se maximice la verosimilitud de los datos de entrenamiento mediante la distribución p, es decir, de manera que se maximice la cantidad:

$$L(p) = \prod_{i=1}^{n} p(h_i, t_i) = \prod_{i=1}^{n} \pi\,\mu \prod_{j=1}^{k} \alpha_j^{f_j(h_i,t_i)}.$$

[1] O también *features*.

Este modelo se puede interpretar también bajo el formalismo de Máxima Entropía, en el cual el objetivo es maximizar la entropía de una distribución sujeta a ciertas restricciones. Aquí, la entropía de la distribución p se define como:

$$H(p) = -\sum_{h\in\mathcal{H},t\in\mathcal{T}} p(h,t)\log(p(h,t))$$

y las restricciones vienen dadas por:

$$E(f_j) = \tilde{E}(f_j), \qquad 1 \leq j \leq k, \tag{6.2}$$

donde la esperanza de las funciones rasgo del modelo es

$$E(f_j) = \sum_{h\in\mathcal{H},t\in\mathcal{T}} p(h,t)f_j(h,t)$$

y la esperanza de las funciones rasgo observadas es

$$\tilde{E}(f_j) = \sum_{i=1}^{n} \tilde{p}(h_i,t_i)f_j(h_i,t_i)$$

y donde $\tilde{p}(h_i,t_i)$ denota la probabilidad de (h_i,t_i) observada en los datos de entrenamiento. Por tanto, las restricciones obligan a que el modelo intente igualar las esperanzas de sus funciones rasgo con las esperanzas observadas durante el entrenamiento. En la práctica, $\mathcal{H}$ es muy grande y las esperanzas $E(f_j)$ del modelo no se pueden calcular directamente. Debido a esto, se utiliza la siguiente aproximación [Lau *et al.* 1993]:

$$E(f_j) \approx \sum_{i=1}^{n} \tilde{p}(h_i)p(t_i|h_i)f_j(h_i,t_i)$$

donde $\tilde{p}(h_i)$ es la probabilidad de la historia h_i observada en el entrenamiento.

Los parámetros del modelo para la distribución p se obtienen mediante la aplicación de un procedimiento denominado *escalado iterativo generalizado*[2] [Darroch y Ratcliff 1972]. Este método garantiza que si la distribución inicial p sigue la forma de la ecuación (6.1) y satisface las k restricciones de (6.2), la maximización de la verosimilitud $L(p)$ para la obtención de la nueva distribución se produce únicamente sobre distribuciones de la forma (6.1) y, de igual manera, la maximización de la entropía $H(p)$ se produce sólo sobre distribuciones que satifacen (6.2).

6.3 Rasgos específicos para la etiquetación

La probabilidad conjunta de una historia h y una etiqueta t se determina mediante aquellos parámetros cuyas funciones rasgo correspondientes estén activadas, es decir, mediante aquellos α_j tales que $f_j(h,t) = 1$. Dados (h,t), la decisión de si una función rasgo se activa o no puede tomarse a partir de las características de cualquier palabra o etiqueta de la historia h. Por tanto, una función rasgo puede y debe codificar cualquier fuente de información que ayude a predecir la etiqueta t. Esas fuentes de información están constituidas por características contextuales tales como, por ejemplo, la terminación de la palabra actual, o la identidad de las dos etiquetas previas. Los contextos específicos de palabra y etiqueta disponibles para una función rasgo vienen dados por la definición de una historia h_i:

$$h_i = \{t_{i-2}, t_{i-1}, w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}\}.$$

[2] *Generalized iterative scaling.*

Ejemplo 6.3 De esta manera, podemos tener, por ejemplo:

$$f_j(h_i, t_i) = \begin{cases} 1 & \text{si } sufijo(w_i) = \texttt{ing} \text{ y } t_i = \texttt{VBG}, \\ 0 & \text{en caso contrario.} \end{cases}$$

Si la función rasgo anterior está presente en el conjunto de funciones rasgo del modelo, su parámetro correspondiente participa en el cálculo de la probabilidad conjunta $p(h_i, t_i)$ cuando la palabra w_i termina en ing y cuando t_i = VBG[3]. □

Por tanto, un parámetro α_j juega efectivamente el papel de un *peso* para una determinada predicción contextual (en el ejemplo anterior, el sufijo ing), la cual se utilizará o no en el cálculo de la probabilidad de observación de una determinada etiqueta (en el ejemplo anterior, VBG).

El modelo genera el conjunto de funciones rasgo comparando cada par (h_i, t_i) del corpus de entrenamiento con la plantilla de funciones rasgo de la tabla 6.1. Dada una historia h_i, una función rasgo siempre efectúa una pregunta de respuesta sí o no sobre h_i, y obliga a que t_i sea una etiqueta concreta. La instanciación de las variables X, Y y T de la tabla 6.1 se obtiene automáticamente a partir de los datos de entrenamiento.

Condición	Funciones rasgo	
w_i no es rara	$w_i = X$	y $t_i = T$
w_i es rara	X es prefijo de w_i, $\lvert X\rvert \leq 4$	y $t_i = T$
	X es sufijo de w_i, $\lvert X\rvert \leq 4$	y $t_i = T$
	w_i contiene un número	y $t_i = T$
	w_i contiene una letra mayúscula	y $t_i = T$
	w_i contiene un guión	y $t_i = T$
$\forall w_i$	$t_{i-1} = X$	y $t_i = T$
	$t_{i-2}t_{i-1} = XY$	y $t_i = T$
	$w_{i-1} = X$	y $t_i = T$
	$w_{i-2} = X$	y $t_i = T$
	$w_{i+1} = X$	y $t_i = T$
	$w_{i+2} = X$	y $t_i = T$

Tabla 6.1: Plantilla genérica de funciones rasgo aplicables a una historia h_i

La generación de funciones rasgo para la etiquetación de las palabras desconocidas se basa en la hipótesis de que las palabras *raras* del corpus de entrenamiento y las palabras desconocidas que aparecerán en los *corpora* de aplicación son similares, en lo que se refiere a cómo su morfología ayuda a predecir la etiqueta correcta. Por tanto, las funciones rasgo específicas para palabras raras de la tabla 6.1 se aplican tanto a las palabras raras como a las palabras desconocidas. En este modelo, se considera *rara* toda palabra que aparece menos de 5 veces en el texto de entrenamiento.

Ejemplo 6.4 A continuación mostramos un ejemplo de una frase de entrenamiento en español procedente del corpus ITU[4]:

la	información	sobre	servicios	suplementarios	es	muy	completa
Dfs	Scfs	P	Scmp	AmpO	V3spiO	Wm	AfsO
1	2	3	4	5	6	7	8

[3] VBG es la etiqueta correspondiente a verbo en gerundio en el juego de etiquetas del *Wall Street Journal corpus* del proyecto *Penn Treebank* [Marcus *et al.* 1994].

[4] *International Telecommunications Union CCITT Handbook* (véase la sección 2.1).

La primera línea está constituida por las palabras de dicha frase. La segunda representa las etiquetas de dichas palabras: `Dfs` (artículo, femenino, singular); `Scfs` (sustantivo común, femenino, singular); `P` (preposición); `Scmp` (sustantivo común, masculino, plural); `AmpO` (adjetivo, masculino, plural, sin grado); `V3spiO` (verbo, tercera persona del singular, presente de indicativo); `Wm` (adverbio modificador); y `AfsO` (adjetivo, femenino, singular, sin grado). La tercera línea indica la posición de cada palabra dentro de la frase.

Las funciones rasgo generadas durante el análisis de (h_3, t_3), donde la palabra actual es `sobre`, son las siguientes:

w_i = `sobre`	y t_i = `P`
w_{i-1} = `información`	y t_i = `P`
w_{i-2} = `la`	y t_i = `P`
w_{i+1} = `servicios`	y t_i = `P`
w_{i+2} = `suplementarios`	y t_i = `P`
t_{i-1} = `Scfs`	y t_i = `P`
$t_{i-2}t_{i-1}$ = `Dfs Scfs`	y t_i = `P`

Las funciones rasgo generadas durante el análisis de (h_5, t_5), donde la palabra actual es `suplementarios`, son las siguientes:

w_{i-1} = `servicios`	y t_i = `AmpO`
w_{i-2} = `sobre`	y t_i = `AmpO`
w_{i+1} = `es`	y t_i = `AmpO`
w_{i+2} = `muy`	y t_i = `AmpO`
t_{i-1} = `Scmp`	y t_i = `AmpO`
$t_{i-2}t_{i-1}$ = `P Scmp`	y t_i = `AmpO`
$prefijo(w_i)$ = `s`	y t_i = `AmpO`
$prefijo(w_i)$ = `su`	y t_i = `AmpO`
$prefijo(w_i)$ = `sup`	y t_i = `AmpO`
$prefijo(w_i)$ = `supl`	y t_i = `AmpO`
$sufijo(w_i)$ = `s`	y t_i = `AmpO`
$sufijo(w_i)$ = `os`	y t_i = `AmpO`
$sufijo(w_i)$ = `ios`	y t_i = `AmpO`
$sufijo(w_i)$ = `rios`	y t_i = `AmpO`

En la porción de texto de entrenamiento a la cual pertenece la frase considerada, la palabra `suplementarios` aparece 3 veces, y por tanto efectivamente se clasifica como palabra rara. □

El comportamiento de una función rasgo que aparece muy pocas veces después de analizar los datos de entrenamiento es a menudo difícil de predecir, ya que dicha función será producto de fenómenos muy dispersos y no existirá una base estadística bien fundamentada que nos pueda garantizar su correcta aplicación. Por tanto, el modelo utiliza la heurística de que cualquier función rasgo que aparezca menos de 10 veces no es fiable y se ignora[5]. Como hemos visto en capítulos anteriores, existen técnicas para el tratamiento de datos dispersos mucho más rigurosas que la simple definición de umbrales de confianza[6], pero este modelo de etiquetación todavía no ha sido investigado en conjunto con dichas técnicas.

[5]Excepto aquellas funciones rasgo que consideran sólo la palabra actual, es decir, funciones rasgo de la forma w_i = <palabra> y t_i = <etiqueta>.

[6]Que además, tanto en el caso de 5 para las palabras raras, como en el caso de 10 para funciones rasgo no fiables, han sido elegidos por los diseñadores de este método de manera totalmente empírica después de un análisis previo de varios *corpora* de entrenamiento y de aplicación.

El paradigma de etiquetación de Máxima Entropía permite utilizar arbitrariamente el contexto, de tal manera que el usuario puede añadir funciones rasgo especiales después del entrenamiento para intentar corregir los errores residuales del modelo base. Típicamente, dichas funciones rasgo especiales se definen para palabras concretas, y podrían incluso no obedecer a la plantilla genérica mostrada en la tabla 6.1. Una función rasgo especial para una palabra determinada se construye combinando algunas de las funciones rasgo del modelo base con preguntas sobre la propia palabra. Por ejemplo, las funciones rasgo que hacen preguntas acerca de las etiquetas previas, o acerca de las palabras anterior y posterior, pueden ahora preguntar también sobre la identidad de la palabra actual.

Ejemplo 6.5 Una función rasgo especial para la etiquetación de la palabra `sobre`:

$$f_j(h_i, t_i) = \begin{cases} 1 & \text{si } w_i = \texttt{sobre} \text{ y } t_{i-2}t_{i-1} = \texttt{Dfs Scfs} \text{ y } t_i = \texttt{P}, \\ 0 & \text{en caso contrario.} \end{cases}$$

Esta función de rasgo es efectivamente una combinación de la identidad de la palabra con la última de las funciones vistas para `sobre` en el ejemplo 6.4. □

Dado que este tipo de funciones rasgo especiales para una determinada palabra estarán relacionadas con fenómenos muy dispersos, el usuario debe introducirlas con precaución. Es conveniente verificar que dichas funciones son suficientemente consistentes con el resto de los fenómenos del texto de entrenamiento relativos a esa misma palabra. De no ser así, se corre el riesgo de introducir ruido en el modelo, lo cual podría traducirse en una degradación del rendimiento del sistema a la hora de etiquetar nuevos textos.

6.4 Complejidad del etiquetador JMX

El corpus de aplicación se etiqueta frase a frase. Este procedimiento requiere una búsqueda previa de las secuencias de etiquetas candidatas para una frase. La secuencia que obtenga mayor probabilidad es la que el sistema ofrece como respuesta.

El algoritmo de búsqueda utiliza la probabilidad condicionada

$$p(t|h) = \frac{p(h,t)}{\sum_{t' \in \mathcal{T}} p(h,t')}$$

y mantiene, cada vez que opera sobre una nueva palabra, las N secuencias de etiquetas candidatas que tengan mayor probabilidad hasta ese punto de la frase[7]. Dada una frase $\{w_1, w_2, \ldots, w_n\}$, la probabilidad condicionada de una secuencia de etiquetas candidatas $\{t_1, t_2, \ldots, t_n\}$ se calcula como

$$P(t_1, t_2, \ldots, t_n | w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n} p(t_i|h_i).$$

El procedimiento de búsqueda puede manejar un diccionario que contiene, para cada palabra, la lista de posibles etiquetas observadas en el corpus de entrenamiento. El uso del diccionario es opcional. Si no se utiliza, el procedimiento de búsqueda considera todo el juego de etiquetas para cada palabra. Si se utiliza, el procedimiento considera, para las palabras conocidas, sólo

[7]Los experimentos realizados por Ratnaparkhi sugieren que un buen valor para N es 5; valores superiores a éste no producen grandes mejoras en el rendimiento y afectan gravemente a la velocidad de etiquetación.

las etiquetas presentes en el diccionario y, para las palabras desconocidas, igualmente todo el juego de etiquetas[8].

Los detalles de la búsqueda que se ejecuta durante el proceso de etiquetación se describen en el siguiente algoritmo.

Algoritmo 6.1 Sea $W = \{w_1, w_2, \ldots, w_n\}$ una frase a etiquetar, y sea s_{ij} la j-ésima secuencia de etiquetas de más alta probabilidad hasta la palabra w_i, incluida. El algoritmo de búsqueda es como sigue:

1. Generar las etiquetas para w_1, buscar las N más probables, ordenarlas de mayor a menor probabilidad, y asignar s_{1j}, $1 \leq j \leq N$, de acuerdo con dicho orden.
2. Inicializar $i = 2$.
   (a) Inicializar $j = 1$.
   (b) Generar las etiquetas para w_i, tomando $s_{(i-1)j}$ como contexto de etiquetas previo, y añadir cada etiqueta a $s_{(i-1)j}$ para formar una nueva secuencia.
   (c) Asignar $j = j + 1$. Si $j \leq N$, volver a 2(b).
3. Buscar las N secuencias de etiquetas de mayor probabilidad generadas por el bucle anterior, ordenarlas de mayor a menor probabilidad, y asignar s_{ij}, $1 \leq j \leq N$, de acuerdo con dicho orden.
4. Asignar $i = i + 1$. Si $i \leq n$, volver a 2(a).
5. Devolver s_{n1}, la secuencia de probabilidad más alta.

Éste es, por tanto, el algoritmo que realmente realiza la etiquetación. □

La complejidad del algoritmo de estimación de parámetros es $\mathcal{O}(T\,E\,A)$, donde T es el número de posibles etiquetas, E es el tamaño del corpus de entrenamiento, y A es el número medio de funciones rasgo que están activas para un suceso dado (h, t). La complejidad temporal del procedimiento de búsqueda que se ejecuta durante la etiquetación, sobre una frase de longitud n, es $\mathcal{O}(n\,T\,A\,N)$, donde T y A son los valores que acabamos de definir, y N ha sido también definido anteriormente como el número de secuencias de etiquetas candidatas de más alta probabilidad que se mantienen cada vez que se estudia una nueva palabra de la frase.

6.5 Relación con otros modelos de etiquetación

El sistema de etiquetación basado en modelos de Máxima Entropía que hemos presentado en este capítulo combina las ventajas de otros métodos, tales como los modelos de Markov ocultos [Weischedel *et al.* 1993, Merialdo 1994], los árboles de decisión [Magerman 1995], o el aprendizaje basado en transformaciones [Brill 1994].

El etiquetador JMX, al igual que el etiquetador de Brill y que los árboles de decisión, utiliza un amplio conjunto de propiedades léxicas y sintácticas del lenguaje capturadas directamente a partir de los textos de entrenamiento. La representación del contexto que rodea a una palabra es similar a la del etiquetador de Brill[9], y la representación del conocimiento necesario

[8] El diccionario incrementa el rendimiento de manera casi insignificante, pero se recomienda su uso porque reduce el número de hipótesis a manejar y por tanto aumenta considerablemente la velocidad del etiquetador.

[9] El etiquetador de Brill utiliza una *ventana* de ±3 palabras respecto a la palabra actual, mientras que el etiquetador JMX utiliza una ventana de ±2.

para el tratamiento de palabras desconocidas es de hecho un subconjunto de las reglas de transformaciones léxicas que utiliza Brill[10].

Sin embargo, el etiquetador de Brill no proporciona información sobre la probabilidad de cada etiqueta. El etiquetador JMX, al igual que los árboles de decisión y que los modelos de Markov, genera las distribuciones de probabilidad de etiquetas para cada palabra, y por tanto podría ser utilizado como un componente probabilístico de un sistema más grande.

Por otra parte, el texto de entrenamiento se divide recursivamente en cada uno de los nodos de un árbol de decisión. Esta fragmentación obliga a utilizar complejos algoritmos de suavización para mitigar el efecto de los datos dispersos. En el etiquetador JMX este problema no se presenta, y un sencillo contador utilizado como umbral de suavización es suficiente para alcanzar prácticamente el mismo nivel de precisión.

En resumen, el modelo de Máxima Entropía es una técnica de gran flexibilidad para la representación de conocimiento lingüístico, que tiene al menos una ventaja sobre cada uno de los sistemas de etiquetación citados anteriormente: maneja un abanico de propiedades del lenguaje mayor que los modelos de Markov, el proceso de entrenamiento requiere técnicas menos sofisticadas que las utilizadas en los árboles de decisión, y, al contrario que el aprendizaje basado en transformaciones, se puede integrar dentro de un marco probabilístico.

Sin embargo, Ratnaparkhi realizó los experimentos de su etiquetador sobre el *Wall Street Journal corpus* del proyecto *Penn Treebank* [Marcus *et al.* 1994], y encontró que el rendimiento de todos estos modelos de etiquetación sobre ese mismo corpus era muy similar. Este fenómeno puede deberse a dos razones:

1. Podría ser que todas estas técnicas estén representando el conocimiento lingüístico mediante un conjunto de predictores que no resulta ser el adecuado para corregir los errores residuales de las etiquetas incorrectas.

2. Pero Ratnaparkhi sugiere que ninguna técnica lingüística basada en corpus conseguirá un rendimiento superior al 96,5% de precisión sobre el *Wall Street Journal corpus*, debido al gran número de inconsistencias presentes en dicho corpus[11].

En el siguiente capítulo veremos cuál es el comportamiento exacto del etiquetador JMX sobre otros textos, tanto en español como en inglés.

[10] Brill utiliza el añadido/borrado de prefijos/sufijos, mientras que JMX no.

[11] En el desarrollo de este corpus participó un equipo de anotadores muy heterogéneo, lo cual muchas veces ha provocado que una misma palabra aparezca etiquetada de diferentes maneras en diferentes puntos del corpus aun cuando está realizando la misma función.

Capítulo 7

Evaluación de los sistemas de etiquetación

Una vez que se han descrito los principios de funcionamiento de los paradigmas de etiquetación considerados en este trabajo, discutimos ahora los dos aspectos fundamentales que es necesario tener en cuenta para realizar un estudio comparativo de los mismos. En primer lugar, presentamos la estrategia que se ha seguido a la hora de realizar los experimentos, tanto con el corpus ITU[1] como con el corpus SUSANNE[2]. Básicamente, los pasos son: construir un corpus de entrenamiento formado por frases tomadas aleatoriamente del corpus de referencia, entrenar las herramientas que se van a evaluar, reetiquetar con ellas la porción restante de corpus, y comparar con el original. Obviamente, esto debe hacerse varias veces y con diferentes tamaños de corpus de entrenamiento. Las siguientes secciones describen este proceso más detalladamente.

Posteriormente, especificamos qué es exactamente lo que vamos a medir. Es decir, cuáles son las fórmulas que calculan los índices de rendimiento que nos permiten decidir si un etiquetador es mejor o peor que otro. Para cada etiquetador, se ha construido un conjunto de tablas con las cifras concretas de aciertos y fallos a partir de las cuales se obtiene dicho rendimiento. Finalmente, se presenta un conjunto de figuras que resumen gráficamente el estudio comparativo de todos los sistemas de etiquetación evaluados.

7.1 Estrategia para la realización de los experimentos

Antes de comenzar con la descripción detallada de los experimentos realizados, enumeramos los cuatro etiquetadores que hemos evaluado, citando explícitamente el paradigma de etiquetación correspondiente y también las características particulares de cada uno de ellos:

1. TNT, abreviatura de *Trigrams'and'Tags* (trigramas y etiquetas), es un eficiente etiquetador que puede ser utilizado sobre cualquier lengua y sobre cualquier juego de etiquetas [Brants 1996]. El módulo de generación de parámetros precisa un corpus de entrenamiento etiquetado. TNT es una implementación del algoritmo de Viterbi para modelos de Markov ocultos de segundo orden. El método utilizado para la suavización es la interpolación lineal, donde los pesos respectivos se determinan mediante el proceso de interpolación de borrado descrito en la sección 4.4.3.2. Las palabras desconocidas se manejan mediante inferencias sucesivas de prueba de sufijos [Samuelsson 1993].

   Este sistema no permite la integración de diccionarios externos, o al menos no directamente. Las últimas versiones de este etiquetador incorporan opciones para generar,

[1] *International Telecommunications Union CCITT Handbook* (véase la sección 2.1).

[2] El banco de árboles utilizado como corpus de referencia para el idioma inglés (véase la sección 2.3).

a partir del corpus de entrenamiento, la información relativa a los unigramas, bigramas y trigramas, y la información relativa al lexicón, de manera independiente [Brants 2000]. Por tanto, dados un corpus y un diccionario externo, se podría construir un modelo generando los n-gramas a partir del corpus, y el lexicón a partir de la unión de ambos recursos. En cualquier caso, este procedimiento no sería más que una simulación del método de integración de diccionarios *Adding One* descrito en la sección 4.4.3.4.

2. BRILL es el representante del paradigma de etiquetación de aprendizaje basado en transformaciones y dirigido por el error [Brill 1993b]. Los principios de funcionamiento de este etiquetador han sido descritos ya en el capítulo 5. Este sistema requiere también un corpus previamente etiquetado para realizar el entrenamiento. El etiquetador se puede aplicar a cualquier idioma y a cualquier juego de etiquetas, y permite la integración de diccionarios externos.

   En el caso del sistema BRILL, es necesario recordar que la primera etiqueta de cada palabra que figura en el lexicón se considera como la etiqueta más probable, y que el resto de etiquetas de esa misma palabra no tienen porqué aparecer en ningún orden determinado. En cualquier caso, el sistema BRILL no hace uso de probabilidades explícitas para las palabras. Por tanto, podrían existir maneras muy diversas de mezclar o integrar juntos el diccionario de entrenamiento y otro diccionario externo. Pero una vez más, para no apartar demasiado las condiciones de nuestros experimentos de las que se producen realmente en la práctica, hemos considerado siempre el lexicón obtenido en el entrenamiento como lexicón primario[3].

3. JMX es un etiquetador basado en modelos de máxima entropía [Ratnaparkhi 1996], que ya fue presentado en el capítulo 6. El proceso de entrenamiento se efectúa también a partir de un corpus previamente etiquetado. El etiquetador se puede aplicar a cualquier idioma y a cualquier juego de etiquetas, pero no permite la integración de diccionarios externos.

4. GALENA, abreviatura de Generador de Analizadores para Lenguajes Naturales [Vilares *et al.* 1995], es un sistema cuyo módulo de etiquetación está basado también en un modelo de Markov oculto de orden 2. Como en las herramientas anteriores, los parámetros de funcionamiento del modelo se obtienen a partir de un corpus previamente etiquetado. Para la suavización de parámetros se puede elegir entre el método de interpolación lineal y el método de *marcha atrás* o *back-off* que hemos visto en la sección 4.5.5. El etiquetador puede ser utilizado sobre cualquier conjunto de etiquetas y sobre cualquier idioma.

   La integración de diccionarios externos se puede realizar mediante el método *Adding One* o mediante el método de Good-Turing descrito en la sección 4.4.3.4. Además, es posible realizar la especificación previa de un conjunto de pares sufijo-etiqueta. Esta especificación, que debe realizarse a través de un estudio previo de la morfología del idioma de aplicación, constituye un apoyo fundamental para el tratamiento de palabras desconocidas, especialmente cuando el corpus de entrenamiento es de tamaño reducido [Sacristán y Graña 1999]. En el presente trabajo, nuestro amplio conocimiento de los fenómenos flexivos y derivativos del español nos ha permitido construir esa especificación, y obtener por tanto un etiquetador especializado para nuestro idioma.

   Al igual que en el etiquetador TNT, la adivinación de palabras desconocidas se realiza mediante inferencias sucesivas de prueba de sufijos [Samuelsson 1993].

[3] En los experimentos con el sistema BRILL, las combinaciones de diccionarios han sido realizadas con el programa Perl `combine-lexicons.prl` que aparece en el directorio `Utilities` de la distribución de dicho etiquetador.

A pesar de que, en general, los etiquetadores pueden ser aplicables a cualquier conjunto de etiquetas, en la mayoría de ellos no es posible especificarlo de manera explícita. De hecho, esto es lo que ocurre con los sistemas TnT, Brill y JMX. Son ellos mismos los que deducen automáticamente el juego de etiquetas como el conjunto de todas las etiquetas diferentes que aparecen en el corpus de entrenamiento.

Los problemas con las etiquetas pueden surgir precisamente al integrar diccionarios externos. Si el diccionario externo contiene algún par palabra-etiqueta, tal que dicha etiqueta no ha aparecido en el corpus de entrenamiento, el etiquetador puede verse obligado a tratar una etiqueta *desconocida*, es decir, una etiqueta de la cual no ha podido deducir nada previamente. Bajo estas circunstancias, sólo el sistema TnT es capaz de adaptarse al problema, aunque el mecanismo de adaptación utilizado no está excesivamente fundamentado[4]. Por tanto, lo más conveniente en este tipo de sistemas es proporcionar un texto de entrenamiento que contenga cada una de las etiquetas del conjunto al menos una vez, pero esto queda bajo la responsabilidad del usuario.

El sistema Galena permite la declaración previa del juego de etiquetas, comprueba que tanto las etiquetas del corpus como las del posible diccionario externo pertenezcan a dicho conjunto, y exige que el corpus contenga cada una de las etiquetas del conjunto al menos una vez, lo cual evita el problema de las etiquetas desconocidas.

7.1.1 Experimentos sobre el corpus ITU

En el caso del corpus ITU, los diferentes textos de entrenamiento que se utilizan en los experimentos se construyen cogiendo aleatoriamente frases del corpus. Esto no garantiza que cada uno de ellos contenga apariciones de todas y cada una de las etiquetas. Por lo tanto, hemos diseñado un componente de texto que estará siempre presente en el entrenamiento, que no forma parte del corpus ITU, y cuya única característica relevante es precisamente el incluir cada etiqueta al menos una vez. A este nuevo texto lo hemos denominado *Corpus de Entrenamiento Cero* o TC0[5].

En general, es necesario considerar varios aspectos a la hora de pensar sobre este componente fijo. Con respecto a su tamaño, cuanto más pequeño mejor, para no alterar demasiado las propiedades que caracterizan al corpus de referencia original. Y con respecto a su construcción, y precisamente para obtener un tamaño lo más reducido posible, hemos preferido crear manualmente un texto nuevo, en lugar de extraer frases del corpus original hasta que todo el juego de etiquetas estuviera cubierto. Es por esta razón por la que decimos que el *Corpus de Entrenamiento Cero* no forma parte del corpus ITU. Por supuesto, dependiendo del cardinal del conjunto de etiquetas, esta elaboración manual del nuevo texto podría consumir gran cantidad de tiempo. En este caso, hemos invertido tres días de trabajo para cubrir las 373 etiquetas que aparecen en el corpus.

A continuación, describimos exactamente las características concretas de nuestro *Corpus de Entrenamiento Cero*:

- El corpus tiene 1.228 palabras. El tamaño del fichero es de 19.921 caracteres. La sintaxis de cada línea es de la forma

```
palabra/etiqueta/lema  palabra/etiqueta/lema ... palabra/etiqueta/lema
```

[4] Dicho mecanismo consiste en asignar a cada una de las etiquetas desconocidas la probabilidad $\frac{1}{N+1}$, $\frac{1}{N+2}$, y así sucesivamente, donde N es el tamaño del corpus de entrenamiento, de tal manera que se establece una dependencia intrínseca en cuanto al orden de aparición de dichas etiquetas, y sus probabilidades, una vez asignadas, no se recalculan, ignorando por tanto un importante dato de realimentación como puede ser la frecuencia del conflicto.

[5] *Training Corpus Zero.*

Cada una de las líneas del fichero es una frase. El corpus tiene 54 frases, es decir, un número medio de 22 palabras por frase.

- El corpus contiene 332 formas verbales, donde 52 formas están en voz pasiva, otras 52 son formas verbales compuestas, y otras 11 son formas verbales con un pronombre enclítico.
- Las características del lexicón o diccionario constituido por todas las formas que aparecen en el corpus son las siguientes:

  544 formas con 1 etiqueta,
  39 formas con 2 etiquetas y
  3 formas con 3 etiquetas.

  Esto es, 586 formas diferentes, con 631 etiquetas posibles, y correspondientes a 386 lemas diferentes. Si calculamos el porcentaje de formas ambiguas y el número medio de etiquetas por forma, obtenemos:

$$\%\ \text{formas ambiguas} = \frac{\#\ \text{formas ambiguas}}{\#\ \text{formas}} \times 100 = \frac{586 - 544}{586} \times 100 = 7,16\ \%$$

$$\#\ \text{medio de etiquetas por forma} = \frac{\#\ \text{etiquetas}}{\#\ \text{formas}} = \frac{631}{586} = 1,08\ \text{etiquetas por forma.}$$

- Mucho más interesante es calcular las mismas características directamente con todas las palabras del corpus, y obtenemos las siguientes cifras:

  987 formas con 1 etiqueta,
  232 formas con 2 etiquetas y
  9 formas con 3 etiquetas.

  Esto es, 1.228 palabras, con 1.478 etiquetas posibles. Si calculamos de nuevo el porcentaje de palabras ambiguas y el número medio de etiquetas por palabra, obtenemos:

$$\%\ \text{palabras ambiguas} = \frac{\#\ \text{palabras ambiguas}}{\#\ \text{palabras}} \times 100 = \frac{1.228 - 987}{1.228} \times 100 = 19,63\ \%$$

$$\#\ \text{medio de etiquetas por palabra} = \frac{\#\ \text{etiquetas}}{\#\ \text{palabras}} = \frac{1.478}{1.228} = 1,20\ \text{etiquetas por palabra.}$$

A la porción de texto de entrenamiento que sí se selecciona aleatoriamente la hemos denominado *Corpus de Entrenamiento Uno* o TC1[6]. Este corpus es, por tanto, variable en cada experimento, pero constituye el componente principal ya que es el de mayor tamaño. Hemos elegido cinco tamaños diferentes: 1.000, 2.000, 3.000, 4.000 y 5.000 frases extraídas aleatoriamente del corpus ITU. Más concretamente, este procedimiento extrae primero un bloque de 5.000 frases de una vez, y después construye 5 textos de entrenamiento diferentes: el primero con las 1.000 primeras frases del bloque, el segundo con las 2.000 primeras frases, el tercero con las 3.000 primeras, y así sucesivamente hasta cubrir las 5.000 frases del bloque. De esta manera, podemos realizar 5 tests diferentes, y la porción de texto a reetiquetar y comparar es siempre la misma en cada uno de ellos.

[6] *Training Corpus One.*

Este proceso de extracción ha sido realizado 3 veces, dando lugar a tres bancos de experimentos, cada uno de 5 tests diferentes. Es decir, un total de 15 tests. Y finalmente, hemos realizado un único entrenamiento con un tamaño de 10.000 frases, lo cual produce un conjunto final de 16 tests. A este último banco lo hemos llamado *Banco Especial*, y a cada uno de los tests como se indica en la tabla 7.1.

Tamaño de TC1 / Banco	Banco 1	Banco 2	Banco 3	Banco Especial
1.000 frases	test11	test21	test31	-
2.000 frases	test12	test22	test32	-
3.000 frases	test13	test23	test33	-
4.000 frases	test14	test24	test34	-
5.000 frases	test15	test25	test35	-
10.000 frases	-	-	-	testSP

Tabla 7.1: Denominación de los tests realizados sobre el corpus ITU

La figura 7.1 es un esquema del proceso de construcción de los *corpora* de entrenamiento a partir de la extracción aleatoria de frases del corpus ITU. Las porciones de texto no seleccionadas en este proceso, R1, R2, R3 y RSP, constituyen los *corpora* de referencia de cada uno de los bancos de experimentos.

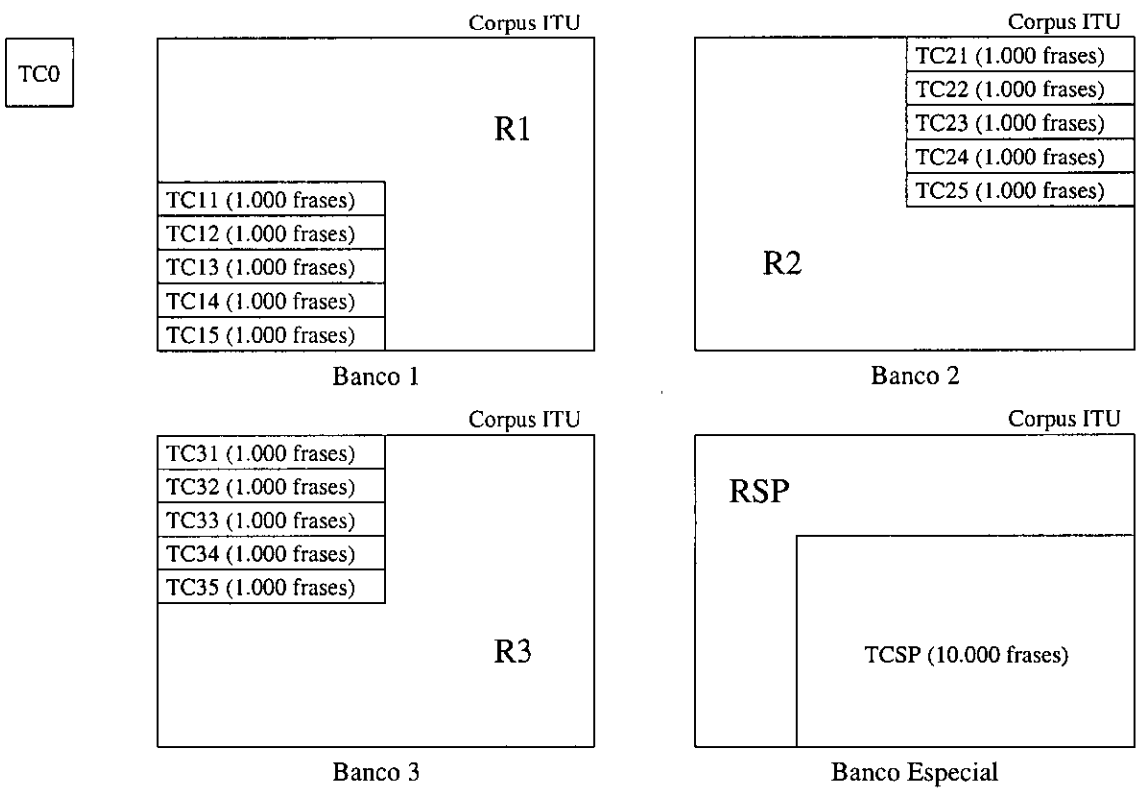


Figura 7.1: *Corpus de Entrenamiento Cero* y extracción aleatoria de frases para la construcción del *Corpus de Entrenamiento Uno* de cada test sobre el corpus ITU

La figura 7.2 detalla las fases de entrenamiento, reetiquetado y evaluación que se realizan en cada test. En esta figura, r1, r2, r3 y rSP representan las versiones sin etiquetar de los *corpora* de referencia R1, R2, R3 y RSP, respectivamente. En cada test, se reetiqueta uno de esos *corpora* sin etiquetar, obteniéndose una nueva versión etiquetada que es la que se compara con la referencia para calcular los índices de rendimiento S1 y S2. Estos índices se describen más adelante en la sección 7.2.1.

Como ya hemos visto, todos los etiquetadores evaluados generan su propio lexicón a partir del corpus de entrenamiento que se les proporciona en cada experimento. A este lexicón lo denominaremos *Lexicón de Entrenamiento*. Sin embargo, hemos visto también que algunos de estos etiquetadores, en concreto el sistema BRILL y el sistema GALENA, son capaces de integrar un diccionario construido de manera totalmente externa al proceso de entrenamiento. Uno de los principales objetivos del presente estudio ha sido el de añadir un lexicón español extra de tamaño medio, el lexicón del sistema GALENA descrito en la sección 2.2.2, con el fin de comprobar en qué medida el uso de un diccionario externo puede ayudar o no a mejorar el rendimiento de estos etiquetadores.

Por otra parte, también hemos querido comprobar el comportamiento de los etiquetadores cuando utilizan un lexicón que contiene todas las palabras que aparecen en el corpus ITU, es decir, cuando no tienen que enfrentarse a palabras desconocidas. Hemos generado este lexicón y lo hemos denominado *Lexicón* ITU. El objetivo aquí es calcular el nivel máximo de eliminación pura de ambigüedades, es decir, la cota máxima de éxito que se puede esperar en el proceso de etiquetación para un determinado tamaño de corpus de entrenamiento.

7.1.2 Experimentos sobre el corpus SUSANNE

El tamaño del corpus SUSANNE es bastante inferior al del corpus ITU. Debido a esto, realizaremos un único experimento en el cual el criterio de partición, como ya hemos visto en la sección 2.3.2, no ha sido la extracción aleatoria de frases. En este caso, el corpus de entrenamiento está formado por las frases sin trazas (4.292 frases, 77.275 palabras), y el corpus de referencia por las frases con trazas (2.188 frases, 60.759 palabras).

El juego de etiquetas del corpus SUSANNE contiene 425 etiquetas, tal y como puede verse en la sección A.4, de las cuales sólo 395 aparecen en el corpus de entrenamiento. Por tanto, aquí también sería necesario construir un *Corpus de Entrenamiento Cero* que cubriera al menos las 30 etiquetas que no aparecen. Por supuesto, este nuevo corpus debería estar constituido por frases en inglés, y además, para poder comparar los rendimientos de los etiquetadores tradicionales con los de las nuevas técnicas de etiquetación basadas en sintaxis que veremos en los próximos capítulos, sería necesario también que dichas frases pudieran ser generadas por la gramática SUSANNE y habría que proporcionar los árboles de análisis sintáctico de las mismas.

Sin embargo, esas 30 etiquetas que no están presentes en el corpus de entrenamiento involucran a un conjunto de palabras que se sitúan en sólo 149 frases de las 2.188 que contiene el corpus de referencia. Así pues, la opción elegida ha sido no considerar estas frases, y eliminarlas del corpus de referencia y del resto de experimentos. El tamaño final del corpus de referencia es, por tanto, de 2.039 frases y 51.426 palabras.

7.2 Metodología para la evaluación del rendimiento

Esta sección está especialmente dedicada a explicar qué es exactamente lo que vamos a medir durante el proceso de evaluación de los distintos etiquetadores. La metodología que proponemos para dicha evaluación [Graña y Rajman 1999] se describe a continuación.

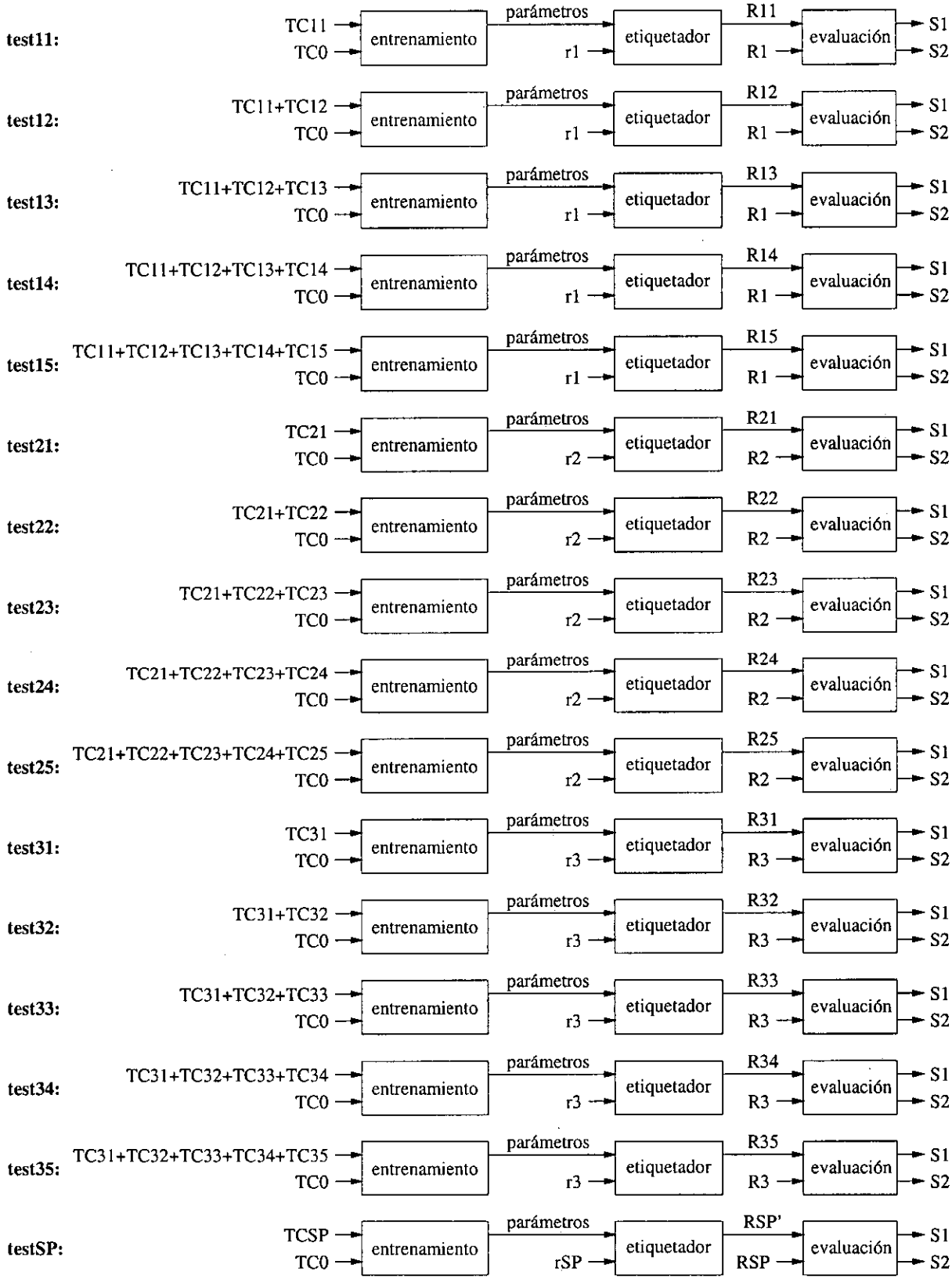


Figura 7.2: Detalle de las fases de entrenamiento, reetiquetado y evaluación de cada test sobre el corpus ITU

En general, dada una palabra de cualquier texto que se vaya a etiquetar, se consideran tres posibles situaciones para ella:

1. Que la palabra no haya aparecido previamente en el corpus de entrenamiento, lo cual implica que no aparecerá tampoco en el lexicón de entrenamiento. Esta palabra se denomina palabra desconocida o forma fuera de vocabulario y nos referiremos a ella como una OOVF[7].

2. Que la palabra haya aparecido previamente en el corpus de entrenamiento con una única etiqueta. Se trata entonces de una forma no ambigua, al menos en lo que se refiere al lexicón de entrenamiento. Nos referiremos a ella como una NAF[8].

3. Que la palabra haya aparecido previamente en el corpus de entrenamiento con varias etiquetas, es decir, es una forma ambigua. Nos referiremos a ella como una AF[9].

Por lo tanto, en cada test, cuando se compara el texto reetiquetado con el corpus de referencia, calculamos los siguientes contadores:

- OOVF+, que indica el número de palabras desconocidas o fuera de vocabulario que han sido etiquetadas correctamente.

- OOVF-, que indica el número de palabras desconocidas o fuera de vocabulario que han sido etiquetadas erróneamente.

- NAF+, que indica el número de palabras no ambiguas que han sido etiquetadas correctamente.

- NAF-, que indica el número de palabras no ambiguas que han sido etiquetadas erróneamente.

- AF+, que indica el número de palabras ambiguas que han sido etiquetadas correctamente.

- AF-, que indica el número de palabras ambiguas que han sido etiquetadas erróneamente.

Posteriormente, a partir de los valores anteriores, calculamos también los siguientes contadores globales:

- #OOVF = (OOVF+) + (OOVF-), que indica el número total de palabras desconocidas o fuera de vocabulario.

- #NAF = (NAF+) + (NAF-), que indica el número total de palabras no ambiguas.

- #AF = (AF+) + (AF-), que indica el número total de palabras ambiguas.

- #F = (#OOVF) + (#NAF) + (#AF), que indica el número total de palabras, tanto en el corpus reetiquetado, como en el corpus de referencia.

Por último, para tener una idea definitiva del nivel de acierto obtenido en cada test, calculamos los índices de rendimiento S1 y S2.

[7] *Out Of Vocabulary Form.*
[8] *Non-Ambiguous Form.*
[9] *Ambiguous Form.*

7.2.1 Índices S1 (precisión) y S2 (decisión)

S1, el *índice de precisión*, determina el nivel de acierto global. Por tanto, en general, se calcula como:

$$S1\ (\text{precisión}) = \frac{ACIERTO}{ACIERTO + ERROR} \times 100$$

y, en nuestro caso, como:

$$S1\ (\text{precisión}) = \frac{(OOVF+) + (NAF+) + (AF+)}{\#F} \times 100.$$

S2, el *índice de decisión*[10], es un concepto que pertenece a la terminología utilizada en extracción de información, y mide el nivel de acierto cuando el sistema tiene una posibilidad real de acertar, es decir, cuando realmente tiene que tomar decisiones. En general se calcula como:

$$S2\ (\text{decisión}) = \frac{ACIERTO + ERROR}{ACIERTO + ERROR + SILENCIO} \times 100$$

y, en nuestro caso, como:

$$S2\ (\text{decisión}) = \frac{(OOVF+) + (AF+)}{\#F - \#NAF} \times 100.$$

Un buen sistema de etiquetación debería presentar valores cercanos a 100%, no sólo en el índice de acierto global, sino en ambos.

Si el sistema de etiquetación que se está evaluando no presenta valores altos en S1 y S2, pero es capaz de generar no sólo la etiqueta más probable de cada palabra, sino varias, entonces podemos considerar otro grupo de palabras constituido por aquellas formas ambiguas que han sido etiquetadas ambiguamente. Nos referiremos a cada una de estas palabras como una AFAT[11]. A partir de esta nueva categoría se pueden calcular tres contadores más:

- AFAT+, que indica el número de palabras ambiguas que han sido etiquetadas ambiguamente, donde alguna de las etiquetas elegidas es la correcta.
- AFAT-, que indica el número de palabras ambiguas que han sido etiquetadas ambiguamente, donde ninguna de las etiquetas elegidas es la correcta.
- #AFAT = (AFAT+) + (AFAT-), que indica el número total de palabras ambiguas que han sido etiquetadas ambiguamente.

Entonces, recalculamos los índices de rendimiento simplemente reemplazando los valores AF por los valores AFAT:

$$S1'\ (\text{precisión}) = \frac{(OOVF+) + (NAF+) + (AFAT+)}{\#F} \times 100$$

y

$$S2'\ (\text{decisión}) = \frac{(OOVF+) + (AFAT+)}{\#F - \#NAF} \times 100.$$

[10] También denominado índice de *recall*.

[11] *Ambiguous Form Ambiguously Tagged.*

Si nuestra intención inicial era desechar el sistema, pero los valores de S1' y S2' pasan ahora a ser aceptables, entonces nos podemos plantear el abordar tareas tales como reprogramar alguna de las partes del etiquetador para mejorar el rendimiento inicial, o filtrar la salida a través de otro sistema.

De todas las herramientas que han sido evaluadas en este trabajo, sólo el sistema BRILL presenta la propiedad de generación de las n etiquetas más probables para cada palabra[12]. Sin embargo, cuando se activa esta funcionalidad, hemos observado que este sistema precisa una gran cantidad de tiempo de entrenamiento extra, y no produce ninguna mejora de rendimiento significativa. Por lo tanto, el cálculo de los índices S1' y S2' no se ha realizado para ninguno de los etiquetadores.

7.2.2 Tablas y gráficos de rendimiento

Las siguientes páginas muestran las tablas de contadores, índices de rendimiento, tiempos de entrenamiento y de etiquetación, tamaños de los *corpora* de entrenamiento y de referencia, etc., para cada uno de los 17 tests (16 sobre el corpus ITU y 1 sobre el corpus SUSANNE) y para cada uno de los 4 sistemas de etiquetación que han sido evaluados en este trabajo: TNT, BRILL, JMX y GALENA.

En el caso de los experimentos sobre el corpus ITU con los sistemas BRILL y GALENA, las tablas aparece seguidas de un par de figuras que representan la evolución de los valores de S1 y S2 con cada uno de los diferentes lexicones utilizados, proporcionando una idea gráfica de la mejora producida al integrar un diccionario externo. Finalmente, la tabla 7.18 muestra la media aritmética de los valores de los tres bancos precedentes, junto con los propios valores del Banco Especial, para los cuatro sistemas cuando compiten en condiciones estándar, es decir, con el lexicón que ellos mismos han generado durante el entrenamiento. Las figuras 7.7 y 7.8 concluyen el estudio comparativo mostrando gráficamente los valores de S1 y S2 de esta tabla resumen. La sección se cierra con las tablas correspondientes a los experimentos realizados sobre el corpus SUSANNE.

Sin embargo, antes de presentar todas estas tablas y figuras, es necesario discutir algunos aspectos de interés para el correcto entendimiento de las mismas:

- La primera cuestión es: ¿qué representan los porcentajes que aparecen al lado del valor de cada contador? Estos porcentajes representan la distribución de aciertos y fallos dentro de cada una de las tres categorías de contadores: OOVF, NAF y AF. Por ejemplo, en el test11, con el sistema TNT, OOVF+ es 26.220 y OOVF- es 12.330. Es decir, OOVF+ es el 68,01% de todas las palabras desconocidas en este test, y OOVF- es el restante 31,99%.

  Por lo tanto, con sólo echar un vistazo a estos porcentajes a lo largo de la tabla, podemos obtener una idea sobre el rendimiento local en cada test, en lo que se refiere a las palabras desconocidas, a las palabras no ambiguas, o a las palabras ambiguas, frente al rendimiento global representado por S1 y S2. En función de estas distribuciones, puede ocurrir que un sistema sea globalmente peor que otro, pero localmente mejor en lo relativo, por ejemplo, a las palabras ambiguas.

- Otra importante cuestión, en las tablas de aquellos sistemas que pueden integrar diccionarios externos, BRILL y GALENA, es la siguiente: ¿por qué NAF- es 0 cuando no hay palabras fuera de vocabulario? O lo que es lo mismo: ¿por qué NAF- es diferente de 0 en otras condiciones? Una vez más, consideremos el test11. Para cada forma no ambigua `f` del lexicón de entrenamiento, $\texttt{f/t}_\texttt{i}$ aparece en `TC0`, o en `TC11`, o en ambos, pero siempre con la etiqueta $\texttt{t}_\texttt{i}$. Reetiquetamos R1 y obtenemos R11. Todas las formas `f` de

[12] También denominada propiedad *n-best tags*.

R11 serán etiquetadas con t_i, pero f/t_j, donde $t_i \neq t_j$, podría aparecer en R1. Esto quiere decir que f es una forma no ambigua con respecto al lexicón de entrenamiento, pero realmente es una forma potencialmente ambigua, que se puede etiquetar mal, y que por tanto constituye una NAF-. Sin embargo, cuando trabajamos con todo el diccionario del corpus ITU, f está presente en el lexicón al menos con las dos etiquetas t_i y t_j. Cuando reetiquetamos R1 y obtenemos R11, obtenemos un acierto o un fallo, pero que se contabilizan o bien en AF+ o bien en AF-, ya que ahora f es una forma ambigua, y esto es precisamente lo que provoca que NAF- sea igual a 0.

De todo esto se deduce que tanto el sistema BRILL como el sistema GALENA utilizan la información contextual para seleccionar las etiquetas sólo de las palabras ambiguas y desconocidas, y suponen que no existe ambigüedad potencial en las palabras no ambiguas. Ésta podría ser una suposición muy fuerte, pero si nos fijamos en los porcentajes de acierto locales para las palabras no ambiguas vemos que son realmente excelentes. Por tanto, introducir un cambio en esta hipótesis podría producir un descenso drástico del rendimiento.

Un comportamiento similar al de los sistemas BRILL y GALENA, en lo relativo al tratamiento de palabras no ambiguas, se observa también en el sistema TNT, pero no en el sistema JMX. Este sistema es el único que explota la ambigüedad potencial de las formas *a priori* no ambiguas, pero tal y como veremos en la siguiente sección, al realizar el análisis de resultados, ésta no parece ser una buena hipótesis de funcionamiento.

- Cuando se integra un diccionario externo, resulta muy interesante conocer cuántas palabras NAF- pasan a ser AF+, y cuántas pasan a ser AF-, o incluso qué otros cambios aparecen en el resto de contadores. El apéndice C muestra los diagramas y tablas de transición de palabras al integrar diccionarios externos en los sistemas BRILL y GALENA. Como veremos más adelante, dichas tablas constituyen también una interesante fuente de información para determinar la ganancia de rendimiento producida por los diccionarios externos.

- Por último, es necesario indicar que los tiempos de entrenamiento (T.Entr.) y los tiempos de etiquetación (T.Etiq.) que aparecen en las tablas se muestran en formato `hh:mm:ss` (horas, minutos y segundos). Sin embargo, hemos trabajado con diferentes máquinas, con el fin de poder diseñar una distribución adecuada de la carga de procesamiento implicada por el gran número de experimentos realizados, y no sobrecargar así un único sistema. Por tanto, para que esta información pueda ser calibrada adecuadamente, citamos a continuación las principales características de las máquinas utilizadas:

  - `asterix.dc.fi.udc.es` es una máquina con un procesador Pentium II a 300 MHz bajo sistema operativo Linux.
  - `covas.dc.fi.udc.es` es una máquina *Sun Ultra-Enterprise* con dos procesadores Sparc bajo sistema operativo Solaris.
  - `ds.cesga.es` es una máquina *Sun Ultra-Enterprise* con un procesador Sparc bajo sistema operativo Solaris.
  - `liasun13.epfl.ch` es una máquina *Sun Ultra-2* con un procesador Sparc bajo sistema operativo Solaris.

  Es importante decir que el hecho de haber trabajado con diferentes plataformas no nos ha impedido extraer conclusiones representativas de las restricciones temporales que presentan cada uno de los etiquetadores evaluados, tal y como veremos al final del capítulo.

Datos Generales del Experimento					
Test	test11	test12	test13	test14	test15
Máquina	asterix	asterix	asterix	asterix	asterix
T.Entr.	00:00:01	00:00:01	00:00:02	00:00:03	00:00:03
Tamaño del Corpus de Entrenamiento					
Frases	1.054	2.054	3.054	4.054	5.054
Palabras	32.774	68.322	96.775	126.302	162.521
Tamaño del Corpus de Referencia					
Frases	9.919	9.919	9.919	9.919	9.919
Palabras	324.780	324.780	324.780	324.780	324.780
Lexicón: Entrenamiento					
T.Etiq.	00:00:18	00:00:14	00:00:13	00:00:12	00:00:11
OOVF+	26.220 68,01%	16.149 70,97%	12.334 76,87%	10.205 80,78%	8.411 82,05%
OOVF-	12.330 31,99%	6.605 29,03%	3.710 23,13%	2.428 19,22%	1.840 17,95%
NAF+	172.907 98,58%	181.111 98,84%	185.763 99,13%	187.425 99,32%	173.623 99,47%
NAF-	2.498 1,42%	2.120 1,16%	1.627 0,87%	1.280 0,68%	929 0,53%
AF+	105.920 95,57%	114.117 96,06%	116.702 96,17%	118.926 96,34%	135.391 96,72%
AF-	4.905 4,43%	4.678 3,94%	4.644 3,83%	4.516 3,66%	4.586 3,23%
S1	93,924	95,873	96,926	97,467	97,735
S2	88,461	92,028	93,919	94,896	95,722

Tabla 7.2: Corpus: ITU - Sistema: TnT - Banco de Experimentos: 1

Datos Generales del Experimento					
Test	test21	test22	test23	test24	test25
Máquina	asterix	asterix	asterix	asterix	asterix
T.Entr.	00:00:01	00:00:01	00:00:02	00:00:03	00:00:03
Tamaño del Corpus de Entrenamiento					
Frases	1.054	2.054	3.054	4.054	5.054
Palabras	33.192	71.367	100.218	129.764	166.064
Tamaño del Corpus de Referencia					
Frases	9.919	9.919	9.919	9.919	9.919
Palabras	321.237	321.237	321.237	321.237	321.237
Lexicón: Entrenamiento					
T.Etiq.	00:00:16	00:00:11	00:00:11	00:00:10	00:00:09
OOVF+	26.339 69,27%	15.673 71,13%	11.915 76,25%	10.022 80,40%	8.438 82,70%
OOVF-	11.681 30,73%	6.359 28,87%	3.711 23,75%	2.443 19,60%	1.764 17,30%
NAF+	169.564 98,61%	178.135 98,74%	182.740 99,09%	184.074 99,30%	183.386 99,47%
NAF-	2.380 1,39%	2.269 1,26%	1.666 0,91%	1.282 0,70%	961 0,53%
AF+	106.510 95,71%	114.187 96,11%	116.662 96,25%	119.057 96,46%	121.769 96,45%
AF-	4.763 4,29%	4.614 3,89%	4.543 3,75%	4.359 3,54%	4.469 3,55%
S1	94,140	95,877	96,911	97,483	97,760
S2	88,985	92,208	93,967	94,994	95,430

Tabla 7.3: Corpus: ITU - Sistema: TnT - Banco de Experimentos: 2

Datos Generales del Experimento										
Test	test31		test32		test33		test34		test35	
Máquina	asterix		asterix		asterix		asterix		asterix	
T.Entr.	00:00:01		00:00:01		00:00:02		00:00:03		00:00:03	
Tamaño del Corpus de Entrenamiento										
Frases	1.054		2.054		3.054		4.054		5.054	
Palabras	33.284		69.764		97.557		127.230		162.819	
Tamaño del Corpus de Referencia										
Frases	9.919		9.919		9.919		9.919		9.919	
Palabras	324.482		324.482		324.482		324.482		324.482	
Lexicón: Entrenamiento										
T.Etiq.	00:00:17		00:00:13		00:00:12		00:00:11		00:00:10	
OOVF+	26.486	68,01%	15.958	70,13%	12.441	76,11%	10.180	80,10%	8.610	82,13%
OOVF-	12.454	31,99%	6.796	29,87%	3.905	23,89%	2.528	19,90%	1.873	17,87%
NAF+	173.063	98,74%	178.555	98,78%	182.177	99,10%	184.065	99,23%	182.840	99,42%
NAF-	2.194	1,26%	2.193	1,22%	1.652	0,90%	1.412	0,77%	1.061	0,58%
AF+	105.600	95,75%	116.382	96,19%	119.784	96,36%	121.962	96,56%	125.550	96,50%
AF-	4.685	4,25%	4.598	3,81%	4.523	3,64%	4.335	3,44%	4.548	3,50%
S1	94,041		95,812		96,893		97,449		97,694	
S2	88,514		92,072		94,007		95,062		95,432	

Tabla 7.4: Corpus: ITU - Sistema: TnT - Banco de Experimentos: 3

Datos Generales		
Test	testSP	
Máquina	asterix	
T.Entr.	00:00:05	
Corpus de Entrenamiento		
Frases	10.054	
Palabras	324.329	
Corpus de Referencia		
Frases	4.919	
Palabras	162.972	
Lexicón: Entrenamiento		
T.Etiq.	00:00:06	
OOVF+	2.795	82,25%
OOVF-	603	17,75%
NAF+	84.705	99,62%
NAF-	327	0,38%
AF+	72.265	96,95%
AF-	2.277	3,05%
S1	98,032	
S2	96,304	

Tabla 7.5: Corpus: ITU - Sistema: TnT - Banco de Experimentos: Especial

Datos Generales del Experimento										
Test	test11		test12		test13		test14		test15	
Máquina	liasun13		liasun13		liasun13		liasun13		liasun13	
T.Entr.	03:40:08		08:53:11		14:23:03		21:12:40		27:33:28	
Tamaño del Corpus de Entrenamiento										
Frases	1.054		2.054		3.054		4.054		5.054	
Palabras	32.774		68.322		96.775		126.302		162.521	
Tamaño del Corpus de Referencia										
Frases	9.919		9.919		9.919		9.919		9.919	
Palabras	324.780		324.780		324.780		324.780		324.780	
Lexicón: Entrenamiento										
T.Etiq.	00:01:15		00:01:37		00:01:52		00:02:11		00:02:26	
OOVF+	24.235	62,87%	14.862	65,32%	11.375	70,90%	9.330	73,85%	7.997	78,01%
OOVF-	14.315	37,13%	7.892	34,68%	4.669	29,10%	3.303	26,15%	2.254	21,99%
NAF+	172.907	98,58%	181.111	98,84%	185.763	99,13%	187.425	99,32%	173.623	99,47%
NAF-	2.498	1,42%	2.120	1,16%	1.627	0,87%	1.280	0,68%	929	0,53%
AF+	104.215	94,04%	113.103	95,21%	115.850	95,47%	118.234	95,78%	135.045	96,48%
AF-	6.610	5,96%	5.692	4,79%	5.496	4,53%	5.208	4,22%	4.932	3,52%
S1	92,788		95,164		96,369		96,985		97,501	
S2	85,991		90,403		92,601		93,745		95,216	
Lexicón: Entrenamiento+GALENA										
T.Etiq.	00:04:59		00:05:27		00:05:43		00:06:03		00:06:17	
OOVF+	13.037	61,02%	9.006	64,18%	7.391	72,40%	6.194	76,82%	5.615	80,88%
OOVF-	8.329	38,98%	5.026	35,82%	2.817	27,60%	1.869	23,18%	1.327	19,12%
NAF+	159.427	98,20%	161.919	98,78%	165.034	99,21%	166.431	99,40%	152.679	99,54%
NAF-	2.926	1,80%	2.003	1,22%	1.314	0,79%	1.000	0,60%	709	0,46%
AF+	131.088	92,93%	138.774	94,52%	140.989	95,12%	142.888	95,71%	158.611	96,45%
AF-	9.973	7,07%	8.052	5,48%	7.235	4,88%	6.398	4,29%	5.839	3,55%
S1	93,463		95,356		96,500		97,146		97,575	
S2	88,732		91,869		93,655		94,746		95,818	
Lexicón: Entrenamiento+ITU										
T.Etiq.	00:01:16		00:01:32		00:01:49		00:02:06		00:02:20	
OOVF+	0	-	0	-	0	-	0	-	0	-
OOVF-	0	-	0	-	0	-	0	-	0	-
NAF+	171.441	100%	171.441	100%	171.441	100%	171.441	100%	171.441	100%
NAF-	0	0%	0	0%	0	0%	0	0%	0	0%
AF+	143.027	93,28%	145.154	94,66%	146.045	95,24%	146.826	95,75%	147.514	96,20%
AF-	10.312	6,72%	8.185	5,34%	7.294	4,76%	6.513	4,25%	5.825	3,80%
S1	96,824		97,479		97,754		97,994		98,206	
S2	93,275		94,662		95,243		95,752		96,201	

Tabla 7.6: Corpus: ITU - Sistema: BRILL - Banco de Experimentos: 1

Datos Generales del Experimento										
Test	test21		test22		test23		test24		test25	
Máquina	covas		covas		covas		covas		covas	
T.Entr.	03:55:12		08:49:43		17:04:23		24:10:13		31:22:33	
Tamaño del Corpus de Entrenamiento										
Frases	1.054		2.054		3.054		4.054		5.054	
Palabras	33.192		71.367		100.218		129.764		166.064	
Tamaño del Corpus de Referencia										
Frases	9.919		9.919		9.919		9.919		9.919	
Palabras	321.237		321.237		321.237		321.237		321.237	
Lexicón: Entrenamiento										
T.Etiq.	00:01:26		00:01:49		00:02:13		00:02:41		00:02:51	
OOVF+	23.216	61,06%	14.198	64,44%	10.966	70,18%	9.073	72,79%	7.870	77,14%
OOVF-	14.804	38,94%	7.834	35,56%	4.660	29,82%	3.392	27,21%	2.332	22,86%
NAF+	169.564	98,62%	178.135	98,74%	182.740	99,10%	184.074	99,31%	183.836	99,48%
NAF-	2.380	1,38%	2.269	1,26%	1.666	0,90%	1.282	0,69%	961	0,52%
AF+	104.272	93,71%	113.529	95,56%	115.823	95,56%	118.401	95,94%	121.166	95,98%
AF-	7.001	6,29%	5.272	4,44%	5.382	4,44%	5.015	4,06%	5.072	3,02%
S1	92,471		95,213		96,355		96,983		97,396	
S2	85,394		90,693		92,661		93,812		94,573	
Lexicón: Entrenamiento+GALENA										
T.Etiq.	00:06:17		00:06:30		00:06:58		00:07:51		00:07:32	
OOVF+	13.051	61,30%	8.835	63,24%	7.413	71,94%	6.213	75,14%	5.443	79,08%
OOVF-	8.241	38,70%	5.136	36,76%	2.892	28,06%	2.056	24,86%	1.440	20,92%
NAF+	156.261	97,92%	158.642	98,70%	161.829	99,17%	163.115	99,40%	162.799	99,58%
NAF-	3.323	2,08%	2.088	1,30%	1.350	0,83%	980	0,60%	681	0,42%
AF+	130.087	92,68%	139.250	95,03%	140.749	95,26%	142.579	95,77%	144.753	95,94%
AF-	10.274	7,32%	7.286	4,97%	7.004	4,74%	6.294	4,23%	6.121	4,06%
S1	93,201		95,483		96,499		97,095		97,434	
S2	88,546		92,260		93,739		94,686		95,207	
Lexicón: Entrenamiento+ITU										
T.Etiq.	00:01:24		00:01:50		00:02:11		00:02:32		00:02:45	
OOVF+	0	-	0	-	0	-	0	-	0	-
OOVF-	0	-	0	-	0	-	0	-	0	-
NAF+	169.795	100%	169.795	100%	169.795	100%	169.795	100%	169.795	100%
NAF-	0	0%	0	0%	0	0%	0	0%	0	0%
AF+	141.289	93,30%	143.878	95,00%	143.281	94,61%	144.977	95,73%	145.603	96,14%
AF-	10.153	6,70%	7.564	5,00%	8.161	5,39%	6.465	4,27%	5.839	3,86%
S1	96,839		97,645		97,459		97,987		98,182	
S2	93,295		95,005		94,611		95,731		96,144	

Tabla 7.7: Corpus: ITU - Sistema: BRILL - Banco de Experimentos: 2

Datos Generales del Experimento										
Test	test31		test32		test33		test34		test35	
Máquina	ds.cesga		ds.cesga		ds.cesga		ds.cesga		ds.cesga	
T.Entr.	03:18:40		08:49:10		15:19:08		23:05:48		28:56:27	
Tamaño del Corpus de Entrenamiento										
Frases	1.054		2.054		3.054		4.054		5.054	
Palabras	33.284		69.764		97.557		127.230		162.819	
Tamaño del Corpus de Referencia										
Frases	9.919		9.919		9.919		9.919		9.919	
Palabras	324.482		324.482		324.482		324.482		324.482	
Lexicón: Entrenamiento										
T.Etiq.	00:01:24		00:01:54		00:02:08		00:02:29		00:02:46	
OOVF+	25.187	64,68%	14.451	63,51%	11.769	72,00%	9.601	75,55%	8.119	77,45%
OOVF-	13.753	35,32%	8.303	36,49%	4.577	28,00%	3.107	24,45%	2.364	22,55%
NAF+	173.063	98,75%	178.555	98,79%	182.177	99,10%	184.065	99,24%	182.840	99,42%
NAF-	2.194	1,25%	2.193	1,21%	1.652	0,90%	1.412	0,76%	1.061	0,58%
AF+	103.575	93,92%	115.541	95,50%	118.960	95,70%	121.369	96,10%	125.189	96,22%
AF-	6.710	6,08%	5.439	4,50%	5.347	4,30%	4.928	3,90%	4.909	3,78%
S1	93,017		95,089		96,432		97,088		97,431	
S2	86,287		90,439		92,944		94,219		94,826	
Lexicón: Entrenamiento+GALENA										
T.Etiq.	00:06:00		00:06:08		00:06:31		00:06:46		00:07:06	
OOVF+	14.140	63,60%	9.138	64,23%	7.942	74,15%	6.566	77,17%	5.683	78,57%
OOVF-	8.093	36,40%	5.090	35,77%	2.769	25,85%	1.943	22,83%	1.550	21,43%
NAF+	159.925	98,22%	159.740	98,67%	161.768	99,09%	163.181	99,35%	162.715	99,53%
NAF-	2.897	1,78%	2.161	1,33%	1.490	0,91%	1.064	0,65%	763	0,47%
AF+	129.652	92,99%	140.882	94,96%	143.782	95,53%	145.602	95,96%	147.834	96,14%
AF-	9.775	7,01%	7.471	5,04%	6.731	4,47%	6.126	4,03%	5.937	3,86%
S1	93,600		95,462		96,613		97,185		97,457	
S2	88,947		92,274		94,107		94,964		95,349	
Lexicón: Entrenamiento+ITU										
T.Etiq.	00:01:26		00:01:54		00:02:10		00:02:26		00:02:40	
OOVF+	0	-	0	-	0	-	0	-	0	-
OOVF-	0	-	0	-	0	-	0	-	0	-
NAF+	171.476	100%	171.476	100%	171.476	100%	171.476	100%	171.476	100%
NAF-	0	0%	0	0%	0	0%	0	0%	0	0%
AF+	143.546	93,82%	145.167	94,88%	146.135	95,51%	146.572	95,79%	147.135	96,17%
AF-	9.460	6,18%	7.839	5,12%	6.871	4,49%	6.434	4,21%	5.871	3,83%
S1	97,084		97,584		97,882		98,017		98,190	
S2	93,817		94,876		95,509		95,794		96,162	

Tabla 7.8: Corpus: ITU - Sistema: BRILL - Banco de Experimentos: 3

Datos Generales		
Test	testSP	
Máquina	liasun13	
T.Entr.	66:54:12	
Corpus de Entrenamiento		
Frases	10.054	
Palabras	324.329	
Corpus de Referencia		
Frases	4.919	
Palabras	162.972	
Lexicón: Entrenamiento		
T.Etiq.	00:01:46	
OOVF+	2.719	80,02%
OOVF-	679	19,98%
NAF+	84.705	99,62%
NAF-	327	0,38%
AF+	72.268	96,95%
AF-	2.274	3,05%
S1	97,987	
S2	96,211	
Lexicón: Entr.+GALENA		
T.Etiq.	00:03:47	
OOVF+	1.985	81,39%
OOVF-	454	18,61%
NAF+	75.123	99,66%
NAF-	253	0,34%
AF+	82.649	97,05%
AF-	2.508	2,95%
S1	98,027	
S2	96,618	
Lexicón: Entr.+ITU		
T.Etiq.	00:01:37	
OOVF+	0	-
OOVF-	0	-
NAF+	86.291	100%
NAF-	0	0%
AF+	74.180	96,74%
AF-	2.501	3,26%
S1	98,465	
S2	96,738	

Tabla 7.9: Corpus: ITU - Sistema: BRILL - Banco de Experimentos: Especial

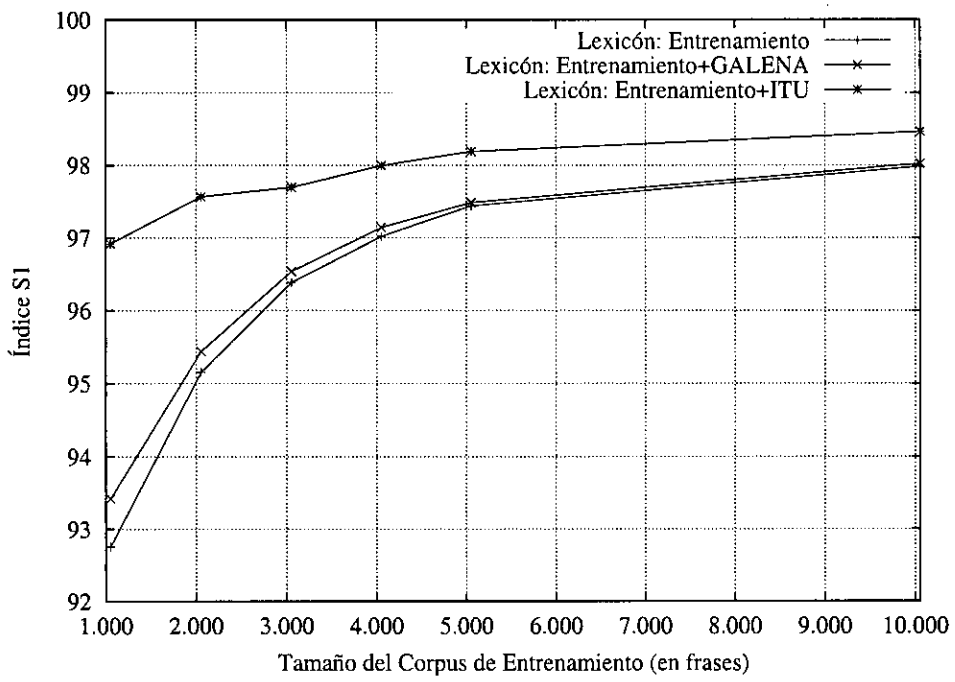


Figura 7.3: Corpus: ITU - Sistema: BRILL - Bancos: 1+2+3+Especial - Índice: S1

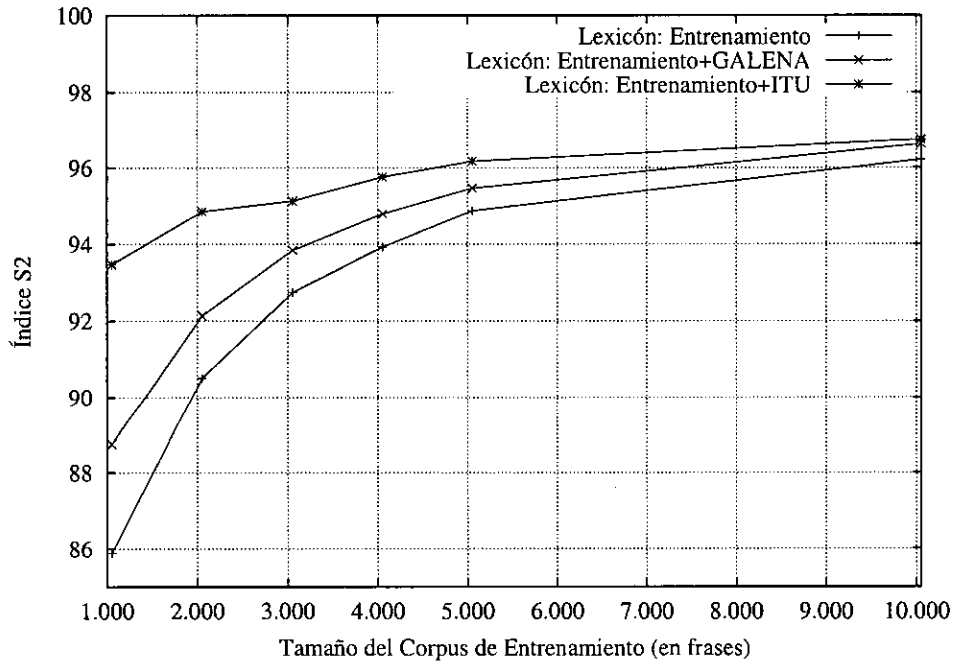


Figura 7.4: Corpus: ITU - Sistema: BRILL - Bancos: 1+2+3+Especial - Índice: S2

Datos Generales del Experimento					
Test	test11	test12	test13	test14	test15
Máquina	covas	covas	covas	covas	covas
T.Entr.	02:20:15	04:57:21	07:13:56	09:14:22	12:31:36
Tamaño del Corpus de Entrenamiento					
Frases	1.054	2.054	3.054	4.054	5.054
Palabras	32.774	68.322	96.775	126.302	162.521
Tamaño del Corpus de Referencia					
Frases	9.919	9.919	9.919	9.919	9.919
Palabras	324.780	324.780	324.780	324.780	324.780
Lexicón: Entrenamiento					
T.Etiq.	06:10:38	04:40:57	04:22:45	04:16:10	04:05:46
OOVF+	29.193 71,46%	16.533 72,65%	12.674 78,99%	10.431 82,56%	8.644 84,32%
OOVF-	11.656 28,54%	6.221 27,35%	3.370 21,01%	2.202 17,44%	1.607 15,68%
NAF+	167.099 96,29%	178.839 97,60%	183.991 98,18%	185.957 98,54%	172.501 98,82%
NAF-	6.437 3,71%	4.392 2,40%	3.399 1,82%	2.748 1,46%	2.051 1,18%
AF+	104.964 95,08%	113.896 95,87%	116.709 96,17%	118.978 96,38%	135.471 96,74%
AF-	5.431 4,92%	4.899 4,13%	4.637 3,83%	4.464 3,62%	4.560 3,26%
S1	92,756	95,223	96,488	97,101	97,469
S2	88,702	92,144	94,172	95,101	95,894

Tabla 7.10: Corpus: ITU - Sistema: JMX - Banco de Experimentos: 1

Datos Generales del Experimento					
Test	test21	test22	test23	test24	test25
Máquina	covas	covas	covas	covas	covas
T.Entr.	02:32:54	05:21:04	07:22:26	09:54:32	12:34:33
Tamaño del Corpus de Entrenamiento					
Frases	1.054	2.054	3.054	4.054	5.054
Palabras	33.192	71.367	100.218	129.764	166.064
Tamaño del Corpus de Referencia					
Frases	9.919	9.919	9.919	9.919	9.919
Palabras	321.237	321.237	321.237	321.237	321.237
Lexicón: Entrenamiento					
T.Etiq.	05:43:11	04:41:03	04:23:21	04:13:40	04:10:26
OOVF+	26.172 68,83%	16.250 73,75%	12.258 78,44%	10.250 82,23%	8.603 84,32%
OOVF-	11.848 31,17%	5.782 26,25%	3.368 21,56%	2.215 17,77%	1.599 15,68%
NAF+	165.629 96,32%	175.841 97,47%	181.186 98,25%	182.740 98,58%	182.693 98,86%
NAF-	6.315 3,68%	4.563 2,53%	3.220 1,75%	2.616 1,42%	2.104 1,14%
AF+	105.911 95,18%	114.066 96,01%	116.676 96,26%	119.060 96,47%	121.905 96,56%
AF-	5.362 4,82%	4.735 3,99%	4.529 3,74%	4.356 3,53%	4.333 3,44%
S1	92,676	95,305	96,539	97,140	97,498
S2	88,472	92,532	94,228	95,164	95,652

Tabla 7.11: Corpus: ITU - Sistema: JMX - Banco de Experimentos: 2

Datos Generales del Experimento					
Test	test31	test32	test33	test34	test35
Máquina	covas	covas	covas	covas	covas
T.Entr.	02:21:42	05:07:54	07:18:17	09:42:54	12:15:01
Tamaño del Corpus de Entrenamiento					
Frases	1.054	2.054	3.054	4.054	5.054
Palabras	33.284	69.764	97.557	127.230	162.819
Tamaño del Corpus de Referencia					
Frases	9.919	9.919	9.919	9.919	9.919
Palabras	324.482	324.482	324.482	324.482	324.482
Lexicón: Entrenamiento					
T.Etiq.	05:33:14	04:55:19	04:37:15	04:27:15	04:14:33
OOVF+	27.153 69,73%	16.349 71,85%	12.697 77,67%	10.292 80,98%	8.680 82,80%
OOVF-	11.787 30,27%	6.405 28,15%	3.649 22,33%	2.416 19,02%	1.803 17,20%
NAF+	168.470 96,12%	176.016 97,38%	180.518 98,19%	182.549 98,42%	181.572 98,73%
NAF-	6.787 3,88%	4.732 2,62%	3.311 1,81%	2.928 1,58%	2.329 1,27%
AF+	104.722 94,95%	116.074 95,94%	119.655 96,25%	121.897 96,50%	125.777 96,67%
AF-	5.563 5,05%	4.906 4,06%	4.652 3,75%	4.400 3,50%	4.321 3,33%
S1	92,561	95,055	96,421	96,997	97,394
S2	88,373	92,130	94,098	95,096	95,643

Tabla 7.12: Corpus: ITU - Sistema: JMX - Banco de Experimentos: 3

Datos Generales	
Test	testSP
Máquina	covas
T.Entr.	25:13:03
Corpus de Entrenamiento	
Frases	10.054
Palabras	324.329
Corpus de Referencia	
Frases	4.919
Palabras	162.972
Lexicón: Entrenamiento	
T.Etiq.	02:02:44
OOVF+	2.902 85,43%
OOVF-	496 14,57%
NAF+	84.321 99,16%
NAF-	711 0,84%
AF+	72.448 97,19%
AF-	2.094 2,81%
S1	97,974
S2	96,676

Tabla 7.13: Corpus: ITU - Sistema: JMX - Banco de Experimentos: Especial

Datos Generales del Experimento										
Test	test11		test12		test13		test14		test15	
Máquina	asterix		asterix		asterix		asterix		asterix	
T.Entr.	00:00:10		00:00:19		00:00:26		00:00:34		00:00:43	
Tamaño del Corpus de Entrenamiento										
Frases	1.054		2.054		3.054		4.054		5.054	
Palabras	32.774		68.322		96.775		126.302		162.521	
Tamaño del Corpus de Referencia										
Frases	9.919		9.919		9.919		9.919		9.919	
Palabras	324.780		324.780		324.780		324.780		324.780	
Lexicón: Entrenamiento										
T.Etiq.	00:02:04		00:01:59		00:01:43		00:01:25		00:00:55	
OOVF+	26.559	68,69%	16.211	71,24%	11.609	72,36%	9.495	75,16%	7.880	76,87%
OOVF-	11.991	31,17%	6.543	28,76%	4.435	27,64%	3.138	24,84%	2.371	23,13%
NAF+	172.907	98,58%	181.111	98,84%	185.763	99,13%	187.425	99,32%	173.623	99,47%
NAF-	2.498	1,42%	2.120	1,16%	1.627	0,87%	1.280	0,68%	929	0,53%
AF+	106.114	95,74%	114.403	96,30%	116.576	96,07%	118.842	96,27%	135.381	96,72%
AF-	4.711	4,26%	4.392	3,70%	4.770	3,93%	4.600	3,73%	4.596	3,28%
S1	94,088		95,980		96,664		97,223		97,568	
S2	88,818		92,274		93,300		94,313		95,362	
Lexicón: Entrenamiento+GALENA										
T.Etiq.	00:03:10		00:02:49		00:01:42		00:01:30		00:01:20	
OOVF+	14.699	68,80%	10.028	71,47%	7.404	72,53%	6.053	75,07%	5.346	77,01%
OOVF-	6.667	31,20%	4.004	28,53%	2.804	27,47%	2.010	24,93%	1.596	22,99%
NAF+	159.427	98,20%	161.919	98,78%	165.034	99,21%	166.431	99,40%	152.679	99,54%
NAF-	2.926	1,80%	2.003	1,22%	1.314	0,79%	1.000	0,60%	709	0,46%
AF+	135.277	95,90%	141.701	96,51%	142.724	96,29%	143.986	96,45%	159.434	96,95%
AF-	5.784	4,10%	5.125	3,49%	5.500	3,71%	5.300	3,55%	5.016	3,05%
S1	95,265		96,572		97,038		97,441		97,745	
S2	92,334		94,324		94,758		95,354		96,142	
Lexicón: Entrenamiento+ITU										
T.Etiq.	00:02:10		00:02:02		00:01:51		00:01:33		00:01:05	
OOVF+	0	-	0	-	0	-	0	-	0	-
OOVF-	0	-	0	-	0	-	0	-	0	-
NAF+	171.441	100%	171.441	100%	171.441	100%	171.441	100%	171.441	100%
NAF-	0	0%	0	0%	0	0%	0	0%	0	0%
AF+	147.205	95,99%	148.156	96,62%	148.324	96,73%	148.508	96,85%	148.708	96,97%
AF-	6.134	4,01%	5.183	3,38%	5.015	3,59%	4.839	3,15%	4.631	3,03%
S1	98,111		98,404		98,455		98,512		98,574	
S2	95,999		96,619		96,729		96,849		96,979	

Tabla 7.14: Corpus: ITU - Sistema: GALENA - Banco de Experimentos: 1

Datos Generales del Experimento										
Test	test21		test22		test23		test24		test25	
Máquina	asterix		asterix		asterix		asterix		asterix	
T.Entr.	00:00:10		00:00:20		00:00:27		00:00:35		00:00:45	
Tamaño del Corpus de Entrenamiento										
Frases	1.054		2.054		3.054		4.054		5.054	
Palabras	33.192		71.367		100.218		129.764		166.064	
Tamaño del Corpus de Referencia										
Frases	9.919		9.919		9.919		9.919		9.919	
Palabras	321.237		321.237		321.237		321.237		321.237	
Lexicón: Entrenamiento										
T.Etiq.	00:02:00		00:01:55		00:01:39		00:01:20		00:00:50	
OOVF+	26.490	69,67%	15.725	71,37%	11.239	71,46%	9.293	74,55%	7.828	76,73%
OOVF-	11.530	30,33%	6.307	28,63%	4.487	28,54%	3.172	25,45%	2.374	23,27%
NAF+	169.564	98,62%	178.135	98,74%	182.740	99,10%	184.074	99,31%	183.836	99,48%
NAF-	2.380	1,38%	2.269	1,26%	1.666	0,90%	1.282	0,69%	961	0,52%
AF+	106.559	95,84%	114.253	96,17%	116.557	96,17%	118.824	96,28%	121.670	96,38%
AF-	4.614	4,16%	4.548	3,83%	4.648	3,83%	4.592	3,72%	4.568	3,62%
S1	94,231		95,914		96,638		97,184		97,539	
S2	89,179		92,292		93,328		94,286		94,912	
Lexicón: Entrenamiento+GALENA										
T.Etiq.	00:02:59		00:02:40		00:01:33		00:01:25		00:01:13	
OOVF+	14.880	69,89%	9.996	71,55%	7.385	71,67%	6.181	74,76%	5.293	76,91%
OOVF-	6.412	30,11%	3.975	28,45%	2.920	28,33%	2.088	25,24%	1.590	23,09%
NAF+	156.261	97,92%	158.642	98,70%	161.829	99,17%	163.115	99,40%	162.799	99,58%
NAF-	3.323	2,08%	2.088	1,30%	1.350	0,83%	980	0,60%	681	0,42%
AF+	134.844	96,07%	141.231	96,38%	142.448	96,41%	143.662	96,50%	145.699	96,57%
AF-	5.517	3,93%	5.304	3,62%	5.305	3,59%	5.211	3,50%	5.175	3,43%
S1	95,252		96,461		97,019		97,442		97,682	
S2	92,620		94,218		94,796		95,355		95,711	
Lexicón: Entrenamiento+ITU										
T.Etiq.	00:02:03		00:01:58		00:01:45		00:01:27		00:01:00	
OOVF+	0	-	0	-	0	-	0	-	0	-
OOVF-	0	-	0	-	0	-	0	-	0	-
NAF+	169.795	100%	169.795	100%	169.795	100%	169.795	100%	169.795	100%
NAF-	0	0%	0	0%	0	0%	0	0%	0	0%
AF+	145.566	96,12%	145.959	96,38%	146.156	96,51%	146.338	96,63%	146.459	96,71%
AF-	5.876	3,88%	5.483	3,62%	5.286	3,49%	5.104	3,37%	4.983	3,29%
S1	98,170		98,293		98,354		98,411		98,448	
S2	96,119		96,379		96,509		96,629		96,709	

Tabla 7.15: Corpus: ITU - Sistema: GALENA - Banco de Experimentos: 2

Datos Generales del Experimento										
Test	test31		test32		test33		test34		test35	
Máquina	asterix		asterix		asterix		asterix		asterix	
T.Entr.	00:00:10		00:00:19		00:00:27		00:00:34		00:00:43	
Tamaño del Corpus de Entrenamiento										
Frases	1.054		2.054		3.054		4.054		5.054	
Palabras	33.284		69.764		97.557		127.230		162.819	
Tamaño del Corpus de Referencia										
Frases	9.919		9.919		9.919		9.919		9.919	
Palabras	324.482		324.482		324.482		324.482		324.482	
Lexicón: Entrenamiento										
T.Etiq.	00:02:02		00:01:58		00:01:41		00:01:23		00:00:52	
OOVF+	26.622	68,36%	16.026	70,43%	11.865	72,58%	9.568	75,29%	8.222	78,43%
OOVF-	12.318	31,64%	6.728	29,57%	4.481	27,42%	3.140	24,71%	2.261	21,57%
NAF+	173.063	98,75%	178.555	98,79%	182.177	99,10%	184.065	99,24%	182.840	99,42%
NAF-	2.194	1,25%	2.193	1,21%	1.652	0,90%	1.412	0,76%	1.061	0,58%
AF+	105.660	95,80%	116.446	96,25%	119.486	96,41%	121.983	96,58%	125.564	96,51%
AF-	4.625	4,20%	4.534	3,75%	4.461	3,59%	4.314	3,42%	4.534	3,49%
S1	94,102		95,853		96,735		97,267		97,578	
S2	88,646		92,164		93,642		94,637		95,166	
Lexicón: Entrenamiento+GALENA										
T.Etiq.	00:03:05		00:02:45		00:01:39		00:01:28		00:01:15	
OOVF+	15.242	68,56%	10.046	70,61%	7.798	72,81%	6.421	75,47%	5.686	78,62%
OOVF-	6.991	31,44%	4.182	29,39%	2.913	27,19%	2.088	24,53%	1.547	21,38%
NAF+	159.925	98,22%	159.740	98,67%	161.768	99,09%	163.181	99,35%	162.715	99,53%
NAF-	2.897	1,78%	2.161	1,33%	1.490	0,91%	1.064	0,65%	763	0,47%
AF+	133.891	96,03%	143.071	96,44%	145.440	96,63%	146.827	96,77%	148.742	96,73%
AF-	5.536	3,97%	5.282	3,56%	5.073	3,37%	4.901	3,23%	5.029	3,27%
S1	95,246		96,417		97,079		97,518		97,738	
S2	92,251		94,178		95,046		95,638		95,915	
Lexicón: Entrenamiento+ITU										
T.Etiq.	00:02:06		00:02:00		00:01:48		00:01:31		00:01:02	
OOVF+	0	-	0	-	0	-	0	-	0	-
OOVF-	0	-	0	-	0	-	0	-	0	-
NAF+	171.476	100%	171.476	100%	171.476	100%	171.476	100%	171.476	100%
NAF-	0	0%	0	0%	0	0%	0	0%	0	0%
AF+	146.992	96,07%	147.727	96,55%	147.941	96,69%	148.216	96,87%	148.278	96,91%
AF-	6.014	3,93%	5.279	3,45%	5.065	3,31%	4.790	3,13%	4.728	3,09%
S1	98,146		98,373		98,439		98,523		98,542	
S2	96,069		96,549		96,689		96,869		96,909	

Tabla 7.16: Corpus: ITU - Sistema: GALENA - Banco de Experimentos: 3

Datos Generales		
Test	testSP	
Máquina	asterix	
T.Entr.	00:01:24	
Corpus de Entrenamiento		
Frases	10.054	
Palabras	324.329	
Corpus de Referencia		
Frases	4.919	
Palabras	162.972	
Lexicón: Entrenamiento		
T.Etiq.	00:00:36	
OOVF+	2.703	79,54%
OOVF-	695	20,46%
NAF+	84.705	99,62%
NAF-	327	0,38%
AF+	72.289	96,97%
AF-	2.253	3,03%
S1	97,990	
S2	96,250	
Lexicón: Entr.+GALENA		
T.Etiq.	00:00:58	
OOVF+	1.945	79,75%
OOVF-	494	20,25%
NAF+	75.123	99,66%
NAF-	253	0,34%
AF+	82.755	97,18%
AF-	2.402	2,82%
S1	98,067	
S2	96,693	
Lexicón: Entr.+ITU		
T.Etiq.	00:00:47	
OOVF+	0	-
OOVF-	0	-
NAF+	86.291	100%
NAF-	0	0%
AF+	74.533	97,20%
AF-	2.148	2,80%
S1	98,681	
S2	97,198	

Tabla 7.17: Corpus: ITU - Sistema: GALENA - Banco de Experimentos: Especial

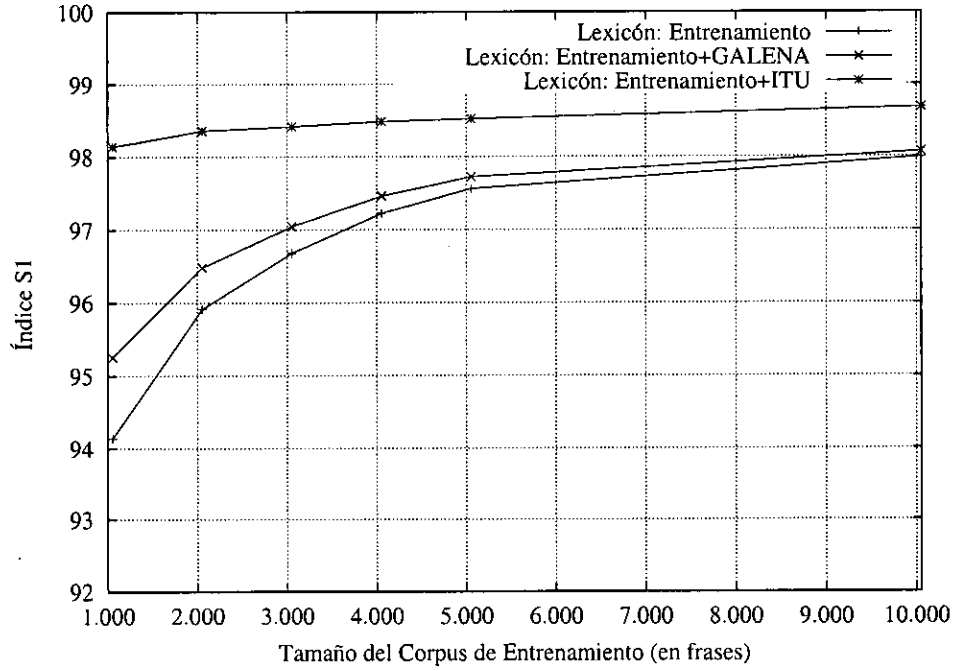


Figura 7.5: Corpus: ITU - Sistema: GALENA - Bancos: 1+2+3+Especial - Índice: S1

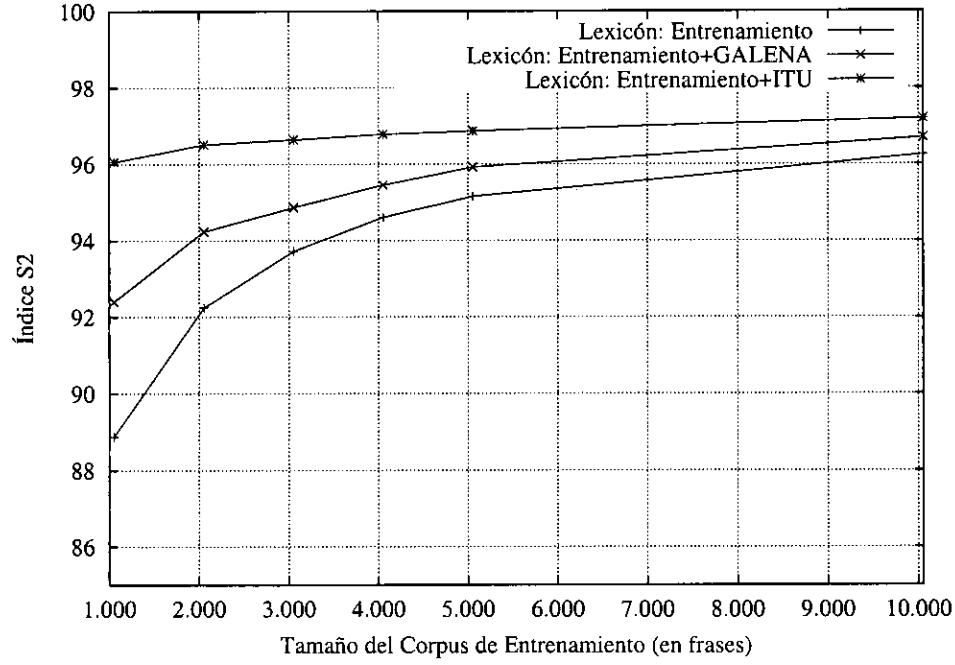


Figura 7.6: Corpus: ITU - Sistema: GALENA - Bancos: 1+2+3+Especial - Índice: S2

Tamaño del Corpus de Entrenamiento							
	Frases	1.054	2.054	3.054	4.054	5.054	10.054
	Palabras	33.284	69.764	97.557	127.230	162.819	324.329
Tamaño del Corpus de Referencia							
	Frases	9.919	9.919	9.919	9.919	9.919	4.919
	Palabras	324.482	324.482	324.482	324.482	324.482	162.972
Lexicón: Entrenamiento							
S1	TnT	94,035	95,854	96,910	97,466	97,730	98,032
	Brill	92,759	95,155	96,385	97,019	97,443	97,987
	JMX	92,664	95,194	96,483	97,079	97,454	97,974
	Galena	94,140	95,915	96,679	97,224	97,561	97,990
S2	TnT	88,653	92,103	93,964	94,984	95,528	96,304
	Brill	85,891	90,512	92,735	93,925	94,872	96,211
	JMX	88,516	92,269	94,166	95,120	95,730	96,676
	Galena	88,881	92,243	93,423	94,412	95,146	96,250

Tabla 7.18: Corpus: ITU - Resumen de Resultados

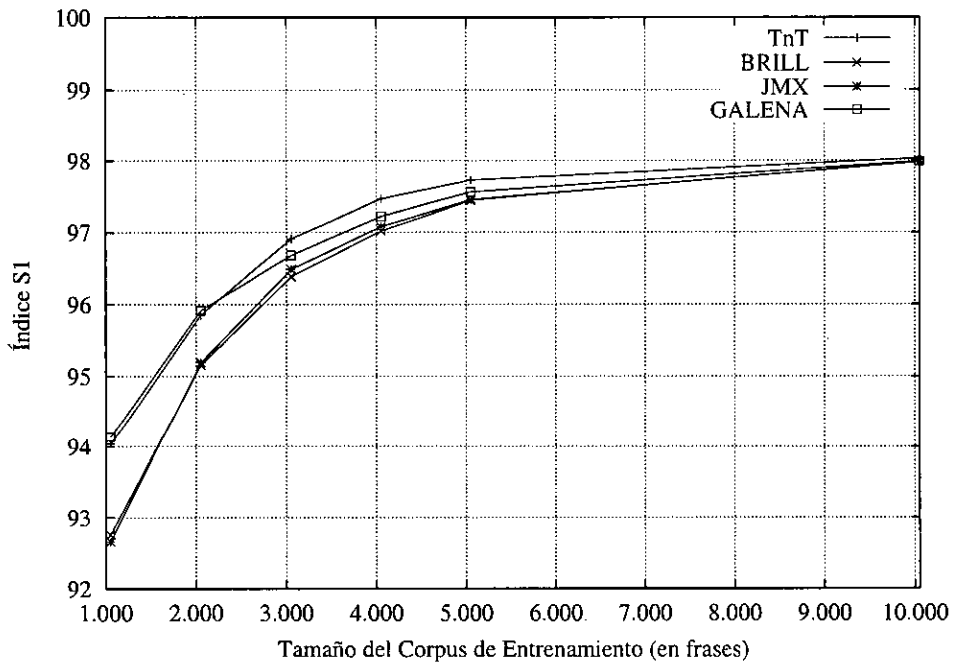


Figura 7.7: Corpus: ITU - Resumen de Resultados - Índice: S1

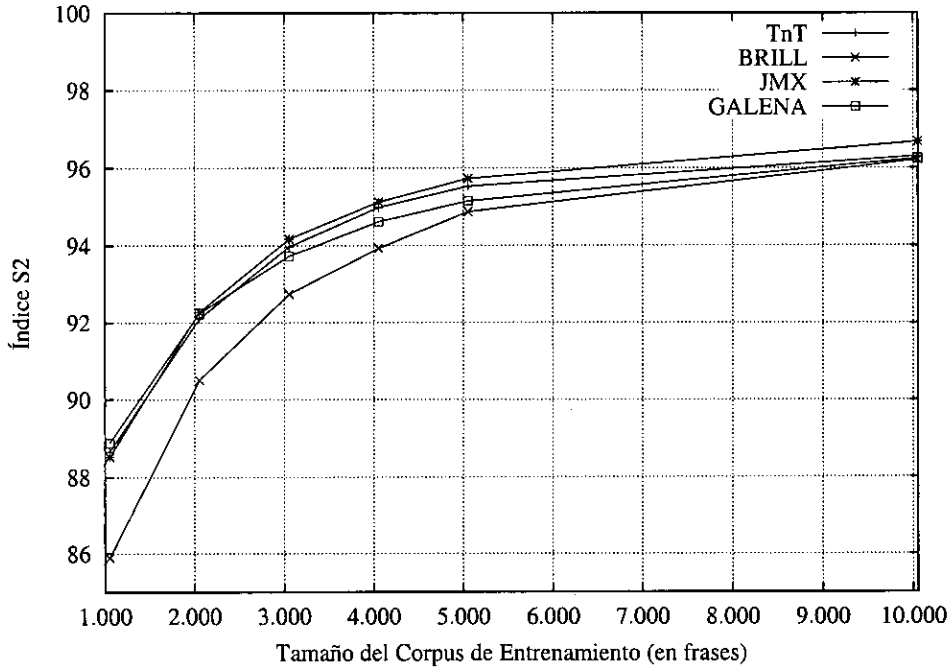


Figura 7.8: Corpus: ITU - Resumen de Resultados - Índice: S2

Datos Generales		
Test	testSusanne	
Máquina	asterix	
T.Entr.	00:00:02	
Corpus de Entrenamiento		
Frases	4.292	
Palabras	77.275	
Corpus de Referencia		
Frases	2.039	
Palabras	51.426	
Lexicón: Entrenamiento		
T.Etiq.	00:00:07	
OOVF+	2.523	58,38%
OOVF-	1.799	41,62%
NAF+	23.576	97,93%
NAF-	498	2,07%
AF+	21.752	94,45%
AF-	1.278	5,55%
S1	93,048	
S2	88,750	

Tabla 7.19: Corpus: SUSANNE - Sistema: TNT

Datos Generales		
Test	testSusanne	
Máquina	covas	
T.Entr.	34:26:39	
Corpus de Entrenamiento		
Frases	4.292	
Palabras	77.275	
Corpus de Referencia		
Frases	2.039	
Palabras	51.426	
Lexicón: Entrenamiento		
T.Etiq.	00:00:34	
OOVF+	2.376	54,97%
OOVF-	1.946	45,03%
NAF+	23.576	97,93%
NAF-	498	2,07%
AF+	21.237	92,21%
AF-	1.793	7,79%
S1	91,761	
S2	83,330	

Tabla 7.20: Corpus: SUSANNE - Sistema: BRILL

Datos Generales		
Test	testSusanne	
Máquina	covas	
T.Entr.	06:19:39	
Corpus de Entrenamiento		
Frases	4.292	
Palabras	77.275	
Corpus de Referencia		
Frases	2.039	
Palabras	51.426	
Lexicón: Entrenamiento		
T.Etiq.	01:08:30	
OOVF+	2.504	57,94%
OOVF-	1.818	42,06%
NAF+	21.880	90,89%
NAF-	2.194	9,11%
AF+	21.302	92,50%
AF-	1.728	7,50%
S1	88,838	
S2	87,035	

Tabla 7.21: Corpus: SUSANNE - Sistema: JMX

Datos Generales		
Test	testSusanne	
Máquina	asterix	
T.Entr.	00:00:21	
Corpus de Entrenamiento		
Frases	4.292	
Palabras	77.275	
Corpus de Referencia		
Frases	2.039	
Palabras	51.426	
Lexicón: Entrenamiento		
T.Etiq.	00:00:18	
OOVF+	2.463	56,99%
OOVF-	1.859	43,01%
NAF+	23.576	97,93%
NAF-	498	2,07%
AF+	21.688	94,17%
AF-	1.342	5,83%
S1	92,807	
S2	88,297	

Tabla 7.22: Corpus: SUSANNE - Sistema: GALENA

7.2.3 Análisis de resultados

Es conveniente comenzar el análisis de resultados con la siguiente reflexión. De los cuatro sistemas evaluados, TNT y GALENA responden claramente al paradigma de etiquetación puramente estocástico. De hecho, ambos utilizan como principio de funcionamiento los modelos de Markov ocultos. El sistema BRILL es un sistema basado en reglas, y el sistema JMX representa un acercamiento híbrido que utiliza fundamentos de las otras dos aproximaciones. Dicho esto, basta con echar un vistazo a las gráficas para observar que los sistemas TNT y GALENA ofrecen claramente mejores rendimientos. Por tanto, la primera conclusión importante que hay que extraer de los experimentos realizados es que para el proceso de etiquetación de textos en lenguaje natural, el marco probabilístico resulta ser más adecuado que las aproximaciones simbólicas.

Además, esta superioridad de prestaciones no sólo se presenta en los índices de acierto, sino también a nivel de los tiempos de entrenamiento y etiquetación. Mientras los sistemas TNT y GALENA entrenan y etiquetan a nivel de segundos, para el sistema JMX ambos procesos cosumen varias horas, y para el sistema BRILL los tiempos de entrenamiento pueden llegar a durar incluso días, aunque afortunadamente los tiempos de etiquetación bajan drásticamente a minutos.

El único que podría hacer frente a los etiquetadores puramente estocásticos es el sistema JMX. De hecho, es el que presenta los valores más altos para el índice S2 en prácticamente toda la gama de experimentos. Su mal rendimiento en el índice S1 se debe, como ya habíamos esbozado, a que es el único etiquetador que explota la ambigüedad potencial de las formas *a priori* no ambiguas. Mirando los porcentajes locales NAF+ y NAF-, y comparándolos con los

del resto de etiquetadores, se puede observar que la hipótesis de ambigüedad potencial asumida por el sistema JMX es muy arriesgada y degenera en un tratamiento erróneo para este fenómeno. Así pues, un cambio de implementación a este nivel podría hacer incrementar el rendimiento del sistema. No obstante, son precisamente sus elevados tiempos de ejecución los que limitan el uso práctico de este etiquetador. Un tiempo de entrenamiento alto todavía podría ser asumible, pero una vez construido el modelo, en la mayoría de las aplicaciones prácticas no es posible dedicar horas a la etiquetación de un nuevo texto.

El sistema BRILL, como ya habíamos comentado también, presenta múltiples atractivos relacionados con la forma en la que codifica los parámetros extraídos durante el entrenamiento. Este sistema establece decisiones sobre un conjunto de propiedades más rico que en el caso de los modelos puramente estocásticos, y las representa mediante un sencillo pero potente formalismo de reglas, mucho más fáciles de entender por el usuario que las probabilidades de transición y de generación de palabras en los etiquetadores probabilísticos. Después de realizar los experimentos, hemos visto que su comportamiento es razonable y comparable al del resto de sistemas. Sin embargo, es importante recordar que este etiquetador no proporciona información sobre la distribución de probabilidad de las etiquetas y las palabras, y por tanto su uso como componente probabilístico de un sistema mayor podría ser cuestionable.

Por todas estas razones, preferimos centrar ahora la discusión en una comparación más localizada de los sistemas TNT y GALENA. Como ya hemos dicho, ambos sistemas son implementaciones del algoritmo de Viterbi para HMM,s de segundo orden, y ambos manejan las palabras desconocidas mediante inferencias sucesivas de prueba de sufijos. Es decir, las diferencias fundamentales se plantean en la suavización de los parámetros relativos a los trigramas, bigramas y unigramas. El sistema TNT utiliza interpolación lineal, y el sistema GALENA implementa tanto la interpolación lineal como el método de *marcha atrás* o *back-off*. Sin embargo, como era de esperar, los rendimientos obtenidos por el sistema GALENA cuando trabaja con interpolación lineal son muy similares a los del sistema TNT, salvo por otro conjunto de diferencias menores entre ambos sistemas que comentaremos después. Por tanto, los datos presentados en este estudio para el sistema GALENA han sido obtenidos utilizando dicho sistema con el método *back-off*. De esta manera, hemos podido realizar una comparación casi directa entre los dos métodos de suavización considerados.

Así pues, la segunda conclusión importante que extraemos es que el método *back-off* se muestra más robusto para la suavización de parámetros en aquellas circunstancias donde los *corpora* de entrenamiento son pequeños. Es cierto que existen otras diferencias entre los sistemas TNT y GALENA que podrían ser la causa de esta diferencia de rendimiento. Por ejemplo, el sistema GALENA permite especificar un conjunto externo de sufijos[13] que produce una mejora del tratamiento de las palabras desconocidas en situaciones como la descrita anteriormente. Por contra, el sistema TNT incorpora otros mecanismos *ad hoc*, tales como el uso de expresiones regulares para la identificación de los numerales. Sin embargo, se ha comprobado que el efecto causado por este tipo de funcionalidades es menos representativo que el causado por el método de suavización. Como aspecto ligeramente negativo tan sólo se observa un ligero incremento de los tiempos de ejecución. La subida del tiempo de entrenamiento es debida a la mayor complejidad de los cálculos implicados en el método *back-off*. La subida del tiempo de etiquetación se debe fundamentalmente a la obtención de los parámetros $\beta(t_1, t_2)$, los cuales, debido a su excesivo número, no se calculan todos *a priori* sino que se calculan y se almacenan sólo los que se necesitan a medida que avanza el proceso de etiquetación. Esto produce el curioso efecto de que la velocidad de etiquetación no es constante, por lo que las últimas frases de un texto se etiquetan más rápidamente que las primeras. Por una razón similar, según aumenta el texto de

[13]En este estudio, hemos trabajado con un conjunto de unos 250 sufijos que cubren los modelos de flexión más comunes en español para sustantivos, adjetivos y verbos.

entrenamiento se pueden extraer más parámetros, y como consecuencia el tiempo de etiquetación disminuye en lugar de aumentar.

Pero en definitiva, vemos que por encima de aproximadamente 2.000 frases de entrenamiento, existen suficientes evidencias como para que la interpolación lineal y el tratamiento de palabras desconocidas estándar incrementen considerablemente el rendimiento del sistema TnT. Sin embargo, por debajo de ese punto, el sistema GALENA ofrece un rendimiento mejor. Esta conclusión es particularmente importante para poder afirmar que hemos alcanzado una técnica que en el futuro permitirá abordar con garantías el procesamiento automático de idiomas para los cuales apenas existen textos de referencia, como es el caso del gallego.

Respecto a la integración de diccionarios externos, el sistema GALENA puede realizar esta tarea mediante el método *Adding One* o mediante el método de Good-Turing. Los datos presentados corresponden a éste último, dado que hemos observado que se comporta ligeramente mejor. En efecto, cuando se integra un conjunto de palabras externo, se produce una importante alteración de los parámetros extraídos a partir del corpus de entrenamiento. El método de Good-Turing demuestra ser más elaborado y, por tanto, desvirtúa menos el modelo. No obstante, es importante recordar que este método no siempre es aplicable y existe el riesgo de utilizarlo incorrectamente. En nuestro caso, este tipo de integración se realiza considerando conjuntos aislados de palabras sobre cada una de las distintas etiquetas, y sólo cuando se cumple que $n_0 \gg n_1 \gg n_2$. Fuera de esta situación, se aplica la integración *Adding One*. Este tipo de comprobaciones, además de los cálculos inherentes al propio método, son la principal causa del incremento que se observa en el tiempo de entrenamiento. En cualquier caso, ambos métodos obtienen una ganancia de rendimiento superior al método de integración que implementa el etiquetador de BRILL, tal y como se muestra en las tablas del apéndice C.

Finalmente, es necesario recordar que uno de los principales inconvenientes de cualquier sistema de etiquetación es que los resultados obtenidos dependen en exceso del estilo de los textos de entrenamiento. En el futuro, la solución ideal sería quizás la utilización de un repositorio de modelos, uno por cada estilo de texto subceptible de ser etiquetado. Como primera consecuencia, los tiempos de entrenamiento bajos se convertirán en una necesidad, y esto es algo que sólo el marco probabilístico garantiza. Otros requerimientos pasan por la definición de funciones de distancia, capaces de determinar a qué estilo de los disponibles está más cercano un determinado texto de entrada, y por el diseño de aplicaciones que permitan a los expertos lingüistas construir más recursos y cubrir así cada vez más estilos. Por supuesto, existen otro tipo de técnicas que también tienen cabida aquí. Sin embargo, la mayoría de ellas precisan un estudio exhaustivo de los errores sistemáticos que cometen los etiquetadores, o de los fenómenos lingüísticos no correctamente formalizados, y una posterior incorporación manual por parte del usuario de información capaz de paliar dichos problemas. A lo largo de todo el presente trabajo, nosotros hemos querido cubrir las aproximaciones que utilizan únicamente la información que se puede extraer de los recursos lingüísticos de manera automática.

Parte III

Análisis sintáctico robusto y etiquetación

Capítulo 8

Análisis sintáctico estocástico

La comunicación humana depende de multitud de factores, pero todos ellos responden a una cierta regularidad y a una cierta estructura. El principal objetivo de la sintaxis dentro de la lingüística es el de intentar aislar dicha estructura. Hasta ahora, la única forma de sintaxis que permitían los métodos de etiquetación que hemos descrito es la simple consideración del orden secuencial de aparición de las palabras dentro de la frase, bien en términos de las propias palabras, o bien en términos de sus categorías léxicas. Desde este momento, nuestro propósito es escapar de la tiranía lineal impuesta por este tipo de modelos, introducir otras nociones de gramática más complejas, y comenzar a explorar su aplicación al proceso de etiquetación.

Después de introducir formalmente el concepto de gramática independiente del contexto estocástica, estudiaremos el problema del análisis sintáctico para este tipo de gramáticas. La tarea del análisis sintáctico[1] ha sido otra de las áreas de gran actividad dentro de la investigación en NLP durante los últimos años [Alonso 2000]. En el presente trabajo, no pretendemos realizar una cobertura de la viabilidad y complejidad de todas y cada una de las aproximaciones que se han desarrollado en este terreno, tal y como hicimos con las técnicas de etiquetación. En lugar de esto, comenzaremos introduciendo un sencillo e intuitivo mecanismo de análisis sintáctico, el algoritmo CYK [Kasami 1965, Younger 1967], que progresivamente iremos adaptando a nuestras necesidades.

Finalmente, veremos que al igual que los etiquetadores tienen que enfrentarse al problema de las palabras desconocidas, es decir, al problema de los diccionarios incompletos, los analizadores sintácticos deben saber enfrentarse al problema de las gramáticas incompletas. Por tanto, nuestro objetivo aquí es el de dejar preparado un marco de análisis sintáctico estocástico que permita experimentar cómodamente con técnicas de análisis sintáctico robusto[2] orientadas específicamente al problema de la etiquetación. Dichas técnicas serán descritas con detalle en el capítulo próximo.

8.1 Gramáticas independientes del contexto estocásticas

El modelo probabilístico más sencillo y más natural para representar las estructuras anidadas y los comportamientos recursivos de los lenguajes es quizás el de las gramáticas independientes del contexto probabilísticas, también llamadas estocásticas. Una gramática de este tipo es simplemente una gramática independiente del contexto que incorpora una probabilidad asociada a cada regla de producción. El propósito de dichas probabilidades es indicar que algunas operaciones de reescritura son más probables que otras. La única restricción impuesta por

[1] También denominada *parsing*.

[2] También denominado *robust parsing*.

este modelo es que las probabilidades de las reglas que comparten la misma parte izquierda, es decir, las reglas correspondientes a las distintas posibilidades de reescritura de un mismo símbolo, deben sumar 1. A continuación presentamos la definición formal.

Definición 8.1 Una *gramática independiente del contexto estocástica* se define como $G = (N, T, P, S)$, donde:

- N es el conjunto de *variables*, *símbolos no terminales*, o *categorías sintácticas*, es decir, símbolos que no forman parte de las frases del lenguaje generado por la gramática, pero que sirven de ayuda a la hora de describirlo.
- T es el conjunto de *símbolos terminales* o *categorías léxicas*, es decir, el conjunto de los símbolos o palabras que sí forman parte de las frases del lenguaje generado por la gramática.
- P es el conjunto de *producciones* o *reglas de reescritura* de la forma $A \to \alpha$, donde $A \in N$, es decir, es un símbolo no terminal, y $\alpha \in (N \cup T)^*$, es decir, es cualquier combinación de cero, uno o más símbolos terminales y no terminales. Cada regla tiene asociada una probabilidad, y este conjunto de probabilidades verifica:

$$\sum_{\alpha} P(A \to \alpha) = 1, \qquad \forall A \in N. \tag{8.1}$$

- S es un elemento destacado del conjunto N, que se denomina *axioma* o *símbolo inicial*. Todas las frases pertenecientes al lenguaje generado por la gramática han de tener un árbol de análisis cuya raíz debe ser el símbolo S.

Es importante señalar que cuando escribimos $P(A \to \alpha)$, lo que realmente queremos decir es $P(A \to \alpha|A)$. Por tanto, para cada símbolo no terminal A, la gramática proporciona la distribución de probabilidad de todas sus posibles transformaciones α. □

El método para diseñar una gramática estocástica, es decir, la manera de identificar los símbolos, las producciones y las probabilidades, puede ser manual cuando la gramática es pequeña. Pero en la práctica, para las gramáticas de los lenguajes naturales, tal y como se discutió ya en la sección 2.3.2, todos estos elementos se suelen extraer automáticamente de recursos lingüísticos especializados en forma de bancos de árboles[3].

Una gramática estocástica es un formalismo que describe o genera un lenguaje. El *lenguaje generado por una gramática* G se denota por $L(G)$. Asociado a dicho formalismo generador existen, como veremos más adelante, algoritmos para verificar si una determinada frase s pertenece o no a $L(G)$. Estos algoritmos son la base de los analizadores sintácticos[4], los cuales deben ser capaces también de obtener el árbol de análisis para dicha frase s, cuando $s \in L(G)$. Podría ocurrir incluso que dicho árbol de análisis no fuera único, en cuyo caso se dice que la gramática es *ambigua*. Al analizar una frase, el analizador debe ser capaz de obtener todos sus posibles árboles de análisis.

Por el momento, nos interesa centrarnos en el problema de cómo asignar una probabilidad a cada frase. La probabilidad de una frase s, de acuerdo con una gramática G, viene dada por:

$$P(s) = \sum_{t} P(s,t) = \sum_{t\,:\,\text{hojas}(t)=s} P(t),$$

[3] O *treebanks*.

[4] O también *parsers*.

donde la variable t recorre el espacio de todos los posibles árboles de análisis para los cuales la secuencia de nodos hoja, leída de izquierda a derecha, coincide con la frase s. Asumiendo la hipótesis de que las reglas de una gramática estocástica G son independientes, la probabilidad de un nodo cualquiera de un árbol t se calcula recursivamente como el producto de las probabilidades de sus subárboles locales y de la probabilidad de la regla de producción de G que los une. La probabilidad de un árbol t viene dada, por tanto, por la probabilidad de su nodo raíz.

Ejemplo 8.1 Sea $G = (N, T, P, S)$ una gramática independiente del contexto estocástica, donde $N = \{S, A, B, C\}$, $T = \{a, b\}$, el conjunto de reglas P, con sus respectivas probabilidades entre paréntesis, viene dado por:

$$\begin{array}{llllll} S \to A\,B & (0,25), & A \to B\,A & (0,5), & B \to C\,C & (0,1), \quad C \to A\,B \quad (0,2), \\ S \to B\,C & (0,75), & A \to a & (0,5), & B \to b & (0,9), \quad C \to a \quad (0,8), \end{array}$$

y $S = S$. La frase $s = b\,b\,a\,b$ tiene dos posibles árboles de análisis, tal y como se muestra en la figura 8.1. En esta figura, los símbolos no terminales de cada nodo llevan como subíndice la probabilidad de la regla mediante la cual generan los subárboles que encabezan. Así pues, la probabilidad de cada árbol es:

$$P(t_1) = 0,9 \times 0,9 \times 0,5 \times 0,5 \times 0,5 \times 0,9 \times 0,25 = 0,02278.$$
$$P(t_2) = 0,9 \times 0,9 \times 0,5 \times 0,5 \times 0,9 \times 0,2 \times 0,75 = 0,02733.$$

Y por tanto, $P(s) = P(t_1) + P(t_2) = 0,02733 + 0,02278 = 0,05011$. □

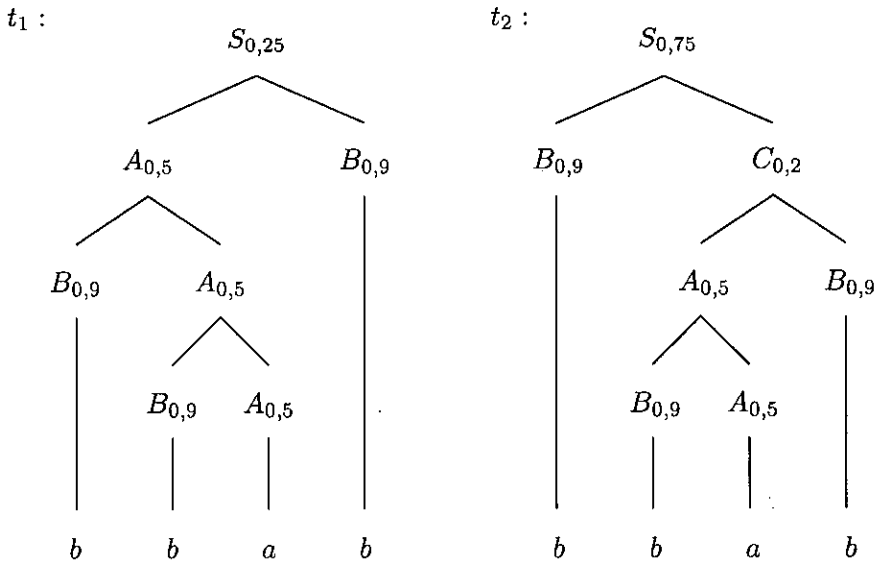


Figura 8.1: Árboles de análisis para la frase $s = b\,b\,a\,b$

8.1.1 Algunas características de las gramáticas estocásticas

A continuación citamos algunas circunstancias en las que puede ser conveniente el uso de gramáticas estocásticas, y también algunas ideas sobre sus limitaciones:

- A medida que las gramáticas se expanden para conseguir la mayor cobertura posible sobre grandes colecciones de textos, la ambigüedad crece también con ellas. Como ya sabemos, este fenómeno da como resultado la existencia de múltiples análisis estructurales diferentes

para una misma secuencia de palabras. Con el uso de gramáticas estocásticas, se puede obtener una cierta idea de la plausibilidad de cada uno de esos análisis. La gramática del ejemplo 8.1 no sólo permite calcular la probabilidad de una frase, en este caso $s = bbab$, sino que además nos indica cuál de los análisis es el más probable, en este caso t_2.

- Como hemos visto, el procedimiento más seguro para la construcción de una gramática para un lenguaje natural es la extracción de reglas a partir de un banco de árboles. Sin embargo, existen métodos capaces de realizar inferencia gramatical[5] sobre textos sin ningún tipo de marcación sintáctica. Aunque este tipo de inferencia gramatical *desde cero* es una difícil tarea todavía sin resolver, parece que el aprendizaje de gramáticas independientes del contexto no se puede realizar sin evidencias negativas, es decir, sin una provisión de ejemplos gramaticalmente incorrectos [Gold 1967], mientras que las gramáticas independientes del contexto estocásticas sí se pueden generar con sólo datos positivos [Horning 1969].

- Las gramáticas estocásticas presentan un buen compromiso de robustez. Los textos reales tienden a reflejar los errores léxicos y sintácticos más comunes de los hablantes. La manera obvia de evitar este problema es identificar las frases en las cuales se localizan los errores y eliminarlas del proceso de extracción de reglas. Sin embargo, se puede prescindir de esa tarea, y dejar que una gramática estocástica asigne de manera natural una probabilidad baja a las frases menos plausibles.

- En la práctica, existen idiomas para los cuales los n-gramas de palabras ($n > 1$) podrían constituir un modelo de lenguaje mejor que el representado por las gramáticas estocásticas. Un modelo de n-gramas de palabras tiene en cuenta dependencias contextuales entre elementos concretos del léxico que, en general, las gramáticas no utilizan.

- Las gramáticas estocásticas no parecen ser del todo *imparciales* en algunos aspectos, lo cual puede resultar inadecuado para determinadas aplicaciones. Por ejemplo, en general, la probabilidad de un árbol pequeño es mayor que la de un árbol grande. Esto podría no ser demasiado importante, ya que las frases de un lenguaje tienden a tener una cierta longitud intermedia. Pero no cabe duda de que una gramática estocástica asigna demasiada masa de probabilidad a las frases más cortas. De igual manera, en los árboles de análisis, los símbolos no terminales con un número bajo de posibles reescrituras se ven favorecidos sobre los no terminales con muchas posibilidades, ya que las reglas individuales de estos últimos tendrán probabilidades mucho más bajas.

- Por último, es importante comentar que no está claro que la sintaxis de todos los lenguajes naturales encaje dentro del marco de las gramáticas independientes del contexto, estocásticas o no estocásticas. Incluso aunque lo hiciera, el formalismo se queda muy justo y presenta limitaciones. No obstante, a pesar de su simplicidad, las gramáticas independientes del contexto todavía permiten expresar una gran variedad de las construcciones sintácticas que aparecen en la mayoría de los idiomas, y llevan asociados algoritmos muy eficientes para múltiples tareas de comprensión del lenguaje, una de las cuales es el análisis sintáctico, como veremos más adelante.

En definitiva, lo importante es que las gramáticas estocásticas proporcionan modelos probabilísticos del lenguaje. En un primer momento, cabría esperar, por tanto, que si todas las reglas de producción verifican la restricción (8.1), entonces

$$\sum_{s \in L(G)} P(s) = \sum_{t} P(t) = 1.$$

[5] *Grammar induction.*

Realmente, esto es cierto sólo si la masa de probabilidad de las reglas se acumula en un número finito de árboles de análisis. Así pues, consideremos el siguiente ejemplo.

Ejemplo 8.2 Sea G una gramática estocástica con un único símbolo no terminal S, un único símbolo terminal a, y un conjunto de reglas:

$$S \rightarrow a \quad (\tfrac{1}{3}),$$
$$S \rightarrow S\,S \quad (\tfrac{2}{3}).$$

Esta gramática genera frases de la forma a, aa, aaa, ... Sin embargo, las probabilidades de estas frases son de la forma:

$$\begin{aligned} P(a) &= \tfrac{1}{3} \\ P(aa) &= \tfrac{2}{3} \times \tfrac{1}{3} \times \tfrac{1}{3} = \tfrac{2}{27} \\ P(aaa) &= (\tfrac{2}{3})^2 \times (\tfrac{1}{3})^3 \times 2 = \tfrac{8}{243} \\ &\vdots \end{aligned}$$

La probabilidad del lenguaje $L(G)$ es la suma de la serie infinita $\frac{1}{3} + \frac{2}{27} + \frac{8}{243} + \ldots$, la cual tiende a $\frac{1}{2}$. Por tanto, la mitad de la masa de probabilidad ha desaparecido en el conjunto infinito de árboles que no generan frases de este lenguaje. □

Distribuciones de probabilidad como la del ejemplo anterior se denominan normalmente *distribuciones inconsistentes*. En la práctica, el uso de distribuciones inconsistentes no presenta excesivos problemas. A menudo, ni siquiera importa si la distribución es consistente o no, especialmente cuando nuestro objetivo principal es la comparación de las magnitudes de probabilidad de los diferentes análisis. Además, Chi y Geman demuestran que si los parámetros de nuestras gramáticas estocásticas se estiman a partir de bancos de árboles, siempre podemos obtener distribuciones de probabilidad consistentes [Chi y Geman 1998].

8.1.2 Relación entre gramáticas estocásticas y HMM,s

Las mismas tres preguntas fundamentales que planteamos en la sección 4.4 para los HMM,s son también aplicables a las gramáticas estocásticas. De hecho, un HMM se puede ver como una gramática regular estocástica. En el caso de las gramáticas estocásticas, las preguntas son las siguientes:

1. Dada una frase s y dada una gramática G, ¿cuál es la probabilidad $P(s|G)$, es decir, la probabilidad de la frase s de acuerdo con la gramática G?
2. ¿Cuál es el $\arg\max_t P(t|s,G)$, es decir, el árbol de análisis más probable para la frase s?
3. ¿Cómo podemos elegir el $\arg\max_G P(s|G)$, es decir, las probabilidades de las reglas de G que maximizan la probabilidad de una determinada frase s?

En relación con la primera pregunta, hemos visto que la probabilidad de una frase de acuerdo con una gramática se puede calcular como la suma de las probabilidades de todos sus árboles de análisis. Sin embargo, desafortunadamente, el número de análisis de una frase puede crecer exponencialmente con la longitud de la frase, de forma que la simple suma de las probabilidades de esos análisis no constituye un buen método. Existen algoritmos especializados para realizar de manera más eficiente este cálculo, tales como el *algoritmo de las probabilidades externas* y

el *algoritmo de las probabilidades internas*. Estos algoritmos se apoyan, respectivamente, en la definición de las probabilidades externas de un nodo de un árbol como

$$\alpha_j(p,q) = P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)n}|G)$$

y de las probabilidades internas como

$$\beta_j(p,q) = P(w_{pq}|N^j_{pq}, G)$$

donde w_{ik} es la secuencia de palabras $w_i, w_{i+1}, \ldots, w_k$ y N^j_{pq} es un subárbol encabezado por el símbolo no terminal N^j que produce la secuencia de palabras w_{pq}. El algoritmo de las probabilidades externas es un algoritmo de programación dinámica similar al *algoritmo hacia adelante* de los HMM,s, y el algoritmo de las probabilidades internas es similar al *algoritmo hacia atrás*.

Para resolver el problema que se plantea en la segunda pregunta, existe también un algoritmo especializado. Una vez más, la estrategia para encontrar el análisis más probable de una frase es similar al algoritmo que busca el camino más probable a través de un HMM que produce dicha frase (el algoritmo de Viterbi). El método para gramáticas estocásticas es una variante del algoritmo de las probabilidades internas, que busca el elemento que maximiza la suma y recuerda qué regla dio lugar a ese máximo. Al igual que en el caso de un HMM, este método funciona gracias a la hipótesis de independencia de las reglas gramaticales. El resultado es un algoritmo de análisis sintáctico de complejidad $\mathcal{O}(n^3m^3)$, donde n es el número de palabras de la frase a analizar y m es el número de símbolos no terminales de la gramática.

Por último, tal y como sugiere la tercera pregunta, existen algoritmos para el *entrenamiento* de gramáticas estocásticas. La idea de entrenamiento aquí es la misma que la del aprendizaje o inferencia gramatical, pero en un sentido limitado. Se asume que la estructura de la gramática, es decir, los conjuntos de símbolos terminales, no terminales y reglas, se conoce de antemano. El entrenamiento de la gramática es entonces simplemente un proceso que intenta optimizar las probabilidades de las reglas. Para determinar estas probabilidades respetando la restricción (8.1), nos gustaría calcular $P(A \to \alpha)$ como

$$P(A \to \alpha) = \frac{\text{número de veces que aparece la regla } A \to \alpha}{\sum_{\beta} \text{número de veces que aparece la regla } A \to \beta}$$

donde A es cualquier símbolo no terminal y α y β son cualquier combinación de símbolos terminales y no terminales. Si tenemos disponible un corpus anotado sintácticamente, es decir, un banco de árboles, las probabilidades se pueden calcular directamente (como se discutió en la sección 2.3.2). Si no tenemos un corpus anotado disponible, al igual que en el caso de los HMM,s, se puede construir un algoritmo de entrenamiento EM[6], que toma una gramática estocástica inicial y un corpus no anotado, y ajusta las probabilidades de las reglas de la gramática de manera que las frases de dicho corpus obtengan la máxima probabilidad posible. Las limitaciones de los métodos de entrenamiento para HMM,s también están presentes aquí, pues sólo garantizan que se encuentren máximos locales [Charniak 1993].

La descripción de todas estas estrategias desborda el ámbito del presente trabajo. Una excelente introducción a los mismos puede verse en [Manning y Schütze 1999, cap. 11]. A partir de ahora preferimos centrar nuestro interés en los algoritmos de análisis sintáctico, los cuales, una vez adaptados al marco estocástico, también pueden proporcionar en sí mismos soluciones a los problemas planteados en las preguntas 1 y 2 de una manera más intuitiva.

[6] *Expectation-Maximization* (maximización de la esperanza), cuya parte E combina las probabilidades internas y externas dando lugar al algoritmo *inside-outside* (hacia adentro y hacia afuera).

8.2 Algoritmo de análisis sintáctico CYK

Como ya hemos esbozado anteriormente, existen multitud de técnicas de análisis sintáctico bien definidas, y multitud de estudios sobre la viabilidad y complejidad de las mismas. Introduciremos ahora un algoritmo básico, el algoritmo CYK [Kasami 1965, Younger 1967], y discutiremos cómo adaptarlo a nuestros objetivos.

El algoritmo CYK[7] es un algoritmo de análisis sintáctico tabular, es decir, basado en programación dinámica, y ascendente, es decir, parte de la frase e intenta llegar al símbolo inicial de la gramática. Aunque resulta de gran interés debido a su simplicidad, presenta dos inconvenientes principales:

- La complejidad temporal es $\mathcal{O}(n^3)$, donde n es el número de palabras de la frase a analizar, independientemente de dicha frase.
- La complejidad espacial es $\mathcal{O}(n^2)$, independientemente también de la frase a analizar.

Aún con todo esto, el algoritmo CYK es fácilmente adaptable a la experimentación con técnicas de análisis sintáctico robusto, tal y como veremos más adelante.

En esta sección, consideraremos el enunciado del algoritmo para gramáticas en forma normal de Chomsky, sin producciones ϵ, dejando para las siguientes secciones la adaptación de este algoritmo básico para su funcionamiento extendido sobre gramáticas independientes del contexto arbitrarias, su integración dentro del marco estocástico, y la discusión de otras mejoras capaces de reducir la complejidad temporal. A continuación definimos el formato que han de seguir las gramáticas tratables por esta primera versión del algoritmo.

Definición 8.2 Una gramática independiente del contexto en *Forma Normal de Chomsky*, sin producciones ϵ, se define como $G = (N, T, P, S)$, donde:

- N es el conjunto de símbolos no terminales.
- T es el conjunto de símbolos terminales.
- P es el conjunto de reglas de producción, y todas las reglas han de ser de la forma

$$A \rightarrow B\ C \qquad \text{ó} \qquad A \rightarrow a$$

  donde A, B y C son cualquier símbolo no terminal y a es cualquier símbolo terminal.
- S es el axioma.

Es decir, sólo se permiten reglas cuya parte derecha contenga o bien dos no terminales, o bien un terminal. Por tanto, no se permiten tampoco reglas de la forma $A \rightarrow \epsilon$ (a no ser $S \rightarrow \epsilon$, como caso especial sólo cuando $\epsilon \in L(G)$). □

La esencia del algoritmo consiste en la construcción de una tabla de análisis triangular, cuyas celdas se denotan por N_{ij}, para $1 \leq i \leq n-j+1$ y $1 \leq j \leq n$, donde n es el número de palabras de la frase a analizar. Cada celda contendrá un subconjunto de los símbolos no terminales de la gramática. Un símbolo no terminal A estará en N_{ij} si y sólo si $A \stackrel{*}{\Rightarrow} w_i, w_{i+1}, \ldots, w_{i+j-1}$, es decir, si A deriva en un número finito de pasos la subfrase que comienza en la posición i y contiene j palabras. La frase pertenece al lenguaje generado por la gramática si el axioma se encuentra en la celda N_{1n}. La descripción formal del algoritmo es la siguiente.

[7] Tradicionalmente conocido como algoritmo Cocke-Younger-Kasami, aunque realmente fue descubierto de manera independiente por distintas personas: Hays presentó una versión de este algoritmo, la cual atribuye a J. Cocke [Hays 1967]; Younger utilizó este algoritmo para mostrar que la complejidad temporal del problema de la pertenencia para lenguajes independientes del contexto es $\mathcal{O}(n^3)$ [Younger 1967]; anteriormente Kasami había presentado ya un algoritmo de análisis similar [Kasami 1965].

Algoritmo 8.1 Algoritmo de análisis sintáctico CYK, para determinar si una frase $s = w_1, w_2, \ldots, w_n$ pertenece al lenguaje generado por una gramática en forma normal de Chomsky, sin ϵ producciones, $G = (N, T, P, S)$:

1. Se inicializa la primera fila de la tabla de análisis, utilizando las reglas que generan directamente los símbolos terminales, como sigue:

$$N_{i1} = \{A \mid A \rightarrow w_{i1} \in P\}, \qquad 1 \leq i \leq n.$$

2. Para $j = 2, 3, \ldots, n$, hacer lo siguiente:

   (a) Para $i = 1, 2, \ldots, n - j + 1$, hacer lo siguiente:

      i. Inicializar N_{ij} al conjunto vacío.

      ii. Para $k = 1, 2, \ldots, j - 1$, añadir a N_{ij} todos los símbolos no terminales A para los cuales $A \rightarrow B\ C \in P$, con $B \in N_{ik}$ y $C \in N_{(i+k)(j-k)}$.

La frase s pertenece a $L(G)$ si y sólo si $S \in N_{1n}$. □

Ejemplo 8.3 La figura 8.2 muestra la tabla de análisis CYK para la gramática del ejemplo 8.1 y la frase $s = b\,b\,a\,b$. Dado que $S \in N_{14}$, la frase s pertenece al lenguaje generado por dicha gramática. □

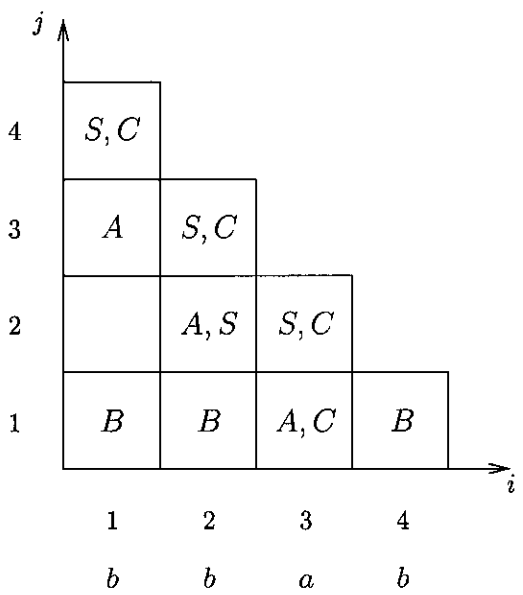


Figura 8.2: Tabla de análisis CYK para la frase $s = b\,b\,a\,b$

En general, la presencia de un símbolo no terminal en una celda N_{ij} implica la existencia de un subárbol encabezado por dicho símbolo, que cubre la subfrase $w_i, w_{i+1}, \ldots, w_{i+j-1}$. Por tanto, si el símbolo inicial de la gramática está presente en la celda N_{1n}, está garantizado que existe un árbol de cobertura total para la frase $w_1, w_2, \ldots, w_n$, cuya raíz es el símbolo inicial, y por ello se puede decir que la frase pertenece al lenguaje generado por la gramática.

Como veremos más adelante, la obtención de los árboles de análisis se puede realizar mediante un recorrido recursivo sobre las celdas de la tabla CYK. Si la celda N_{1n} está vacía, entonces no existirá ningún árbol de cobertura total. Sin embargo, desde cualquier otra celda o celdas no vacías, se pueden recuperar directamente todos aquellos subárboles que nos ineteresen, y posteriormente aplicar técnicas de análisis sintáctico robusto que operen con ellos.

8.3 Algoritmo de Earley

Como paso previo a la extensión del algoritmo CYK, presentamos ahora el acercamiento de Earley, un método de análisis sintáctico para gramáticas independientes del contexto arbitrarias [Earley 1970]. Este algoritmo presenta también complejidad cúbica en tiempo y cuadrática en espacio, con respecto a la longitud de la frase de entrada, pero tiene la particularidad de que se puede reducir la complejidad en tiempo a cuadrática si la gramática no es ambigua, o incluso llevar las dos complejidades mencionadas a lineales, bajo ciertas hipótesis[8].

La idea central del algoritmo se basa en la generación de ítems de la forma $[A \to \alpha \bullet \beta, i, j]$, donde A es un símbolo no terminal, α y β son cualquier combinación de símbolos terminales y no terminales, $A \to \alpha\beta$ es una regla de la gramática, y $0 \leq i \leq j \leq n$, siendo n el número de palabras de la frase de entrada. La existencia de un ítem $[A \to \alpha \bullet \beta, i, j]$ implica que $\alpha \stackrel{*}{\Rightarrow} w_{i+1}, \ldots, w_j$. Por tanto, si $\beta = \epsilon$, existe un árbol de análisis con raíz A que cubre la subfrase $w_{i+1}, \ldots, w_j$.

Además existe un tipo especial de ítems de la forma $[w, i-1, i]$, para indicar que el símbolo terminal w se encuentra en la posición i de la frase de entrada. Los pasos del algoritmo se describen a continuación.

Algoritmo 8.2 Algoritmo de análisis sintáctico de Earley, para determinar si una frase $s = w_1, w_2, \ldots, w_n$ pertenece al lenguaje generado por una gramática independiente del contexto $G = (N, T, P, S)$:

1. Paso de inicialización:

   (a) Generar los ítems correspondientes a los símbolos terminales. Es decir, para cada palabra w_i, $1 \leq i \leq n$, de la frase de entrada, se genera el ítem $[w_i, i-1, i]$.

   (b) Generar los ítems iniciales $[S \to \bullet\alpha, 0, 0]$, tales que $S \to \alpha \in P$ y S es el axioma de la gramática. Esta segunda fase del paso de inicialización se denomina también regla **Init**.

2. Paso de recurrencia:

   Para cada uno de los ítems generados en el paso de inicialización y para cada uno de los ítems que se van generando en cada iteración de este paso de recurrencia, verificar si se cumple alguna de las siguientes tres condiciones, y ejecutar la acción correspondiente:

Nomenclatura	Condición	Acción
Regla **Pred**	Si el ítem es de la forma $[A \to \alpha \bullet B\beta, i, j]$ y existe alguna regla $B \to \gamma$	Generar el ítem: $[B \to \bullet\gamma, j, j]$
Regla **Scan**	Si el ítem es de la forma $[A \to \alpha \bullet w\beta, i, j]$ y existe algún ítem $[w, j, j+1]$	Generar el ítem: $[A \to \alpha w \bullet \beta, i, j+1]$
Regla **Comp**	Si el ítem es de la forma $[A \to \alpha \bullet B\beta, i, j]$ y existe algún ítem $[B \to \gamma\bullet, j, k]$	Generar el ítem: $[A \to \alpha B \bullet \beta, i, k]$

   Repetir este paso de recurrencia hasta que no se puedan generar más ítems.

Una vez generados todos los ítems, si entre éstos se encuentra uno de la forma $[S \to \alpha\bullet, 0, n]$, la frase pertenece al lenguaje generado por la gramática. □

[8] Por ejemplo, con la mayoría de las gramáticas de los lenguajes de programación.

La obtención de los árboles de análisis se puede realizar partiendo de los ítems que determinan si la frase pertenece al lenguaje. Dichos ítems serán los ítems raíz de cada árbol. El resto de nodos se obtienen recursivamente estudiando cómo se produce cada nuevo ítem involucrado, es decir, a partir de los ítems y de las reglas que lo generan. Una construcción alternativa es la propuesta por Vilares, donde se identifican estructuras sintácticas e ítems, permitiendo la generación de bosques compartidos al mismo tiempo que se realiza el reconocimiento sintáctico [Vilares 1992].

Ejemplo 8.4 Consideremos $G = (N, T, P, S)$, una gramática independiente del contexto, donde $N = \{S, A\}$, $T = \{a, b, c\}$, el conjunto de reglas P viene dado por:

$$S \to a\ A\ b, \qquad A \to a\ A\ b, \qquad A \to c,$$

y $S = S$. De acuerdo con la gramática G, los ítems generados por el algoritmo de Earley para la frase $s = a\,a\,c\,b\,b$ son los siguientes:

Identificador	Ítem	Operación que lo genera
1	$[a, 0, 1]$	
2	$[a, 1, 2]$	
3	$[c, 2, 3]$	
4	$[b, 3, 4]$	
5	$[b, 4, 5]$	
6	$[S \to \bullet aAb, 0, 0]$	Init
7	$[S \to a \bullet Ab, 0, 1]$	Scan(6,1)
8	$[A \to \bullet aAb, 1, 1]$	Pred(7)
9	$[A \to \bullet c, 1, 1]$	Pred(7)
10	$[A \to a \bullet Ab, 1, 2]$	Scan (8,2)
11	$[A \to \bullet aAb, 2, 2]$	Pred(10)
12	$[A \to \bullet c, 2, 2]$	Pred(10)
13	$[A \to c\bullet, 2, 3]$	Scan(12,3)
14	$[A \to aA \bullet b, 1, 3]$	Comp(10,13)
15	$[A \to aAb\bullet, 1, 4]$	Scan(14,4)
16	$[S \to aA \bullet b, 0, 4]$	Comp(7,15)
17	$[S \to aAb\bullet, 0, 5]$	Scan(16,5)

La existencia del ítem $[S \to aAb\bullet, 0, 5]$ indica que la frase s pertenece a $L(G)$. El árbol de análisis de dicha frase se muestra en la figura 8.3. En esta figura, los ítems 6, 8 y 12 no se han representado. Se trata de ítems con el punto al principio de la parte derecha de la regla, y que por tanto derivan la cadena vacía ϵ. De igual manera, los ítems 7, 10, 14 y 16 se podrían también eliminar del árbol. En este caso, se trata de ítems con el punto en algún lugar intermedio de la parte derecha de la regla. Al eliminarlos, sus hijos pasarían a ser hijos de su primer ítem ancestro que tenga el punto al final de la parte derecha de la regla, y obtendríamos un árbol isomorfo al de la figura 8.4, que constituye el verdadero árbol de análisis de la frase s. □

8.4 Algoritmo CYK extendido

Como ya hemos comentado anteriormente, el algoritmo CYK puede extenderse para trabajar con cualquier tipo de gramática independiente del contexto, no restringiéndose a la representación en forma normal de Chomsky. Una alternativa para conseguir esto es a través de la manipulación de la gramática original, convirtiéndola a forma normal de Chomsky sin ϵ-producciones, para lo

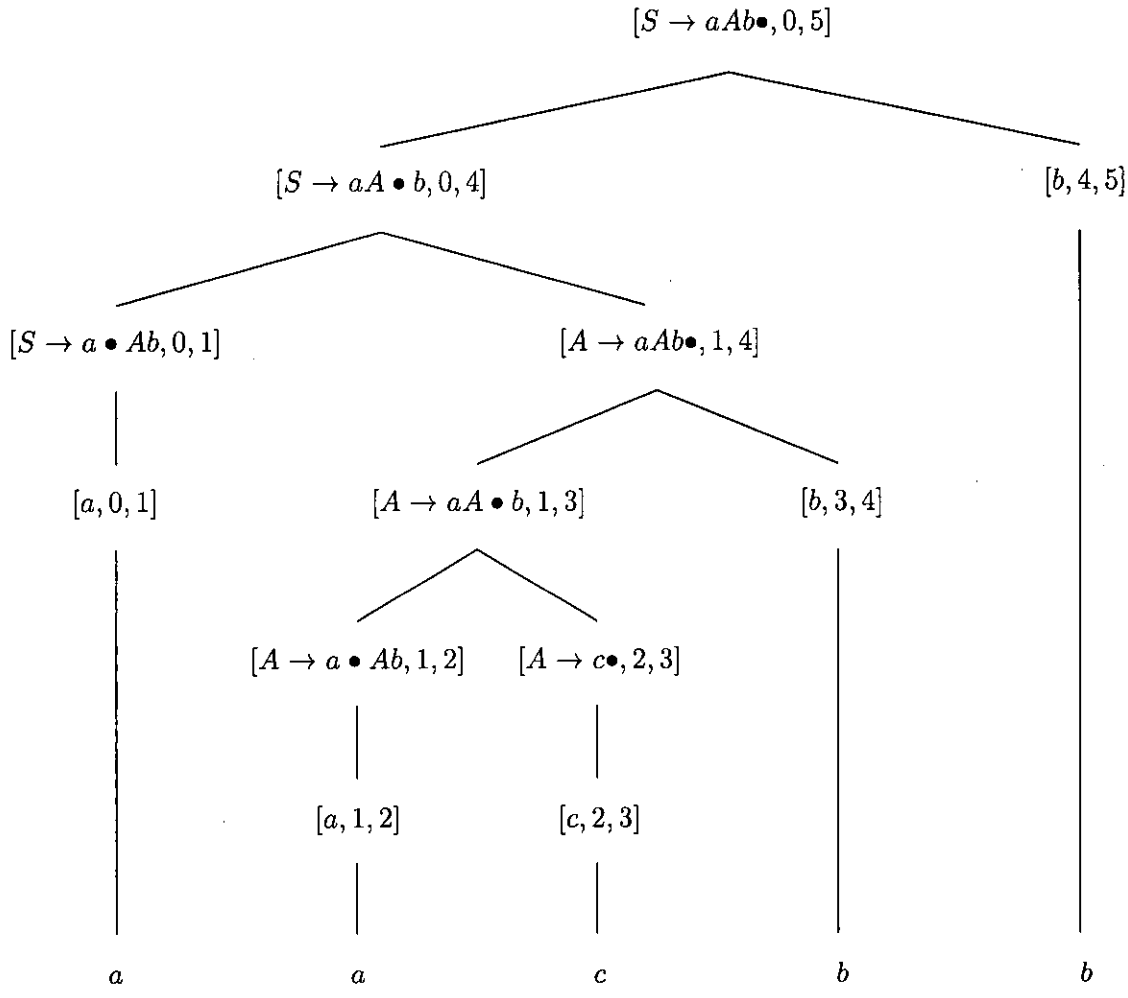


Figura 8.3: Árbol de análisis Earley para la frase $s = a\,a\,c\,b\,b$

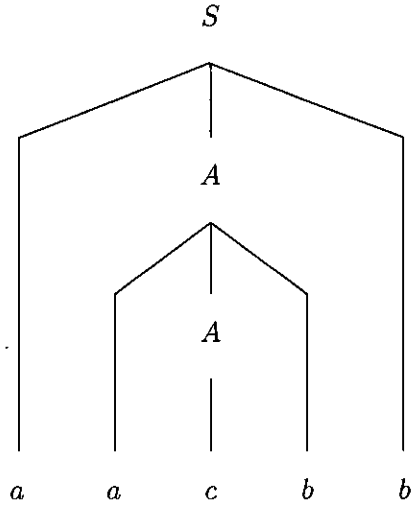


Figura 8.4: Árbol de análisis para la frase $s = a\,a\,c\,b\,b$

cual existen diversos algoritmos [Kelley 1998]. Pero el problema fundamental de esta alternativa radica en que es necesaria una reestructuración de la gramática, que conlleva la aparición de nuevos símbolos no terminales auxiliares y una pérdida importante de comprensión de la misma.

Esto provoca que la interpretación de la tabla de análisis sea mucho más compleja, y que los árboles de análisis a los que pueda dar lugar no se correspondan con los árboles lógicos o intuitivos que refleja la gramática original.

Siguiendo los trabajos de [Erbach 1994] y de [Chappelier y Rajman 1998], surge otra alternativa mucho más interesante, que consiste en readaptar el algoritmo para que pueda trabajar con cualquier gramática independiente del contexto de manera directa. El nuevo algoritmo, que denominaremos *algoritmo CYK extendido*, puede verse como un algoritmo Earley puramente ascendente, con ítems generalizados y sin predicción, o bien como un algoritmo CYK con binarización dinámica de la gramática.

Este nuevo algoritmo puede manejar gramáticas independientes del contexto tanto estocásticas como no estocásticas. Para una frase de entrada es posible, por tanto, saber si pertenece o no al lenguaje generado por la gramática, calcular su probabilidad (no sólo la de sus análisis), obtener los N análisis más probables de dicha frase o de cualquier subsecuencia de la misma y, lo que es más importante, todo ello manteniendo la estructura de la gramática original. También es posible la extracción explícita de los árboles de análisis asociados a la frase de entrada o a cualquier subfrase de ella.

En esta sección, comenzaremos presentando el algoritmo CYK extendido básico, dejando de lado por un momento las consideraciones estocásticas. Además, a pesar de que el algoritmo puede trabajar con gramáticas independientes del contexto arbitrarias, nos ocuparemos sólo de aquellas gramáticas que estén formadas por reglas *no parcialmente lexicalizadas*. En este tipo de gramáticas, los símbolos terminales aparecen sólo en reglas de la forma $A \rightarrow w_1\, w_2 \ldots w_k$. La mayoría de las veces k será igual a 1. Los casos donde $k > 1$ corresponden a las palabras compuestas, expresiones hechas o locuciones. Esta restricción no es crítica para el algoritmo, y se introduce únicamente para aislar el procesamiento de las reglas léxicas en el paso de inicialización, que en la práctica se realiza típicamente mediante el acceso a un diccionario.

8.4.1 Aproximación no estocástica

La estructura de procesamiento del algoritmo sigue siendo la tabla de análisis del algoritmo CYK básico, con la excepción de que las celdas contendrán ahora ítems similares a los que se utilizan en el algoritmo de Earley, en lugar de los símbolos no terminales utilizados por el algoritmo original.

La tabla de análisis[9] es una matriz triangular inferior con $\frac{n(n+1)}{2}$ celdas, donde n es el número de palabras de la frase que se desea analizar. Cada celda N_{ij} de la tabla contiene dos listas de ítems:

- Los ítems de la primera lista de una celda N_{ij}, denominada lista tipo 1, representan los símbolos no terminales que derivan la subsecuencia $w_i, w_{i+1}, \ldots, w_{i+j-1}$, es decir, si A es uno de esos símbolos, entonces $A \stackrel{*}{\Rightarrow} w_i w_{i+1} \ldots w_{i+j-1}$, y el ítem correspondiente se denota como $[A; i, j]$.

- Los ítems de la segunda lista de una celda N_{ij}, denominada lista tipo 2, representan análisis parciales α de la subsecuencia $w_i, w_{i+1}, \ldots, w_{i+j-1}$, es decir, secuencias α de símbolos no terminales tales que $\alpha \stackrel{*}{\Rightarrow} w_i w_{i+1} \ldots w_{i+j-1}$ para los cuales existe al menos una regla en la gramática de la forma $A \rightarrow \alpha\beta$, donde β es una secuencia no vacía de símbolos no terminales ($\beta = \epsilon$ es precisamente el caso que consideran los ítems de la lista tipo 1). Estos ítems se denotan como $[\alpha \bullet \ldots; i, j]$.

  Como se puede comprobar, los ítems de la lista tipo 2 representan una generalización de las reglas punteadas utilizadas en el algoritmo de Earley, donde sólo se representa la

[9] También denominada *chart*.

parte inicial de la regla (es decir, lo que se ha analizado haste ese momento), y se obvia la parte izquierda (que de hecho aún no ha sido reescrita) y la parte final (que aún no ha sido analizada). Esto proporciona una representación mucho más compacta de las reglas punteadas, y se puede realizar debido a la naturaleza ascendente del análisis.

Además, a cada ítem de cualquiera de las dos listas se le asocia también la lista de todas sus posibles producciones. Esto permite una factorización que evita la repetición del ítem para cada una de esas producciones, obtiene los ítems una sola vez, aún cuando puedan ser producidos varias veces en la misma celda, y permite una mayor velocidad en el proceso de análisis. Para propósitos de extracción de los árboles de análisis, cada producción contiene una referencia explícita a los ítems de las celdas correspondientes que han sido utilizados para crearlos.

Algoritmo 8.3 Algoritmo de análisis sintáctico CYK extendido, para determinar si una frase $s = w_1, w_2, \ldots, w_n$ pertenece al lenguaje generado por una gramática independiente del contexto, con reglas no parcialmente lexicalizadas, $G = (N, T, P, S)$:

1. **Paso de inicialización.** Este paso consiste en rellenar todas las celdas de la tabla de análisis para las cuales existe una regla léxica asociada con las palabras o secuencias de palabras de la frase de entrada. Es decir, se rellenan las listas tipo 1 de las celdas de las primeras filas de la tabla. Más concretamente, si una regla $A \rightarrow w_i \ldots w_{i+j-1} \in P$, el ítem $[A; i, j]$ se añade a la lista tipo 1 de la celda N_{ij}.

   Este paso de inicialización es importante ya que refleja todas las interpretaciones potenciales de secuencias de palabras correspondientes a las formas compuestas que se tendrán en cuenta. Por ejemplo, si la frase a analizar contiene la subsecuencia `guardia civil` en la posición i, todos los no terminales que producen `guardia` (respectivamente, `civil`) se almacenan en la celda N_{i1} (respectivamente, $N_{(i+1)1}$), y se almacenan en la celda N_{i2} todos los no terminales que producen `guardia civil`[10].

   Para completar el paso de inicialización, se necesita una fase de autorrellenado, que actualice las listas tipo 2 de estas celdas. Esta fase también se utiliza en el paso de análisis que veremos a continuación, por lo que es ahí donde se desarrolla con detalle.

2. **Paso de análisis.** Este paso consiste en aplicar dos fases, la fase de rellenado estándar y la fase de autorrellenado que se explican a continuación, sobre todas las celdas N_{ij} de la tabla de análisis. Dichas fases se aplican fila a fila, de forma ascendente.

   Más concretamente, la fase de autorrellenado se aplica sobre todas las celdas, mientras que la fase de rellenado estándar se aplica sobre todas las celdas excepto las de la primera fila de la tabla.

   - **Fase de rellenado estándar.** Esta fase es un procedimiento de rellenado de celdas en el que se consideran todas las combinaciones de ítems de dos celdas N_{ik} y $N_{(i+k)(j-k)}$ que puedan producir una nueva (y eventualmente parcial) interpretación para la celda N_{ij}, es decir, que puedan generar un nuevo ítem. Como se puede observar, la combinación de celdas es igual a la del algoritmo CYK básico.

     Más concretamente, se combina un ítem $[\alpha \bullet \ldots; i, k]$ de la lista 2 de una celda, con un ítem $[B; i + k, j - k]$ de la lista tipo 1 de otra celda $(1 \leq k < j)$, si y sólo si existe

[10] Otro aspecto importante de este paso de inicialización es el que permite que el algoritmo CYK extendido pueda ser utilizado en aplicaciones de procesamiento de voz, con sólo acomodar en la misma tabla de análisis las hipótesis producidas por una entrada acústica dada. En este caso, el tratamiento de todas las interpretaciones del mismo instante de tiempo es exactamente igual al mencionado previamente para las palabras compuestas. Cada instante de tiempo de un *word lattice* (grafo de palabras) se corresponde con una columna de la tabla.

una regla de la forma $A \rightarrow \alpha B \beta$ en la gramática. Si $\beta = \epsilon$, el ítem $[A; i, j]$ se añade a la lista tipo 1 de la celda N_{ij}. En caso contrario, se añade el ítem $[\alpha B \bullet \ldots; i, j]$ a la lista tipo 2 de la celda N_{ij}.

Hay que tener en cuenta que se añade el ítem completo siempre y cuando éste no exista ya. En el caso de que el ítem generado ya exista, sólo se añade la nueva producción a la sublista de producciones del mismo.

- **Fase de autorrellenado**. Esta fase es un procedimiento mediante el cual, para cada ítem $[B; i, j]$ de la lista tipo 1 de cada celda, y para cada regla de la forma $A \rightarrow B\beta$, se hace lo siguiente:
  - Si $\beta = \epsilon$, se añade el ítem $[A; i, j]$ a la lista tipo 1 de la celda. Nótese que sobre este nuevo ítem generado se debe aplicar nuevamente la fase de autorrellenado.
  - Si $\beta \neq \epsilon$, se añade el ítem $[B \bullet \ldots; i, j]$ a la lista tipo 2 de la celda.

  Esta segunda fase es necesaria para considerar el encadenamiento de reglas unitarias de la forma $A \rightarrow B$, y para mantener actualizadas las listas de tipo 2 después de que la fase de rellenado estándar haya sido realizada.

Una vez más, $s \in L(G)$ si el ítem $[S; 1, n]$ está en la celda N_{1n}. □

Ejemplo 8.5 Sea $G = (N, T, P, S)$ una gramática independiente del contexto, donde $N =$ { `S`, `GN`, `GP`, `GV`, `Det`, `N`, `Adj`, `Np`, `P`, `V` }, $T =$ { `El`, `el`, `gato`, `tejado`, `rojo`, `Luis`, `de`, `en`, `está` }, el conjunto de reglas P viene dado por:

```
S   → GN GV              Det → El
GN  → Det N                  | el
    | Det N Adj         N   → gato
    | Det N GP               | tejado
    | Det N Adj GP      Adj → rojo
    | Np               Np  → Luis
GP  → P GN             P   → de
GV  → V                      | en
    | V GP              V   → está
    | V GN GP
```

y $S =$ `S`. La tabla de análisis para la frase de entrada `El gato de Luis está en el tejado rojo` es la que se muestra en la figura 8.5. En dicha figura, hemos omitido los índices i y j de los ítems, ya que éstos quedan reflejados, respectivamente, por la columna y la fila en la que se sitúa cada ítem. Las flechas punteadas gruesas muestran el recorrido de ítems que habría que seguir para extraer el árbol de análisis de la frase completa. Las flechas continuas más delgadas muestran otro conjunto de subárboles correspondientes a algunas porciones de dicha frase. □

8.4.2 Producciones ϵ

Si la gramática contiene reglas ϵ, es decir, reglas de la forma $A \rightarrow \epsilon$, el algoritmo debe completarse con las dos siguientes consideraciones:

- Cuando se produce un ítem $[\alpha \bullet \ldots; i, j]$ en la lista tipo 2 de una celda, para cada no terminal B que produzca ϵ (es decir, tal que $B \rightarrow \epsilon \in P$), y que pueda completar ese ítem (es decir, tal que $A \rightarrow \alpha B \beta \in P$), se hace lo siguiente:

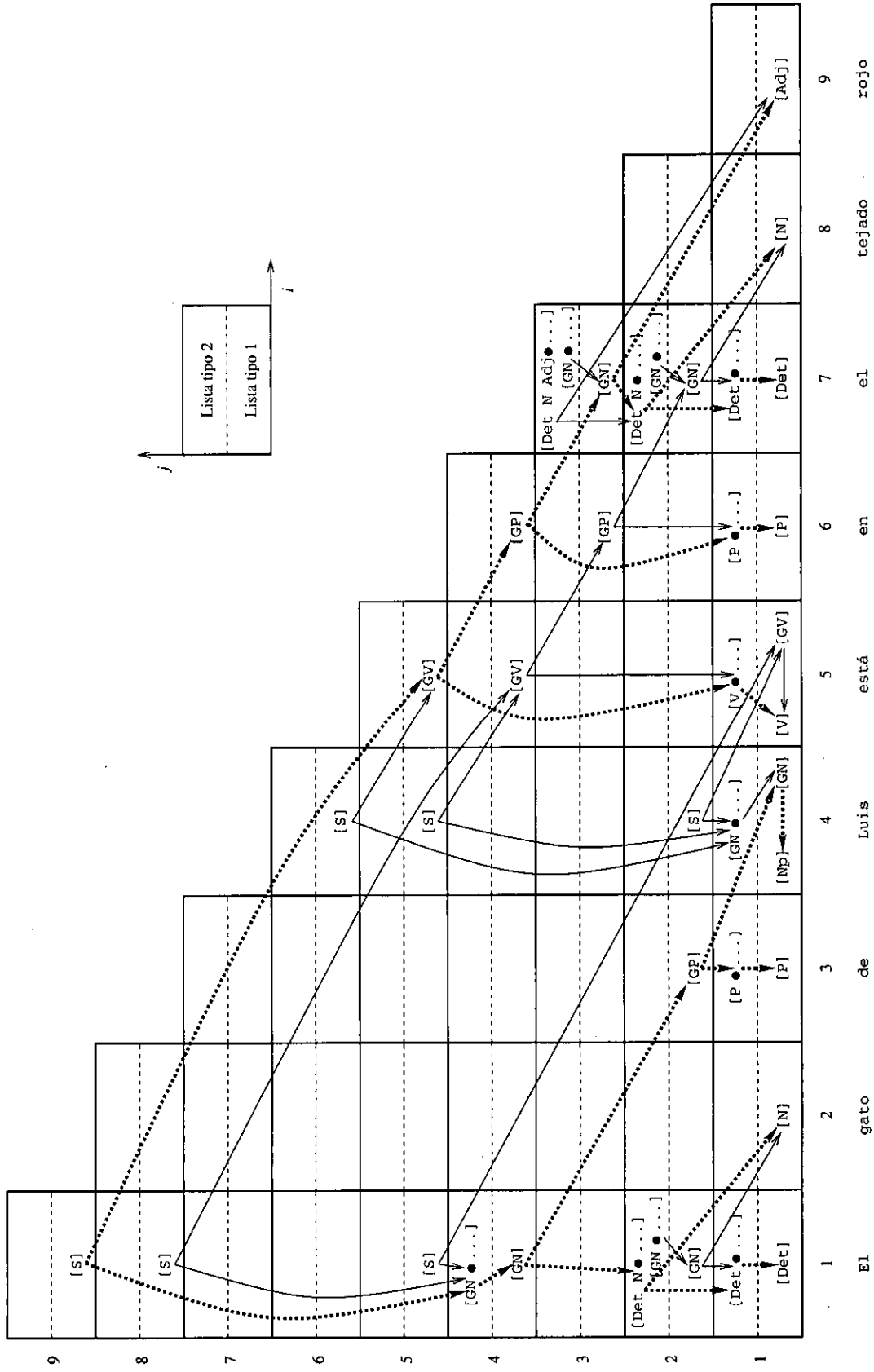


Figura 8.5: Tabla de análisis CYK extendido para la frase `El gato de Luis está en el tejado rojo`

- Si $\beta = \epsilon$, se añade el ítem $[A; i, j]$ a la lista tipo 1 de la misma celda.
- Si $\beta \neq \epsilon$, se añade el ítem $[\alpha B \bullet \ldots; i, j]$ a la lista tipo 2 de la misma celda.

Este proceso debe ser realizado recursivamente, es decir, se debe aplicar también a los ítems nuevos que se van generando, hasta que no se puedan añadir más ítems.

- La fase de autorrellenado debe ser también modificada. A lo que ya existía se debe añadir lo siguiente. Para cada ítem $[B; i, j]$ de la lista tipo 1 de una celda, si existe una regla de la forma $A \rightarrow CB\beta$, donde $C \rightarrow \epsilon$, entonces:
  - Si $\beta = \epsilon$, se añade el ítem $[A; i, j]$ a la lista tipo 1 de la misma celda.
  - Si $\beta \neq \epsilon$, se añade el ítem $[CB \bullet \ldots; i, j]$ a la lista tipo 2 de la misma celda.

8.4.3 Extracción de los árboles de análisis

La extracción de los árboles de análisis a partir de la tabla se realiza recorriendo las producciones de los ítems de tipo 1 de la celda superior que contengan el símbolo inicial de la gramática, si es que existe alguno. En todo caso, el procedimiento que se describe aquí es también aplicable a cualquier otra celda, para la extracción de los subárboles.

Cada producción de estos ítems se considera como un árbol de análisis. Cabe destacar que cada producción consta de una referencia, que surge de la fase de autorrellenado, o de dos, que surgen de la fase de rellenado estándar, por lo que la estructura a recorrer es un árbol en el que cada nodo contiene uno o dos hijos. Para cada uno de estos árboles o producciones del ítem elegido como raíz, se realiza un recorrido recursivo de manera que:

1. Se crea el nodo raíz, etiquetado con el símbolo del ítem elegido.
2. Por cada ítem al que se llega en el recorrido se genera un nodo, hijo del nodo generado por el ítem origen. Si el ítem es de tipo 1, se etiqueta el nodo con el símbolo incluido en dicho ítem. Por el contrario, si el ítem es de tipo 2, este nodo no se etiqueta.
3. Los nodos no etiquetados se eliminan, haciendo que sus hijos etiquetados pasen a ser hijos de su primer ancestro etiquetado.
4. Se añaden las palabras de la frase.

Las figuras 8.6 y 8.7 muestran, respectivamente, el árbol de análisis antes de la eliminación de los nodos no etiquetados y el árbol definitivo, para la frase del ejemplo 8.5.

8.4.4 Consideraciones estocásticas

Para las gramáticas estocásticas, la probabilidad de un árbol de análisis es el producto de la probabilidad de la regla utilizada para generar los subárboles del nodo raíz, y de las probabilidades de cada uno de esos subárboles[11].

En el caso del algoritmo CYK extendido, se puede definir la probabilidad de una producción p de un ítem tipo 1 $[A; i, j]$ como la probabilidad del subárbol de análisis correspondiente a la interpretación A de la secuencia $w_i, \ldots, w_{i+j-1}$ para esa producción p. O lo que es lo mismo, se puede calcular la probabilidad de una producción p del ítem $[A; i, j]$ como el producto de

[11] Aunque una vez más, mediante el uso de probabilidades logarítmicas, se podrían cambiar los productos por sumas, lo cual acelera los cálculos y evita los problemas de pérdida de precisión resultantes de multiplicar números menores que 1.

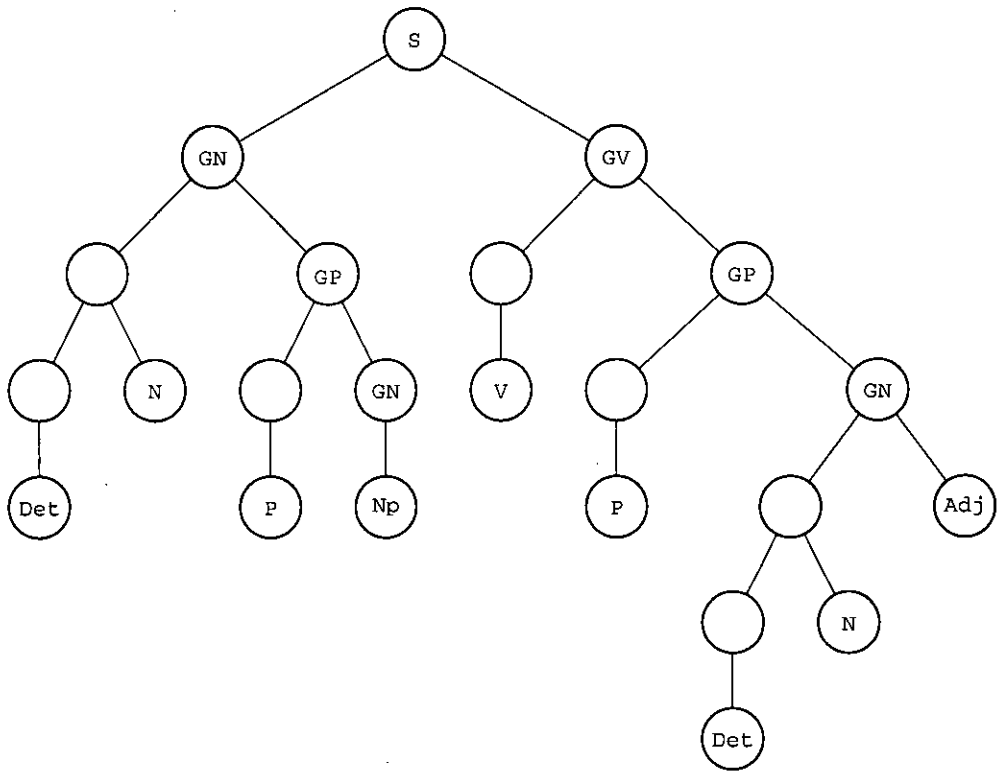


Figura 8.6: Esqueleto previo del árbol de análisis de la frase `El gato de Luis está en el tejado rojo`

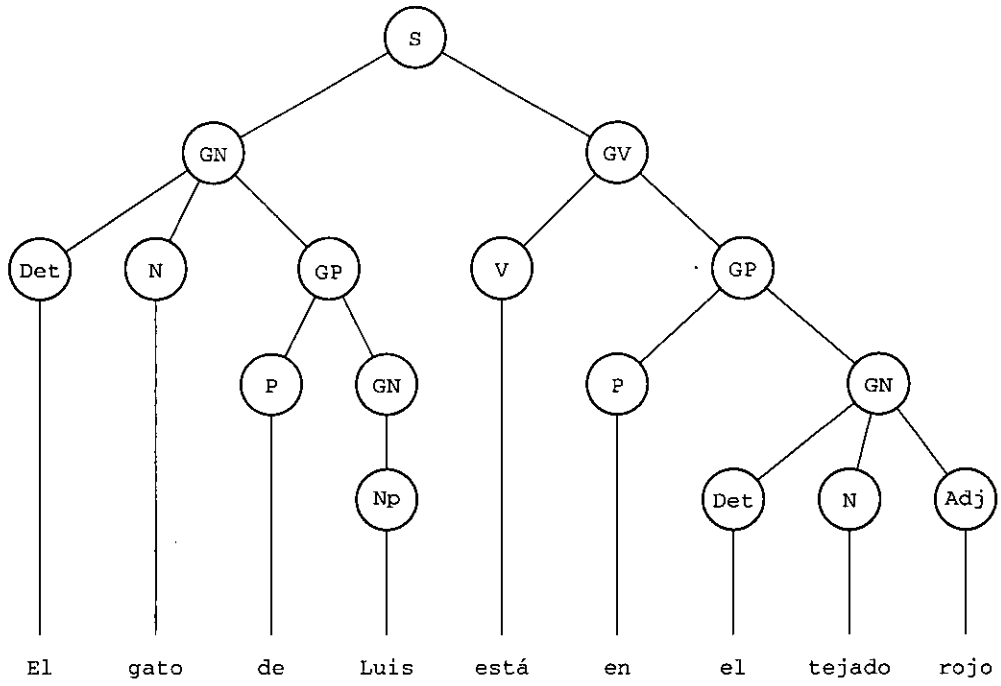


Figura 8.7: Árbol de análisis definitivo para la frase `El gato de Luis está en el tejado rojo`

las probabilidades de los ítems a partir de los cuales se genera, y la probabilidad de la regla involucrada en dicha generación.

De la misma forma, para los ítems de tipo 2 se puede definir la probabilidad de una producción p de un ítem $[\alpha \bullet \ldots; i, j]$ como el producto de las probabilidades de los ítems que la generan. En este caso no se multiplica por la probabilidad de ninguna regla, ya que los ítems de tipo 2 consideran interpretaciones parciales.

A partir de esto, es posible definir la probabilidad máxima para un ítem como el máximo de las probabilidades de cada una de las producciones, evitando así mantener las probabilidades de todas, y teniendo en cuenta que una producción de un ítem se obtiene combinando solamente dos ítems ya existentes, uno de tipo 2 con otro de tipo 1. Esto plantea la ventaja de que se pueden mantener los N análisis más probables durante la construcción de la tabla, ya que los N valores más grandes para un ítem están incluidos entre los N^2 productos de los N análisis más probables de cada uno de los dos ítems que intervienen en su generación. Si *a posteriori* estos N análisis más probables no son suficientes, se pueden extraer recursivamente todos los análisis o subanálisis de una frase de entrada, calculando sus probabilidades durante el proceso de extracción a partir de las probabilidades de sus constituyentes.

Además, la probabilidad de cualquier frase o de cualquier subsecuencia de la misma, definida como la suma de las probabilidades de todos sus análisis, se calcula también durante el proceso de construcción de la tabla de análisis, simplemente sumando siempre las probabilidades de cada nueva producción de un ítem, incluso aunque no vaya a ser mantenida entre las N mejores.

8.4.5 Palabras fuera de vocabulario

La principal restricción de las gramáticas estocásticas es que $\sum_\alpha P(A \rightarrow \alpha) = 1$. Sin embargo, para el procesamiento de palabras fuera de vocabulario esta restricción se puede relajar, permitiendo que $\sum_\alpha P(A \rightarrow \alpha) \leq 1$. La desigualdad estricta se cumple en las categorías abiertas a las que puedan pertenecer las palabras fuera de vocabulario. Por lo tanto, para cada una de estas categorías abiertas, se podría introducir una probabilidad distinta de cero, $P_u(A) = 1 - \sum_\alpha P(A \rightarrow \alpha)$, que puede ser interpretada como la probabilidad de que una palabra desconocida pudiera ser producida por la regla $A \rightarrow u$. El símbolo u sería un nuevo símbolo terminal para representar a cualquier palabra desconocida, evitando así un conjunto posiblemente infinito de reglas, para las posiblemente infinitas palabras desconocidas que pudieran aparecer. De esta forma, cuando el algoritmo se encuentra con una palabra desconocida en la frase de entrada, una alternativa es generar durante la fase de inicialización todos los no terminales de las categorías abiertas en la celda correspondiente, y considerar para los cálculos estas nuevas probabilidades $P_u(A)$.

No obstante, en la práctica, como ya se ha comentado, el tratamiento de las reglas léxicas, es decir, el reconocimiento de las palabras y la obtención de sus etiquetas y de sus probabilidades, suele aparecer asociado al uso de un diccionario y de un adivinador. Por tanto, una alternativa mucho mejor es dejar que sea el adivinador quien proponga, para cada palabra desconocida, tanto la lista de categorías candidatas, como sus correspondientes probabilidades. En la sección 4.4.3.5 se muestran algunas técnicas para realizar esta tarea.

8.4.6 Gramáticas con ciclos

Un problema que queda por solucionar es el tratamiento de gramáticas cíclicas. Dado que el analizador sintáctico es ascendente, los ciclos sólo pueden ser producidos por las reglas unitarias, esto es, por reglas del estilo $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow A$. En estos casos, el algoritmo de generación de la tabla de análisis, concretamente en la fase de autorrellenado, entraría en un lazo sin fin.

Este fenómeno se podría evitar comprobando que no se inserta una producción de un ítem ya existente, es decir, comprobando que no exista el mismo par ítem-producción en la lista, en cuyo caso se pasaría a procesar otro par de ítems. Con esto se consigue construir la tabla, pero seguiría existiendo el problema del lazo infinito a la hora de extraer los árboles de análisis, ya que un ciclo provoca realmente la existencia de un número infinito de árboles. Una vez más, el algoritmo resuelve este problema prohibiendo procesar cada producción más de una vez, generando así el número más pequeño posible de árboles correspondientes a ese número infinito[12].

8.4.7 Representación interna de las gramáticas

Existen diferentes mecanismos para la representación interna de las gramáticas, de manera que su posterior utilización por los algoritmos de análisis sintáctico sea eficiente. Uno de estos mecanismos se basa en expresar la gramática en forma de árbol. Para construir dicho árbol, se considera la parte derecha de cada regla de la gramática como una palabra que debe ser insertada en un árbol de letras. Cada símbolo que aparece en la regla constituye una letra de esa palabra ficticia.

La figura 8.8 muestra la representación arborescente de la gramática del ejemplo 8.5. En dicha figura no se incluyen las reglas léxicas, ya que en la práctica el tratamiento de este tipo de reglas suele aparecer asociado al uso de un diccionario.

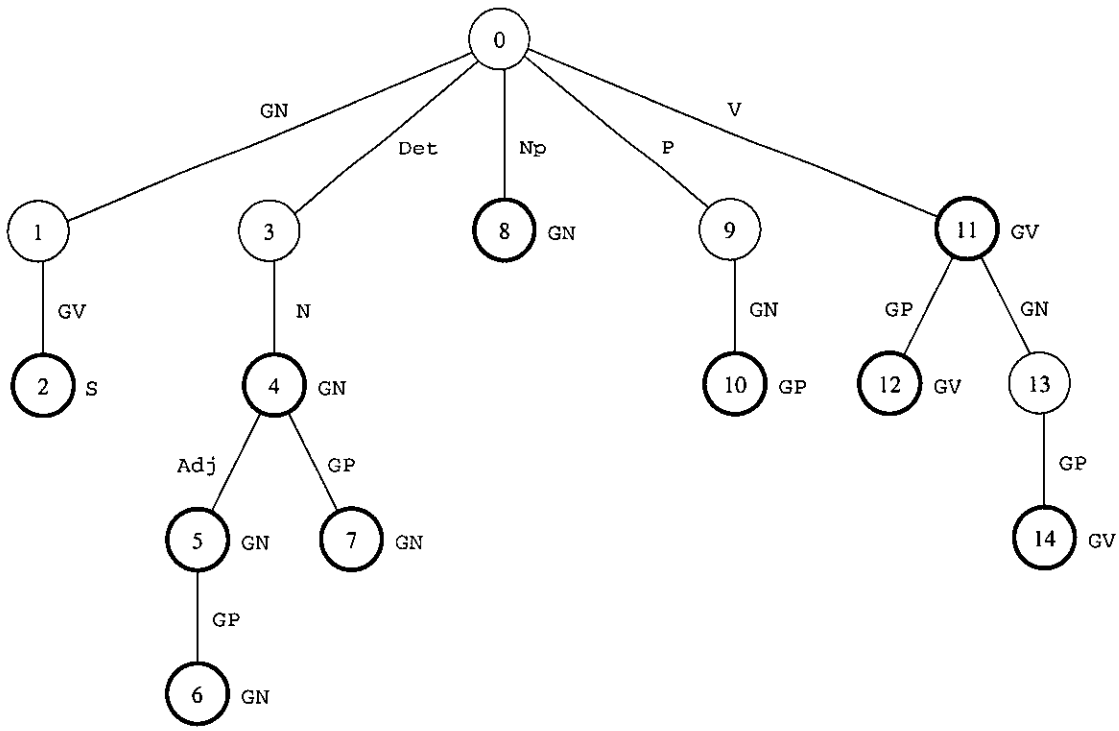


Figura 8.8: Representación arborescente de las reglas de una gramática

Mediante esta representación, la utilización de la gramática se acelera y el almacenamiento de los ítems es más compacto. La primera componente de un ítem de tipo 2, por ejemplo

[12]Sin embargo, la presencia de un ciclo implica un cambio de probabilidad cada vez que se pasa por él. Este fenómeno no afecta al cálculo de la probabilidad máxima, pero sí al de la probabilidad de la frase. Una solución es representar las probabilidades como si se tratara de expresiones regulares. Por ejemplo, $p + q^*$, donde q es la probabilidad del ciclo. En cualquier caso, la mejor alternativa es realizar *a priori* una detección y eliminación de ciclos, tal y como se discutió en la sección 2.3.2.

[V GN•...; i, j], se representa simplemente con el número de uno de los nodos de este árbol, en este caso [13; i, j]. Dicho nodo representa el estado concreto en el que se encuentra el reconocimiento parcial de una regla. Cuando se alcanza uno de los estados finales, marcados en la figura con círculos más gruesos, se completa una regla. Por tanto, cada estado final lleva asociado el símbolo no terminal que constituye la parte izquierda de la regla que se ha completado.

8.4.8 Consideraciones de paralelismo

La complejidad temporal del algoritmo CYK extendido, medida en términos de n, la longitud de la frase de entrada, es claramente proporcional a $\mathcal{O}(n^3)$. Sin embargo, dicha complejidad se puede rebajar, debido a la naturaleza intrínsecamente paralela de los algoritmos tipo CYK: existen muchas celdas de la tabla de análisis que pueden ser calculadas simultáneamente. El apéndice D resume las diferentes alternativas que se han considerado para la paralelización del algoritmo CYK extendido [Barcala y Graña 1999].

8.5 Aplicación del algoritmo CYK extendido al problema de la etiquetación

Al igual que los etiquetadores tienen que enfrentarse al problema de las palabras desconocidas, es decir, al problema de los diccionarios incompletos, los analizadores sintácticos deben saber enfrentarse al problema de las gramáticas incompletas. Cuando una frase es correcta, pero la gramática no es capaz de analizarla, todavía es posible considerar los subárboles correspondientes a los análisis parciales de fragmentos válidos de la frase. Cualquiera de las tareas que se realice posteriormente, a partir de estos subárboles, se engloba dentro de las denominadas técnicas de análisis sintáctico robusto. Una de esas posibles tareas puede ser, por ejemplo, la de completar la gramática, generando automáticamente las reglas sintácticas necesarias para analizar la frase. Éste es precisamente el objetivo más ambicioso del análisis sintáctico robusto.

En nuestro caso particular, resulta de especial interés la consideración de las etiquetas de las palabras de dichos subárboles como información adicional de apoyo para las técnicas tradicionales de etiquetación. Combinando esas subsecuencias de etiquetas, es posible generar varias etiquetaciones completas de la frase en cuestión, y posteriormente aplicar un filtro estadístico para elegir la secuencia global más probable. Las estrategias para realizar esas combinaciones se discutirán con detalle en el próximo capítulo.

Nuestro interés inmediato se centra en el hecho de que los algoritmos tipo CYK ofrecen un marco especialmente sencillo para la implementación de técnicas de análisis sintáctico robusto. Como ya hemos visto, la existencia de información en cualquier celda N_{ij} de la tabla de análisis implica la existencia de un árbol que cubre la subfrase $w_i, \ldots, w_{i+j-1}$. Por lo tanto, el objetivo es explotar esta predisposición natural y orientarla hacia el proceso de etiquetación.

Para ello, sin embargo, creemos que en este caso no resulta conveniente seguir los pasos clásicos de construir la tabla de análisis, extraer los subárboles y efectuar recorridos sobre ellos para obtener las secuencias de etiquetas. Determinadas características de las gramáticas de los lenguajes naturales, tales como las numerosas ambigüedades presentes en ellas, pueden provocar la existencia de gran cantidad de subárboles de análisis, pero podrían provocar también que muchos de ellos generasen las mismas secuencias de etiquetas. Como veremos a continuación, esto sugiere algunas reflexiones sobre el uso del algoritmo CYK extendido en este contexto. Dichas reflexiones están en la línea de no retrasar tanto el cálculo de las secuencias de etiquetas y hacer que el algoritmo las genere durante la construcción de la tabla.

Por tanto, a la hora de aplicar el algoritmo CYK extendido como ayuda al proceso de etiquetación es necesario realizar algunas consideraciones que permitan una optimización en el

procesamiento de los ítems. Estas consideraciones se basan en que, como hemos dicho, para el caso de la etiquetación no estamos interesados en los árboles o subárboles de análisis, sino en sus hojas, es decir, en las secuencias o subsecuencias de etiquetas. Por esta razón, es necesario realizar algunas modificaciones en el algoritmo:

- En este caso no resulta interesante mantener en la tabla de análisis las posibles producciones de un ítem. En lugar de esto, cada ítem llevará asociada una secuencia de etiquetas. En el caso de que sea un ítem de tipo 1, $[A; i, j]$, esa secuencia es la secuencia de etiquetas que se le asigna a la subfrase $w_i, \ldots, w_{i+j-1}$ en el subárbol de análisis encabezado por A, el símbolo no terminal asociado a ese ítem. Si por el contrario es un ítem de tipo 2, $[\alpha \bullet \ldots; i, j]$, esta secuencia es la secuencia de etiquetas de la subfrase que surge del análisis parcial α.

- De esta forma, se inserta un nuevo ítem en una lista junto con su secuencia de etiquetas, siempre y cuando no exista el mismo ítem, o bien, en el caso de que exista, siempre y cuando las secuencias de etiquetas que tiene asociadas sean diferentes.

Esta aproximación [Barcala y Graña 1999] consigue sintetizar aún más la información que se almacena en la tabla de análisis, ya que muchos de los subárboles darán lugar a las mismas secuencias de etiquetas, y mejora la extracción de dicha información para propósitos de análisis sintáctico robusto.

Con esto queda cumplido el objetivo que nos habíamos marcado hasta aquí: dejar preparado un marco de análisis sintáctico estocástico que permita experimentar cómodamente con técnicas de análisis sintáctico robusto orientadas específicamente al problema de la etiquetación.

Ejemplo 8.6 Finalmente, para ilustrar mejor el tipo de datos que deberemos manejar en el capítulo próximo, incluimos ahora, como ejemplo, la información que nos proporciona el analizador sintáctico al procesar 5 frases no completamente analizables de la parte con trazas del corpus SUSANNE (véase la sección 2.3.2):

1. `That <apos>s all I ask`
2. `Each is still glorified as a national hero`
3. `All his people ask for is no more war`
4. `The mystique of sex , combined with marijuana and jazz , is intended to provide a design for living`
5. `<ldquo> I <apos>d just turned on the ignition when there was a big flash and I was lying on the driveway <rdquo> , he said`

Para cada una de estas frases se incluyen dos figuras. La primera de ellas muestra el bosque de secuencias de etiquetas obtenidas. La última línea no sombreada no es información proporcionada por el analizador, sino que corresponde a la secuencia de etiquetas correctas tal y como figura en el corpus de referencia. Por razones de espacio, hemos utilizado los números de las etiquetas en lugar de sus nombres. La correspondencia entre números y nombres puede consultarse en la sección A.4.

La segunda figura es el árbol de análisis completo tal y como se indica en el corpus de referencia. Como se puede observar, todos estos árboles contienen trazas, de ahí que no hayan sido utilizados en el proceso de extracción de reglas para construir la gramática. Ésta es la razón por la cual todas estas frases no son completamente analizables. La notación utilizada para los símbolos no terminales de los nodos se puede consultar en el apéndice B. □

	1	2	3	4	5
palabras	That	's	all	I	ask
etqs (#etqs)	(3) 29 53 303	(6) 27 100 274 373 378 383	(4) 51 334 335 336	(4) 97 156 234 275	(1) 389
1	53	274	51	275	389
		373	334		
		378			
		383			
2		27	51		
		27	334		
				275	389
3	53	373	51		
Ref	53	373	51	275	389

Figura 8.9: Bosque de secuencias de etiquetas para la frase 1 del ejemplo 8.6 (frase de 5 palabras con un producto cruzado de 288 posibles etiquetaciones)

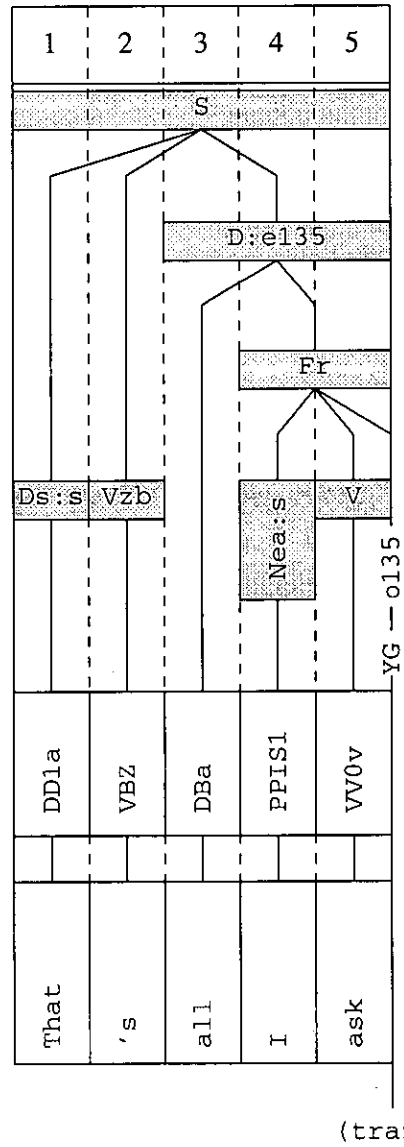


Figura 8.10: Árbol de referencia para la frase 1 del ejemplo 8.6

	1	2	3	4	5	6	7	8
palabras	Each	is	still	glorified	as	a	national	hero
etiquetas (#etiquetas)	(1)	(2)	(2)	(1)	(14)	(9)	(1)	(1)
	59	304	133	399	15	9	133	164
		373	334		17	54		
					23	64		
					24	68		
					26	70		
					27	86		
					30	91		
					36	96		
					107	425		
					108			
					116			
					315			
					335			
					373			
1	59	373	133	399	116	96	133	164
			334		373			
2			133	399			133	164
			334	399		9	133	
					116	96		
3	59	373	133			9	133	164
4	59	373	133	399				
					27	9	133	164
					116	9	133	164
5				399	116	9	133	164
				399	373	9	133	164
6			334	399	373	9	133	164
Ref	59	373	334	399	116	9	133	164

Figura 8.11: Bosque de secuencias de etiquetas para la frase 2 del ejemplo 8.6 (frase de 8 palabras con un producto cruzado de 504 posibles etiquetaciones)

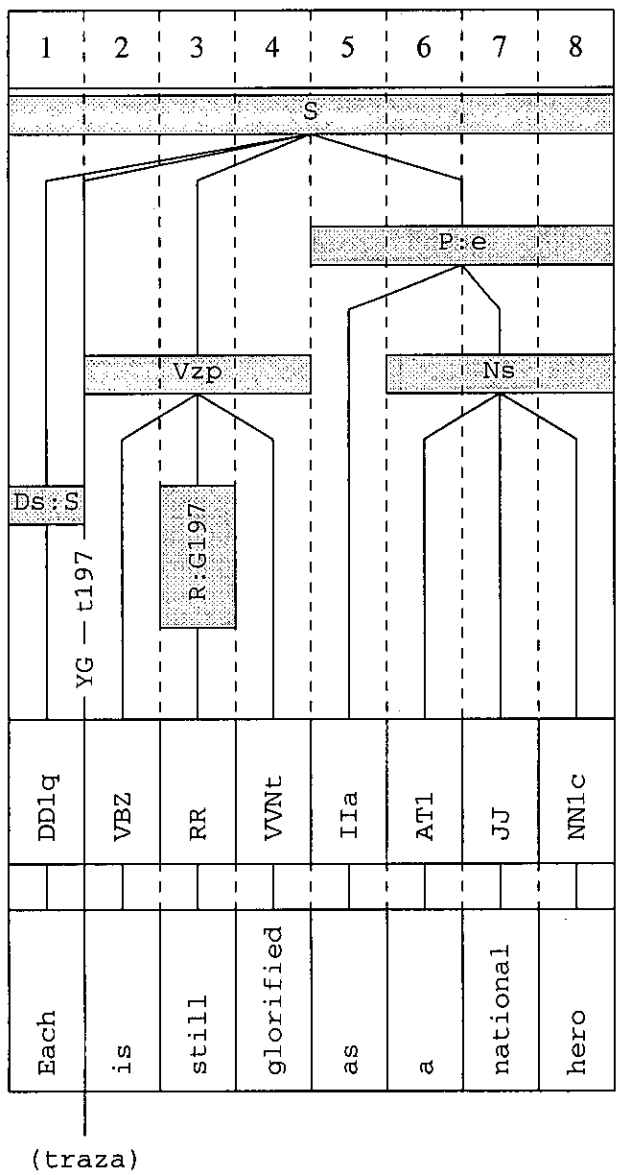


Figura 8.12: Árbol de referencia para la frase 2 del ejemplo 8.6

	1	2	3	4	5	6	7	8	9
palabras	All	his	people	ask	for	is	no	more	war
etqs (#etqs)	(4) 51 126 334 335	(2) 6 264	(1) 215	(1) 389	(5) 33 105 108 303 335	(2) 304 373	(5) 11 79 252 335 363	(3) 44 335 336	(1) 167
1	51 334	6 264	215	389		373	363	44	167
2		6	215		303	304	11 335	44 336	
3							11 335	44 336	167 167
4					303	304	335	336	
Ref	51	6	215	389	105	373	11	44	167

Figura 8.13: Bosque de secuencias de etiquetas para la frase 3 del ejemplo 8.6 (frase de 9 palabras con un producto cruzado de 1.200 posibles etiquetaciones)

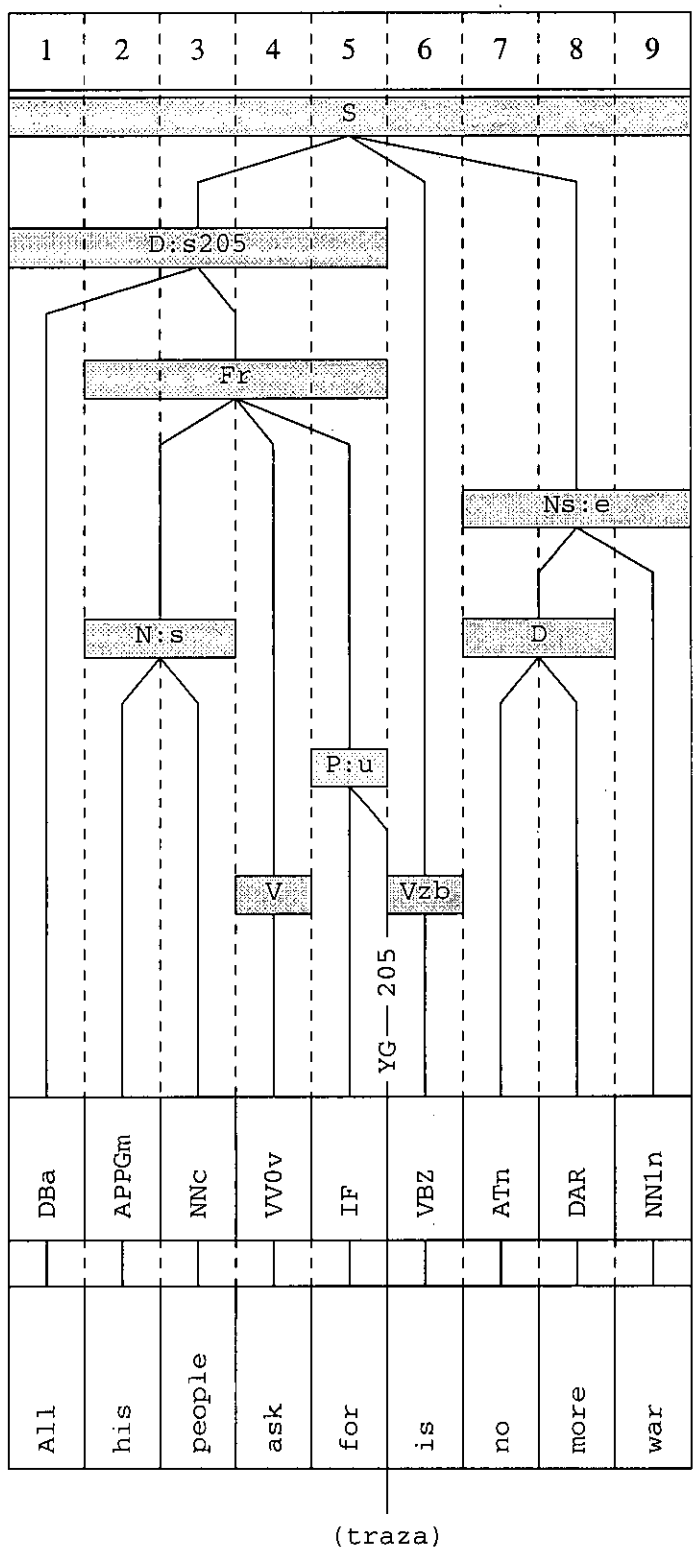


Figura 8.14: Árbol de referencia para la frase 3 del ejemplo 8.6

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
palabras	The	mystique	of	sex	,	combined	with	marijuana	and	jazz	,	is	intended	to	provide	a	design	for	living
etqs (#etqs)	(2)	(1)	(8)	(1)	(1)	(2)	(5)	(1)	(4)	(1)	(1)	(2)	(2)	(7)	(1)	(9)	(1)	(5)	(3)
	8	164	20	167	408	392	108	168	14	168	408	304	391	108	389	9	167	33	128
	66		108			400	109		61			373	399	111		54		105	167
			111				111		292					119		64		108	396
			115				122		338					320		68		303	
			121				124							325		70		335	
			323											342		86			
			335											362		91			
			336													96			
																425			
1		164		167		392		168	14	168		373	391	325	389	96	167		128
						400							399	362					167
																			396
2	8	164					122	168				373	399		389	96		105	167
			20	167					14	168			391	325				105	396
			121	167									391	362		9	167		
													399	325					
														362	389				
3		164	20	167		392	122	168					391	362	389		167	105	167
		164	121	167		400	122	168					399	362	389		167	105	396
				167	408	400		168	14	168				362	389	96			
															389	9	167		
4	8	164	20	167			122	168	14	168			391	362	389	96			
	8	164	121	167									399	362	389	96			
			121	167	408	400								362	389	9	167		
																9	167	105	167
																9	167	105	396
5		164	121	167	408	400							391	362	389	9	167		
				167	408	400	122	168					399	362	389	9	167		
						392	122	168	14	168					389	9	167	105	167
						400	122	168	14	168					389	9	167	105	396
6	8	164	121	167	408	400								362	389	9	167	105	167
			121	167	408	400	122	168						362	389	9	167	105	396
7		164	121	167	408	400	122	168					391	362	389	9	167	105	167
				167	408	400	122	168	14	168			391	362	389	9	167	105	396
													399	362	389	9	167	105	167
													399	362	389	9	167	105	396
8	8	164	121	167	408	400	122	168											
			121	167	408	400	122	168	14	168									
9		164	121	167	408	400	122	168	14	168									
10	8	164	121	167	408	400	122	168	14	168									
Ref	8	164	121	167	408	400	122	168	14	168	408	373	399	362	389	9	167	105	396

Figura 8.15: Bosque de secuencias de etiquetas para la frase 4 del ejemplo 8.6 (frase de 19 palabras con un producto cruzado de 2.419.200 posibles etiquetaciones)

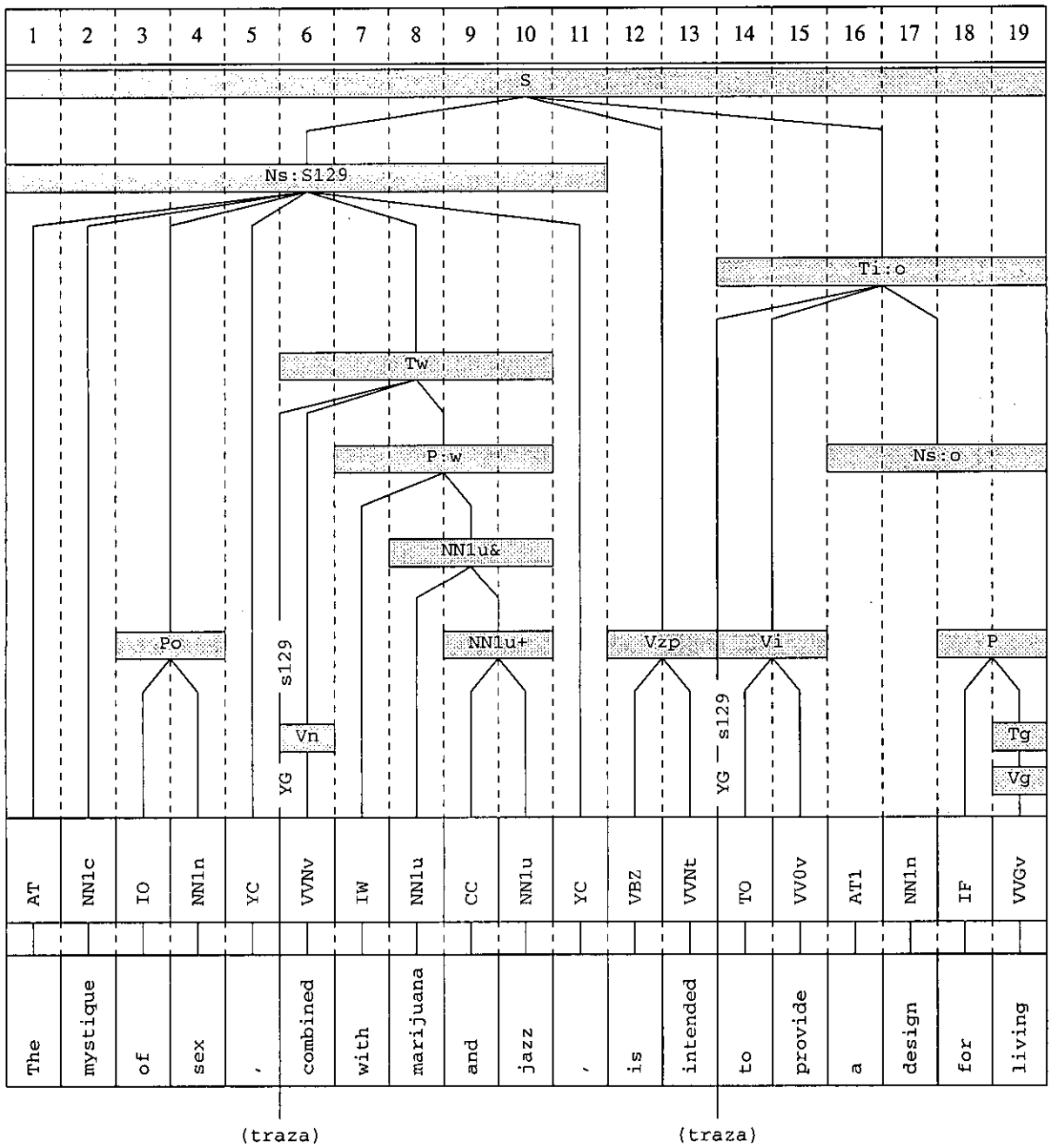


Figura 8.16: Árbol de referencia para la frase 4 del ejemplo 8.6

Figura 8.17: Bosque de secuencias de etiquetas para la frase 5 del ejemplo 8.6 (frase de 25 palabras con un producto cruzado de 49.766.400 posibles etiquetaciones)

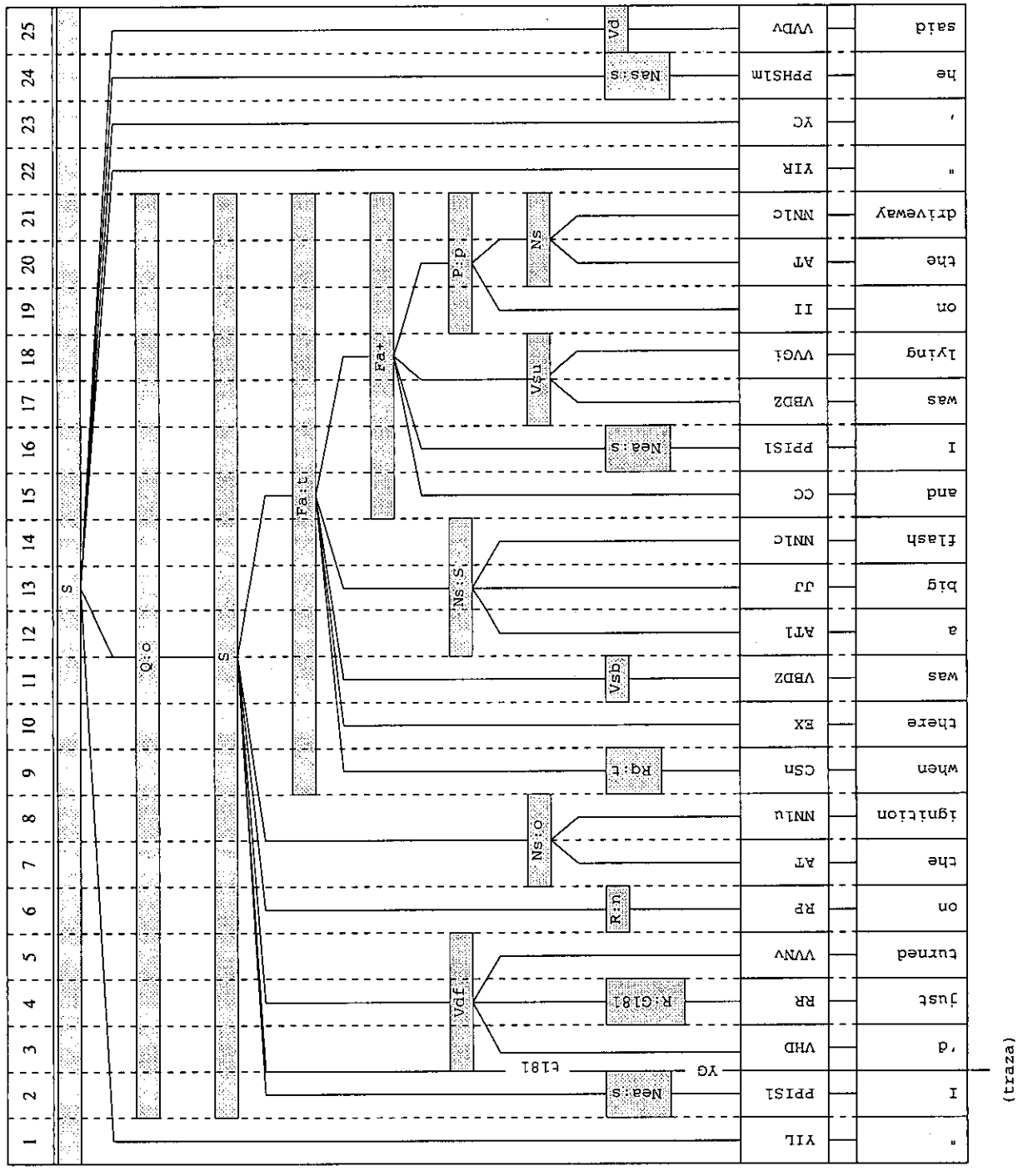


Figura 8.18: Árbol de referencia para la frase 5 del ejemplo 8.6

Realmente, el analizador sintáctico proporciona más información que la que muestran estas figuras, la cual no ha sido incluida también por razones de espacio. Además de los datos ya comentados, podemos manejar, para cada subsecuencia de etiquetas, el número de subárboles diferentes que la generan y la probabilidad del mejor de esos subárboles. Por ejemplo, para la frase 4 (figura 8.15), existen 4 secuencias diferentes de longitud 7 comenzando en la posición 13. Para tener una mejor idea del volumen de los datos manejados, mostramos a continuación el número de árboles diferentes que generan cada una de esas secuencias, la probabilidad logarítmica del mejor de ellos y las etiquetas que forman dichas secuencias:

```
126 (-48,087340) 391 362 389 9 167 105 167
 38 (-45,854593) 391 362 389 9 167 105 396
 16 (-42,822315) 399 362 389 9 167 105 167
  5 (-40,589568) 399 362 389 9 167 105 396
```

Por tanto, existen $126 + 38 + 16 + 5 = 185$ árboles diferentes para cubrir estas 7 palabras, pero todos ellos generan tan solo 4 etiquetaciones posibles. Otro dato interesante es que, a pesar de que existen 126 árboles que generan la primera secuencia de etiquetas, el más probable de los 185 es uno de los 5 que generan la última secuencia.

Capítulo 9

Estrategias de análisis sintáctico robusto para la etiquetación

Una característica inherente a los sistemas de etiquetación puramente estocásticos es que etiquetan erróneamente alrededor del 2 ó 3% de las palabras, tal y como hemos visto en los capítulos anteriores. La manera de enfrentarse con garantías a este pequeño porcentaje de errores pasa inevitablemente por incorporar alguna fuente de información de más alto nivel, como puede ser una gramática. Por esta razón, pensamos que el uso combinado de los sistemas de etiquetación tradicionales con información sintáctica procedente de una gramática puede producir una importante mejora en el proceso de etiquetación de textos en lenguaje natural, y éste es precisamente el objetivo que se aborda en el presente capítulo.

Por supuesto, resulta muy difícil disponer de una gramática que genere todas y cada una de las estructuras que aparecen en las frases de un idioma dado. Aún así, cuando una frase es correcta, pero la gramática no es capaz de analizarla, todavía es posible considerar los subárboles correspondientes a los análisis parciales de fragmentos válidos de la frase. El posterior estudio de estos subárboles puede ser utilizado, por ejemplo, para completar la gramática, generando automáticamente las reglas sintácticas necesarias para analizar la frase. Éste es precisamente el objetivo más ambicioso del análisis sintáctico robusto. En nuestro caso particular, resulta de especial interés la consideración de las etiquetas de las palabras de dichos subárboles como información adicional de apoyo para las técnicas tradicionales de etiquetación. La estrategia consiste en combinar esas subsecuencias de etiquetas para generar varias etiquetaciones completas posibles de la frase en cuestión, y posteriormente aplicar un filtro estadístico para elegir la secuencia global más probable.

Para poder experimentar con esta nueva técnica necesitamos un analizador sintáctico y un corpus de referencia. En relación con el primero de estos recursos, utilizaremos el analizador sintáctico descrito en el capítulo anterior. Se trata de un analizador ascendente para gramáticas independientes del contexto estocásticas que, además de implementar las funcionalidades usuales de este tipo de analizadores, es capaz también de obtener los N análisis más probables de la frase de entrada o de cualquier subsecuencia de la misma, de manejar las múltiples interpretaciones de frases que contienen palabras compuestas y de enfrentarse a las palabras desconocidas. También es posible la extracción explícita de los árboles de análisis y de las secuencias de etiquetas asociadas a la frase de entrada o a cualquier subfrase de ella. En relación con el segundo recurso, utilizaremos el corpus SUSANNE[1].

Como ya hemos esbozado, la idea general consiste entonces en dividir el corpus de referencia en dos partes, extraer una gramática de la primera parte, analizar la segunda, y recuperar

[1] El banco de árboles utilizado como corpus de referencia para el idioma inglés (véase la sección 2.3).

las subsecuencias de etiquetas de todos los subárboles de aquellas frases no completamente analizables. Posteriormente, se diseñan estrategias similares a las utilizadas en el análisis sintáctico robusto que combinen dichas subsecuencias para construir secuencias de cobertura total, y se reetiquetan las frases no analizables aplicando un filtro estadístico sobre dichas secuencias. Por último, se evalúa el rendimiento de estas nuevas técnicas de etiquetación y se compara con el de las técnicas tradicionales. A continuación presentamos en detalle todos estos pasos.

9.1 Cobertura de la gramática SUSANNE

El criterio de partición del corpus SUSANNE, como ya hemos visto en la sección 2.3.2, no ha sido la extracción aleatoria de frases. En este caso, la gramática se ha extraído a partir de las frases sin trazas (4.292 frases y 77.275 palabras, que dan lugar a una gramática de 17.669 reglas y 1.525 símbolos no terminales), y se ha reservado la parte de frases con trazas (2.039 frases y 51.426 palabras) para realizar los experimentos relativos a las nuevas técnicas de etiquetación. Tras obtener la gramática, resulta muy interesante analizar su cobertura sobre ambas partes del corpus.

9.1.1 Cobertura sobre las frases sin trazas

Por supuesto, todas las frases sin trazas son analizables, ya que todas las reglas involucradas en los árboles del corpus de referencia están presentes en la gramática SUSANNE. No obstante, pueden surgir otras interpretaciones diferentes, es decir, el árbol de referencia podría no ser el único árbol posible para cada frase. Por tanto, es importante responder a las dos cuestiones siguientes:

1. ¿Cuál es el número de ambigüedades generado por la gramática para cada frase de la parte sin trazas?

2. Si consideramos el conjunto de todos los árboles que proporciona el analizador para cada frase, ordenados del más probable al menos probable, ¿cuál es la posición del árbol de referencia dentro de ese conjunto?

En relación al número de ambigüedades, los datos concretos aparecen en la sección E.1 y se resumen gráficamente en la figura 9.1. La característica más relevante de los valores de esta distribución es que la mediana se sitúa entre 7 y 8 interpretaciones por frase. Con respecto a la posición del árbol de referencia, los datos concretos aparecen en la sección E.2 y se resumen gráficamente en la figura 9.2. La característica más relevante es que el análisis de máxima probabilidad coincide con el árbol de referencia para el 77,70% de las frases. Este valor es superior a los porcentajes de acierto que se suelen obtener con otros bancos de árboles, los cuales se sitúan entre el 60 y el 65%.

Los datos que hemos obtenido, un número bajo de ambigüedades y un porcentaje de coincidencia alto con el análisis de referencia, son los principales parámetros que definen la calidad de un banco de árboles y de la correspondiente gramática que se puede extraer de un recurso lingüístico de este tipo. En nuestro caso podemos afirmar que el corpus SUSANNE presenta un nivel de diseño más que aceptable, y que tanto el proceso de extracción de la gramática como el modelo de análisis sintáctico estocástico que hemos aplicado a este corpus funcionan correctamente.

Sin embargo, nuestro interés inmediato se centra en el proceso de etiquetación y queremos saber si, en general, la información sintáctica puede ser útil para este proceso o no. Por tanto,

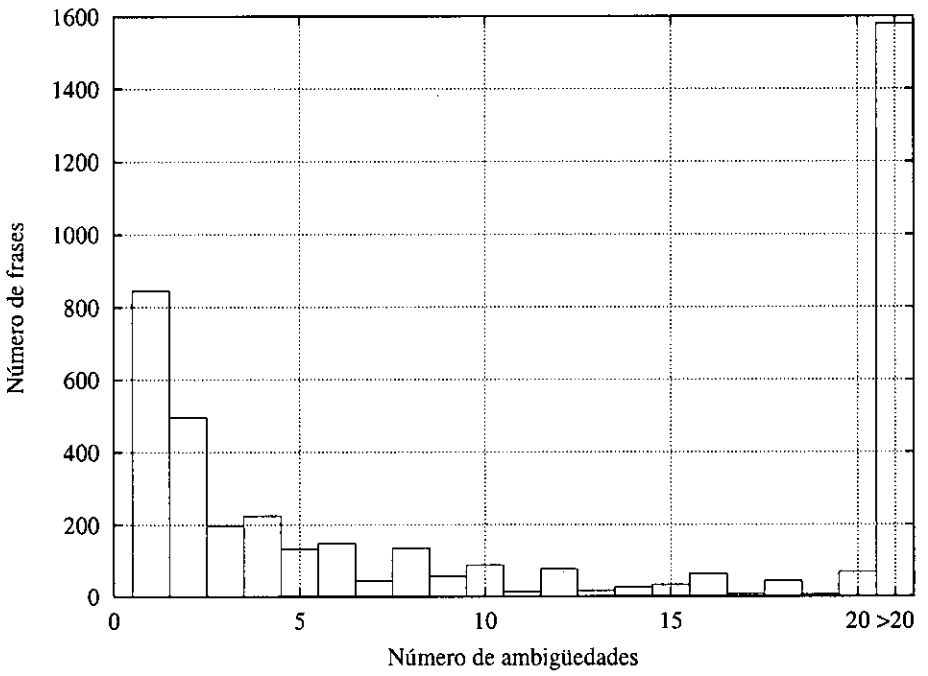


Figura 9.1: Ambigüedades de la gramática SUSANNE sobre las frases sin trazas

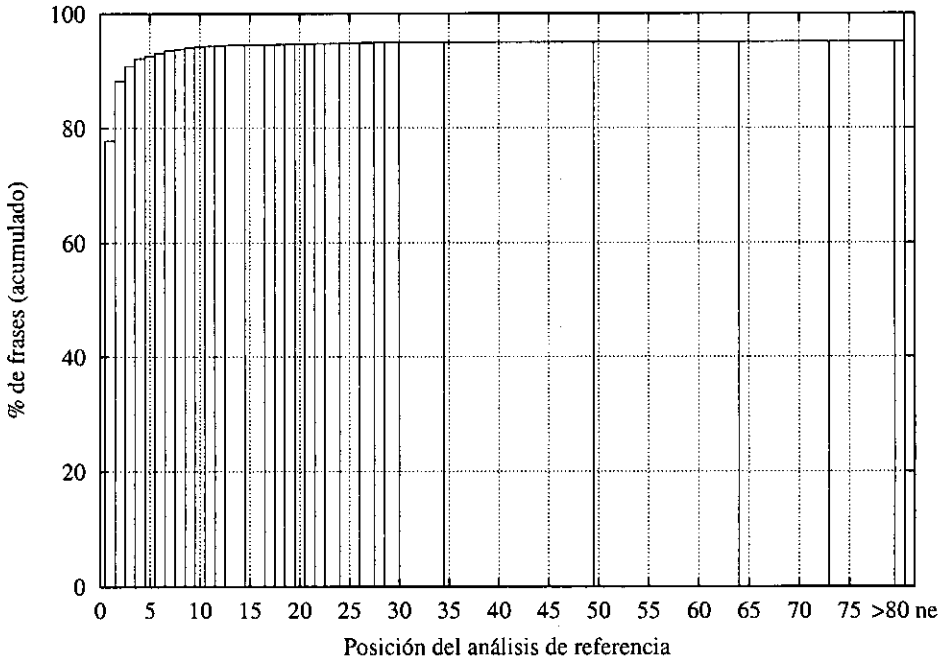


Figura 9.2: Nivel de acierto de la gramática SUSANNE sobre las frases sin trazas

hemos realizado un test más con la gramática SUSANNE sobre la parte de frases sin trazas. Para cada una de estas frases, consideramos sólo el árbol de análisis más probable de entre todos los que obtiene el analizador, incluso aunque sea diferente del árbol de referencia. Posteriormente, tomamos las etiquetas que dicho árbol más probable asigna a las palabras de la frase, y las comparamos con las etiquetas del árbol de referencia. Al mismo tiempo, para poder tener una idea clara sobre el resultado obtenido, lo hemos comparado también con algunas de las

herramientas representativas de los principales modelos de etiquetación tradicionales. En este caso, hemos entrenado el sistema BRILL [Brill 1993b] y el sistema TNT [Brants 1996] sobre la parte sin trazas del corpus SUSANNE, y hemos reetiquetado esa misma parte del corpus. En principio, estas deberían ser las mejores condiciones de trabajo para estos modelos: no tener que enfrentarse a palabras desconocidas, y reetiquetar el mismo corpus de entrenamiento a partir del cual se han extraído los parámetros de funcionamiento. La tabla 9.1 muestra el rendimiento para cada una de las tres situaciones mencionadas, utilizando la misma metodología de evaluación descrita en la sección 7.2.

	Sistema BRILL		Sistema TNT		Gramática SUSANNE	
OOVF+	0	-	0	-	0	-
OOVF-	0	-	0	-	0	-
NAF+	40.945	100%	40.945	100%	40.945	100%
NAF-	0	0%	0	0%	0	0%
AF+	32.853	90,43%	35.974	99,02%	36.034	99,19%
AF-	3.477	9,57%	356	0,98%	296	0,81%
S1	95,500		99,539		99,616	
S2	90,429		99,020		99,185	

Tabla 9.1: Sistemas BRILL y TNT frente a la gramática SUSANNE etiquetando la parte sin trazas del corpus SUSANNE

En dicha tabla podemos observar que la gramática SUSANNE, trabajando como si de un etiquetador se tratara, presenta valores muy próximos al 100% de éxito tanto en el índice de precisión como en el de decisión. Estos resultados aparecen seguidos muy de cerca por los del sistema TNT, que vuelve a poner de manifiesto la especial adecuación del marco probabilístico para las tareas de etiquetación. En cualquier caso, la realización de este test es de gran importancia, ya que nos permite reforzar la idea de que la información sintáctica proporcionada por un gramática puede jugar un papel muy relevante en el proceso de etiquetación de textos en lenguaje natural, tal y como trataremos de demostrar en las siguientes secciones.

9.1.2 Cobertura sobre las frases con trazas

Por último, sólo nos resta conocer qué es lo que ocurre al analizar con la gramática SUSANNE la parte de frases con trazas. Las cifras sobre ambigüedades y porcentajes de acierto pueden verse en las secciones E.3 y E.4, respectivamente. La característica más relevante es que sólo 60 de las 2.039 frases de la parte con trazas pueden ser analizadas con la gramática SUSANNE. Esto implica una cobertura pobre de la gramática sobre esta parte del corpus, pero al mismo tiempo nos encontramos frente a un buen conjunto de datos, formado por las restantes 1.979 frases, sobre el que podemos realizar todo tipo de experimentos de análisis sintáctico robusto.

A este respecto, es importante recordar que nuestro objetivo último es la etiquetación, y que nuestro analizador sintáctico, para cada frase no completamente analizable, es capaz de obtener todas las subsecuencias de etiquetas correspondientes al bosque de subárboles parciales. Para cada subsecuencia, podemos conocer también el número de subárboles distintos que cubren la misma porción de frase con las mismas etiquetas, y la probabilidad del subárbol más plausible, tal y como hemos visto en el ejemplo 8.6. Toda esta información constituye la materia prima de trabajo de las técnicas de etiquetación dirigidas por la sintáxis que vamos a proponer a continuación. La siguiente sección se ocupa del diseño de estrategias para la adecuada combinación de las restricciones sintácticas impuestas por estas subsecuencias de etiquetas.

9.2 Estrategias para la combinación de las subsecuencias de etiquetas

Consideremos por un momento el cardinal del producto cruzado de las etiquetas de las palabras de una frase dada. Esta cifra representa el número total de posibles etiquetaciones para dicha frase. Como se puede ver en los datos que aparecen en la sección E.5, resumidos gráficamente en la figura 9.3, este valor puede llegar a ser muy alto a lo largo de las frases de la parte con trazas del corpus SUSANNE. El número medio de palabras por frase es 25, y la mediana es 23. El número medio de posibles etiquetaciones por frase es 10^{18}, y la mediana es 10^{6}.

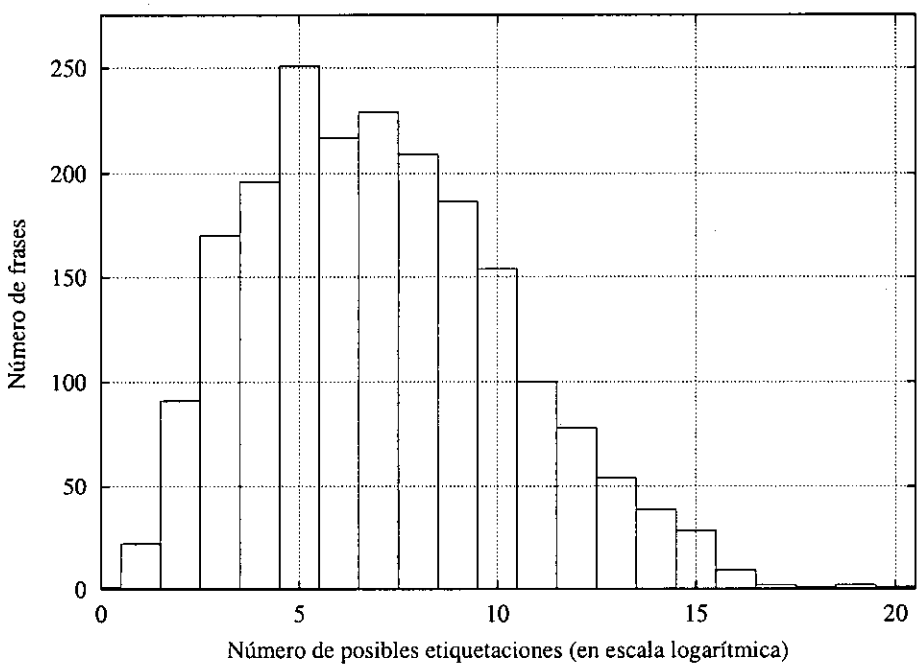


Figura 9.3: Producto cruzado de etiquetas sobre la parte de frases con trazas del corpus SUSANNE

Al estudiar las técnicas de etiquetación tradicionales, hemos visto que existen algoritmos capaces de manejar estos números tan elevados de posibles etiquetaciones y de obtener de manera muy eficiente la secuencia de etiquetas más probable para una frase dada. El algoritmo de Viterbi constituye el principal ejemplo de este tipo de algoritmos, cuyo principio de funcionamiento radica en la aplicación de técnicas de programación dinámica [Viterbi 1967, Forney 1973]. Sin embargo, la secuencia de etiquetas más probable para una frase dada no siempre coincide con la secuencia involucrada en la interpretación sintáctica de dicha frase. Por otra parte, la integración de algún tipo de restricción sintáctica que vaya más allá de la simple visión lineal de las etiquetas de las palabras resulta imposible en estos métodos. Por tanto, nuestro interés inmediato se centra ahora en el uso de las subsecuencias parciales de etiquetas que se pueden obtener mediante un análisis sintáctico de la frase.

El objetivo es combinar dichas subsecuencias para reducir drásticamente el producto cruzado de etiquetas a un número mucho menor, por ejemplo entre 20 y 50 posibles interpretaciones completas para cada frase. Como se puede observar, lo que perseguimos con esto no es una mejora en el tiempo de cálculo, sino un filtrado que nos permita trabajar sólo con las interpretaciones que son sintácticamente más plausibles, en lugar de evaluarlas todas. Después de esta reducción, la probabilidad de cada secuencia global elegida se puede calcular mediante los algoritmos 4.1 o 4.2, algoritmos de cálculo hacia adelante y hacia atrás, respectivamente, aplicados sobre un

HMM entrenado previamente con la parte de frases sin trazas del corpus SUSANNE. De esta manera, se puede establecer un orden de probabilidad sobre la lista de secuencias globales que han pasado el filtro, y elegir una o varias de entre las más probables.

Para realizar esta reducción, hemos implementado dos estrategias diferentes de combinación de las subsecuencias de etiquetación parcial [Graña *et al.* 1999]. Ambas se describen detalladamente en las siguientes secciones.

9.2.1 Estrategia 1: elegir la secuencia más larga y más probable

Para una frase dada, la primera estrategia diseñada comienza a recorrer la tabla de análisis CYK desde la cima hacia la base, hasta encontrar la subsecuencia de etiquetas más larga. Posteriormente, rellena los huecos de manera recursiva utilizando el mismo criterio en cada subtabla, hasta que todas las palabras de la frase hayan sido cubiertas, tal y como se muestra en el siguiente algoritmo.

Algoritmo 9.1 Pseudo-código de la estrategia 1 para la combinación de subsecuencias de etiquetación parcial:

```
function Obtener_Subsecuencia (Tabla, Comienzo, Longitud) =
  begin
    if (Longitud = 0) then
      return secuencia vacía;

    A ← Mejor_Árbol (Tabla, Comienzo, Longitud);
    i ← punto de comienzo del árbol A;
    j ← número de palabras cubiertas por el árbol A;
    S ← secuencia de etiquetas de las hojas del árbol A;
    if (i = Comienzo) and (j = Longitud) then
      return S;

    Izq ← Obtener_Subsecuencia (Tabla, Comienzo, i − Comienzo);
    Dch ← Obtener_Subsecuencia (Tabla, i + j, Longitud + Comienzo − i − j);
    return Izq, S, Dch
  end;

function Estrategia_1 (Frase, Gramática) =
  begin
    n ← número de palabras en Frase;
    T ← tabla CYK obtenida al analizar Frase con Gramática;
    S ← Obtener_Subsecuencia (T, 1, n);
    return S
  end;
```

En busca de un conjunto de secuencias de cobertura total para la frase, la función *Estrategia_1* invoca con las coordenadas de la celda situada en la cima de la tabla de análisis a la función *Obtener_Subsecuencia*. Dentro de ella, la función *Mejor_Árbol* es precisamente la encargada de buscar el árbol de análisis con la secuencia de hojas más larga en la subtabla de *Tabla* que comienza en la posición *Comienzo* y tiene longitud *Longitud*. Si encuentra varios árboles con secuencias de igual longitud, elige siempre áquel que tenga la probabilidad más alta. □

Sin embargo, el comportamiento totalmente determinista de la función *Mejor_Árbol* hace que el método produzca una única etiquetación completa para cada frase, tal y como podemos observar en el siguiente ejemplo.

Ejemplo 9.1 A continuación mostramos las secuencias de etiquetas que genera este método de combinación para cada una de las frases del ejemplo 8.6, junto con la correspondiente secuencia correcta presente en el corpus de referencia:

```
Frase 1: 5 palabras, 1 secuencia, 0 errores.
 1x 53 373 51 275 389 (0)
Ref 53 373 51 275 389

Frase 2: 8 palabras, 1 secuencia, 1 error.
 1x 59 373 334 399 373 9 133 164 (1)
Ref 59 373 334 399 116 9 133 164

Frase 3: 9 palabras, 1 secuencia, 5 errores.
 1x 126 6 215 389 303 304 335 336 167 (5)
Ref  51 6 215 389 105 373  11  44 167

Frase 4: 19 palabras, 1 secuencia, 0 errores.
 1x 8 164 121 167 408 400 122 168 14 168 408 373 399 362 389 9 167 105 396 (0)
Ref 8 164 121 167 408 400 122 168 14 168 408 373 399 362 389 9 167 105 396

Frase 5: 25 palabras, 1 secuencia, 2 errores.
 1x ... 8 168 38 89 368 9 133 164 292 275 368 394 106 8 164 415 408 271 392 (2)
Ref ... 8 168 38 89 368 9 133 164  14 275 368 394 106 8 164 415 408 271 392
```

Para cada frase, indicamos el número de palabras, el número de posibles secuencias globales de etiquetación (siempre 1 con esta estrategia), y el número de errores de la mejor de ellas. Para cada secuencia, la primera cifra seguida inmediatamente por una x representa el número de veces que el proceso de combinación genera esa secuencia (siempre 1 con esta estrategia), posteriormente aparecen los números correspondientes a las etiquetas, y la última cifra entre paréntesis representa el número de errores encontrados al comparar con la referencia. □

La tabla 9.2 muestra los resultados de la evaluación de esta primera estrategia de combinación sobre todas las frases de la parte con trazas del corpus SUSANNE. Si comparamos estos resultados con los de las tablas 7.19, 7.20, 7.21 y 7.22, podemos observar que, salvo en el caso del sistema JMX, el rendimiento global de la primera estrategia es, en general, peor que el de los etiquetadores tradicionales.

	Estrategia 1	
OOVF+	2.440	56,46%
OOVF-	1.882	43,54%
NAF+	23.576	97,93%
NAF-	498	2,07%
AF+	20.394	88,55%
AF-	2.636	11,46%
S1	90,246	
S2	83,482	

Tabla 9.2: Resultados de etiquetación de la Estrategia 1 sobre la parte con trazas del corpus SUSANNE

Una razón que podría explicar este mal rendimiento es que hemos implementado una reducción demasiado drástica. Para cada frase, partimos de un número muy alto de posibles etiquetaciones y nos quedamos con sólo una. Sin embargo, la principal idea que subyace bajo este primer experimento es que quizás los subárboles más largos o de mayor cobertura imponen unas restricciones de análisis demasiado fuertes. Como consecuencia, no siempre son los mejores candidatos a elegir a la hora de combinar sus correspondientes subsecuencias de etiquetas en busca de la correcta interpretación global de una frase. Por tanto, para diseñar una técnica de etiquetación que ofrezca mejores resultados, se hace necesaria la obtención de un punto de partida menos determinista, como el que utiliza la estrategia que presentamos a continuación.

9.2.2 Estrategia 2: aplicar la estrategia 1 una vez en cada celda

Debido a las razones expuestas anteriormente, hemos implementado una segunda estrategia de combinación. Igual que antes, trabajaremos con la información que reside en la tabla de análisis CYK, y construiremos recursivamente varias secuencias de etiquetas. Una vez más, el criterio de combinación elige la subsecuencia de etiquetas más larga y más probable. Sin embargo, en lugar de aplicarlo sólo en la cima de la tabla, este procedimiento se ejecuta una vez en cada una de las celdas.

Algoritmo 9.2 Pseudo-código de la estrategia 2 para la combinación de subsecuencias de etiquetación parcial:

```
function Estrategia_2 (Frase, Gramática) =
  begin
    n ← número de palabras en Frase;
    T ← tabla CYK obtenida al analizar Frase con Gramática;
    C ← ∅;

    for j ← n downto 1 do
      for i ← 1 to (n − j + 1) do
        begin
          Izq ← Obtener_Subsecuencia (T, 1, i − 1);
          S ← Obtener_Subsecuencia (T, i, j);
          Dch ← Obtener_Subsecuencia (T, i + j, 1 + n − i − j);
          Añadir la secuencia Izq, S, Dch al conjunto C
        end;

    return C
  end;
```

La función *Obtener_Subsecuencia* es idéntica a la del algoritmo 9.1. □

Si preguntamos cuántas etiquetaciones completas genera este método para cada frase, la respuesta podría ser tantas como celdas tenga la tabla de análisis. Sin embargo, hemos visto que la presencia de un determinado número de subárboles en una celda no implica la existencia de ese mismo número de subsecuencias de etiquetas distintas en ese punto, ya que típicamente habrá muchas subsecuencias iguales compartidas por varios de esos árboles. Un fenómeno parecido ocurre también aquí, y se puede observar que esta estrategia también producirá secuencias globales repetidas. Por tanto, el método de combinación genera entre 1 y como máximo

$$\frac{n \times (n+1)}{2}$$

etiquetaciones completas para una frase dada, donde n es el número de palabras de dicha frase.

Ejemplo 9.2 A continuación mostramos las secuencias de etiquetas que genera este segundo método de combinación para cada una de las frases del ejemplo 8.6:

```
Frase 1: 5 palabras, 8 secuencias, 0 errores.
 2x 53 100  51 275 389 (1)
 2x 53  27 334 275 389 (2)
 4x 53 373  51 275 389 (0)
Ref 53 373  51 275 389

Frase 2: 8 palabras, 18 secuencias, 0 errores.
 5x 59 373 133 399 116  9 133 164 (1)
 3x 59 373 133 399 116 96 133 164 (2)
 1x 59 373 133 399  15 54 133 164 (3)
 4x 59 373 133 399  15  9 133 164 (2)
 2x 59 373 334 399 116  9 133 164 (0)
 3x 59 373 334 399 373  9 133 164 (1)
Ref 59 373 334 399 116  9 133 164

Frase 3: 9 palabras, 14 secuencias, 1 error.
 2x 126 264 215 389 303 304 335 336 167 (6)
 2x 126   6 215 389 105 373  11  44 167 (1)
 3x 126   6 215 389 303 304  11  44 167 (3)
 5x 126   6 215 389 303 304 335 336 167 (5)
 2x 126   6 215 389 303 304  79  44 167 (4)
Ref  51   6 215 389 105 373  11  44 167

Frase 4: 19 palabras, 58 secuencias, 0 errores.
 6x 66 164 121 167 408 400 122 168  14 168 408 373 399 362 389  9 167 105 396 (1)
 5x  8 164 115 167 408 400 122 168  14 168 408 373 399 362 389  9 167 105 396 (1)
 1x  8 164 121 167 408 400 122 168  14 168 408 373 391 362 389  9 167 105 396 (1)
 1x  8 164 121 167 408 400 122 168  14 168 408 373 399 325 389  9 167 105 396 (1)
 4x  8 164 121 167 408 400 122 168  14 168 408 373 399 342 389  9 167 105 396 (1)
 1x  8 164 121 167 408 400 122 168  14 168 408 373 399 342 389 96 167 105 396 (2)
 1x  8 164 121 167 408 400 122 168  14 168 408 373 399 362 389 54 167 105 396 (1)
 2x  8 164 121 167 408 400 122 168  14 168 408 373 399 362 389  9 167 105 128 (1)
28x  8 164 121 167 408 400 122 168  14 168 408 373 399 362 389  9 167 105 396 (0)
 4x  8 164 121 167 408 400 122 168  14 168 408 373 399 362 389 96 167 105 396 (1)
 2x  8 164 121 167 408 400 122 168 292 168 408 373 399 362 389  9 167 105 396 (1)
 3x  8 164 121 167 408 400 124 168  14 168 408 373 399 362 389  9 167 105 396 (1)
Ref  8 164 121 167 408 400 122 168  14 168 408 373 399 362 389  9 167 105 396

Frase 5: 25 palabras, 54 secuencias, 1 error.
 3x ...   8 168 38 89 368  9 133 164 292 275 368 394 106   8 164 415 408 271 392 (4)
 2x ...   8 168 38 89 368  9 133 164 292 275 368 394 106   8 164 415 408 271 392 (3)
 2x ... 113 168 38 89 368  9 133 164 292 275 368 394 106   8 164 415 408 271 392 (4)
 2x ...   8 168 38 89 368  9 133 164 292 275 368 394 106   8 164 415 408 271 392 (3)
 1x ...   8 168 38 89 368  9 133 164 292 275 368 394 106   8 164 415 408 271 392 (2)
 2x ...   8 168 38 89 368  9 133 164 292 275 368 394 106   8 164 415 408 271 392 (3)
 1x ...   8 168 38 89 368 54 133 164 292 275 368 394 106   8 164 415 408 271 392 (3)
 8x ...   8 168 38 89 368  9 133 164  14 275 368 394 106   8 164 415 408 271 392 (1)
 1x ...   8 168 38 89 368  9 133 164  14 275 368 394 112   8 164 415 408 271 392 (2)
 1x ...   8 168 38 89 368  9 133 164  14 275 368 394 332   8 164 415 408 271 392 (2)
 3x ...   8 168 38 89 368  9 133 164 292 275 368 394 106 113 164 415 408 271 392 (3)
24x ...   8 168 38 89 368  9 133 164 292 275 368 394 106   8 164 415 408 271 392 (2)
```

```
4x ...   8 168 38 89 368 96 133 164 292 275 368 394 106   8 164 415 408 271 392 (3)
Ref ...  8 168 38 89 368  9 133 164  14 275 368 394 106   8 164 415 408 271 392
```

Como antes, la primera cifra seguida inmediatamente por una x representa el número de veces que se genera esa secuencia, y la última cifra entre paréntesis representa el número de errores al comparar con la referencia. □

La tabla 9.3 muestra los resultados de la evaluación de esta segunda estrategia de combinación sobre todas las frases de la parte con trazas del corpus SUSANNE. Comparando de nuevo estos resultados con los de las tablas 7.19, 7.20, 7.21 y 7.22, podemos observar que en esta ocasión el rendimiento global de la segunda estrategia es ligeramente superior al de los etiquetadores tradicionales.

	Estrategia 2	
OOVF+	2.591	59,95%
OOVF-	1.731	40,05%
NAF+	23.576	97,93%
NAF-	498	2,07%
AF+	21.814	94,72%
AF-	1.216	5,28%
S1	93,301	
S2	89,225	

Tabla 9.3: Resultados de etiquetación de la Estrategia 2 sobre la parte con trazas del corpus SUSANNE

9.2.3 Análisis de resultados

Hemos visto que el uso de las restricciones sintácticas impuestas por una gramática puede constituir una gran ayuda a la hora de etiquetar un texto en lenguaje natural. No obstante, el presente trabajo no está cerrado. Como sabemos, para cada subsecuencia de etiquetas, el analizador sintáctico proporciona dos datos importantes: el número de subárboles diferentes que la generan y la probabilidad del mejor de esos subárboles. Respecto al primero de estos datos, aunque su uso es cuestionable ya que los árboles más numerosos no son necesariamente los más probables, lo que sí es evidente es que no ha sido probado en ninguna de las estrategias. Respecto al segundo dato, la probabilidad de los subárboles ha sido utilizada para elegir una sola de entre todas las subsecuencias que presentan igual longitud en un momento dado, pero dicha probabilidad no ha sido integrada de forma alguna en el HMM responsable de realizar el filtrado estadístico. Por tanto, todavía queda una gran cantidad de experimentos por realizar, la mayor parte de los cuales estarían destinados a integrar nuevas y más sofisticadas estrategias de combinación de las subsecuencias, que puedan mejorar aún más el proceso de etiquetación.

Sobre la repercusión de los resultados obtenidos, es necesario hacer también las siguientes reflexiones. En primer lugar, para poder abordar la aplicación de las nuevas técnicas de etiquetación que acabamos de presentar, resulta imprescindible la disponibilidad de una gramática o de un banco de árboles del que poder extraerla. No habiendo sido posible contar con este tipo de recursos ligüísticos para el español, la experimentación desarrollada en este punto se ha realizado únicamente para el idioma inglés. Por tanto, constituye un gran desafío el repetir los mismos tests sobre el español en cuanto esto sea posible, con el fin de dar más representatividad a los datos obtenidos.

Por otra parte, e independientemente del idioma de aplicación, hubiera sido también muy conveniente el haber contado con recursos ligüísticos más extensos. Tal y como hemos visto anteriormente, dos de los parámetros que definen el nivel de diseño de un banco de árboles, y por tanto la calidad de la correspondiente gramática que se puede extraer de él, son el número de ambigüedades, que debe ser bajo, y el porcentaje de coincidencia con el análisis de referencia, que debe ser alto. Pues bien, existe aún otro parámetro de calidad relacionado con el conjunto de reglas de la gramática y con sus respectivas probabilidades. Ambos conjuntos deberían mantener un cierto grado de estabilidad a medida que el tamaño del banco de árboles va creciendo. Si las variaciones entre los conjuntos de reglas y probabilidades en cada etapa de extracción son bajas, se puede afirmar también que la calidad del recurso es alta. Este tipo de medición incremental, similar a la que hemos realizado con los modelos de etiquetación tradicionales, daría una mayor coherencia a los experimentos, y podría proporcionar ideas sobre la localización de un posible umbral que defina cuándo el uso de la gramática aporta beneficios y cuándo deja de hacerlo. Sin embargo, esto sólo sería posible con un banco de árboles mucho mayor que el del corpus SUSANNE. El tamaño medio de este último no lo hace apropiado para realizar más particiones a mayores de las que ya han sido efectuadas sobre él, para extraer por un lado una gramática y un corpus de entrenamiento, y por otro un corpus de aplicación.

Por último, es importante destacar que una gramática, o un banco de árboles desde donde extraerla, es un recurso lingüístico muy costoso y complejo de diseñar, y podríamos estar infrautilizándolo si solamente se aprovecha para la etiquetación de textos. En este punto, resulta obligado plantearse para qué más pueden ser aprovechables las técnicas que hemos presentado aquí. Una posibilidad es, por supuesto, integrar estrategias para extraer automáticamente las reglas sintácticas que completen la gramática, lo cual constituye precisamente el objetivo último del análisis sintáctico robusto. Sin embargo, una aplicación menos ambiciosa, pero de enorme utilidad práctica, puede ser la integración inmediata de estas nuevas técnicas en sistemas de recuperación de información [Fagan 1987, Smeaton 1992]. El filtrado estadístico realizado sobre las subsecuencias de etiquetas implica de manera inherente un filtrado de los subárboles de análisis parcial, que puede mejorar la viabilidad del procesamiento automático de consultas en lenguaje natural en este tipo de sistemas. Estas y otras líneas de trabajo futuro, junto con las principales conclusiones del presente trabajo, se exponen más detalladamente en el capítulo final.

Capítulo 10

Conclusiones y trabajo futuro

Para poder llevar a cabo con garantías el proceso de etiquetación de un texto en lenguaje natural, hemos visto que resulta indispensable tener en cuenta el contexto en el que aparecen cada una de las palabras de ese texto. En el presente trabajo se han estudiado diferentes métodos capaces de modelizar dicho contexto. La mayoría de estos métodos se pueden agrupar en dos clases principales. Por un lado encontramos los sistemas basados en modelos estocásticos, y por otro los basados en reglas o transformaciones contextuales. En este trabajo se ha realizado también una evaluación exhaustiva de cada uno de estos modelos sobre textos en español y en inglés. Para el caso del español, como ya habíamos avanzado, esta evaluación resulta de gran interés ya en sí misma debido a que actualmente los recursos lingüísticos disponibles no abundan. Por tanto, este estudio ha cubierto uno de nuestros principales objetivos, que era el de aportar cifras concretas que proporcionen una idea más clara del comportamiento de los etiquetadores sobre nuestro idioma.

De los cuatro sistemas evaluados, TnT [Brants 1996] y Galena [Vilares *et al.* 1995, Sacristán y Graña 1999] responden de forma clara al paradigma de etiquetación estocástica. Ambos utilizan como principio de funcionamiento los modelos de Markov ocultos o HMM,s. El sistema Brill es un sistema basado en reglas [Brill 1993b], y el sistema JMX representa un acercamiento híbrido que utiliza fundamentos de las otras dos aproximaciones [Ratnaparkhi 1996]. Tras el diseño de una estrategia de evaluación específica para este tipo de sistemas [Graña y Rajman 1999] y tras su posterior aplicación para el análisis de los experimentos realizados, se ha observado que los sistemas TnT y Galena ofrecen claramente mejores rendimientos, tanto en los índices de acierto como en los tiempos de entrenamiento y etiquetación. Por tanto, la primera conclusión importante que se ha extraído es que, para el proceso de etiquetación de textos en lenguaje natural, el marco probabilístico resulta ser más adecuado que las aproximaciones simbólicas.

10.1 Extensiones del formalismo HMM para la etiquetación

Desde ese momento, y por los motivos anteriormente expuestos, la mayor parte de los esfuerzos se dirigieron hacia un estudio profundo y una posterior extensión de los sistemas de etiquetación puramente estocásticos, en un intento de adaptación a situaciones donde los recursos disponibles para el entrenamiento de dichos sistemas resultan particularmente escasos. A este nivel, las principales aportaciones de esta tesis se centran en los dos aspectos siguientes:

1. Cualquier sistema de etiquetación basado en un HMM debe enfrentarse inevitablemente a los problemas causados por el fenómeno de los datos dispersos. La manera usual de paliar este tipo de problemas ha sido casi siempre la aplicación de métodos de suavización

de parámetros basados en esquemas de interpolación lineal. En este trabajo hemos querido recuperar el método de *marcha atrás* o *back-off* [Katz 1987], un procedimiento que tradicionalmente había sido utilizado de manera casi exclusiva para la suavización de los n-gramas de palabras en entornos de reconocimiento de voz. En nuestro contexto, dicho método se ha utilizado para la suavización de los n-gramas de etiquetas, que son precisamente los parámetros que formalizan las transiciones entre los estados del HMM subyacente.

El método de *back-off* confía en las frecuencias de los sucesos, si es que hay un número suficiente de apariciones de los mismos. Si no lo hay, pero el suceso está todavía presente, entonces utiliza las fórmulas de Good-Turing. Y sólo utiliza una hipótesis de menor nivel, como puede ser el suavizado lineal, si el suceso no está presente en absoluto. Como consecuencia, cuando el conjunto de datos de entrenamiento es reducido y la simple interpolación lineal no dispone de suficientes evidencias como para realizar un suavizado óptimo, el método de *back-off* demuestra ser más elaborado y más robusto.

2. En el presente trabajo se ha realizado también un importante esfuerzo destinado a estudiar los métodos de integración de diccionarios externos dentro de un marco de etiquetación estocástica. La manera más natural de realizar esta integración es el método *de sumar uno* o método *adding one*, que consiste en utilizar el diccionario como si se tratara de un corpus etiquetado adicional, donde a cada uno de los pares palabra-etiqueta se le asigna una frecuencia de aparición igual a 1. Sin embargo, con este método, dadas dos palabras con la misma etiqueta, w_i y w_j, la primera presente sólo en el diccionario y la segunda presente sólo en el corpus de entrenamiento, encontramos que ambas tendrán la misma frecuencia y por tanto la misma probabilidad de emisión, lo cual no produce una representación coherente del modelo que se está estimando. Es decir, cuando se integra un conjunto de palabras externo, inevitablemente se produce una importante alteración de los parámetros del modelo, lo que nos ha llevado a considerar otro tipo de procedimientos como puede ser el método de Good-Turing [Church y Gale 1991].

   Cada uno de los pares palabra-etiqueta presentes en el diccionario y no en el corpus de entrenamiento puede verse como un suceso no observado, es decir, de frecuencia cero, y las fórmulas de Good-Turing constituyen en sí mismas un método capaz de asignar probabilidades distintas de cero a este tipo de eventos. Ocurre además que, en casos como el anteriormente descrito, esta técnica garantiza que la probabilidad de emisión de w_i siempre será menor que la de w_j, lo cual desvirtúa menos el modelo. No obstante, es importante recordar que, en nuestro caso, este tipo de integración se realiza considerando conjuntos aislados de palabras sobre cada una de las distintas etiquetas, y sólo cuando se cumple que $n_0 \gg n_1 \gg n_2$. Fuera de esta situación, este método no siempre es aplicable y existe el riesgo de utilizarlo incorrectamente.

Se ha observado que estas adaptaciones se traducen efectivamente en mejoras de rendimiento, que se hacen más palpables en las condiciones anteriormente planteadas, es decir, cuando los textos de entrenamiento son muy pequeños y sin embargo los diccionarios externos son muy grandes. Este tipo de situaciones son precisamente las que definen el estado actual del procesamiento automático del español. Como conclusión importante, podemos afirmar que en un futuro inmediato estas mejoras permitirán abordar con más garantías el procesamiento automático de este idioma, e incluso el de otros para los cuales apenas existen textos de referencia, como es el caso del gallego.

No obstante, el trabajo a este nivel no está todavía cerrado. A lo largo de este estudio, hemos trabajado bajo la hipótesis de que los textos llegan correctamente segmentados al sistema de

etiquetación. Entre otras tareas, el proceso de segmentación se encarga de identificar unidades de información tales como las frases o las propias palabras. En el caso de las palabras, la problemática se centra en que el concepto ortográfico de palabra no siempre coincide con el concepto lingüístico. Las aproximaciones más sencillas consideran igualmente las palabras ortográficas y amplían las etiquetas para representar aquellos fenómenos que sean relevantes. Por ejemplo, en este estudio, la palabra `reconocerse` se etiqueta como `VOOOfOPE1` aun cuando está formada por un verbo y un pronombre enclítico, y las palabras de la expresión `and so on` se etiquetan respectivamente como `RAc31`, `RAc32` y `RAc33` aun cuando constituyen un único término. Sin embargo, en idiomas como el gallego, este planteamiento no es viable, ya que su gran complejidad morfológica produciría un crecimiento excesivo del juego de etiquetas.

La solución pasa entonces por no ampliar el juego de etiquetas básico. Como ventajas, la complejidad del proceso de etiquetación no se ve afectada por un número elevado de etiquetas, y la información relativa a cada término lingüístico se puede expresar de manera más precisa. Como desventaja, se complican las labores del preprocesador, que unas veces tendrá que partir una palabra en varias, y otras veces tendrá que juntar varias palabras en una sola. Las mayores dificultades surgen cuando esta segmentación es ambigua. Por ejemplo, la palabra `ténselo` puede ser una forma del verbo `tener` con dos pronombres enclíticos, o bien una forma del verbo `tensar` con un solo pronombre. De igual forma, la expresión `sin embargo` se etiquetará normalmente de manera conjunta como una conjunción, pero en algún otro contexto podría ser una secuencia formada por una preposición y un sustantivo.

Una vez más, este tipo de ambigüedades sólo pueden ser resueltas estudiando el contexto. Como esto es precisamente lo que realiza el proceso de etiquetación, cuando el preprocesador no pueda decidir, todas las posibilidades de segmentación deberían pasar al etiquetador. Pero en casos como los anteriores, nos encontramos con que debemos de evaluar varios flujos de palabras de distinta longitud. La dificultad radica en que, en principio, esos flujos no pueden ser evaluados sobre el mismo enrejado, ya que típicamente los flujos más largos alcanzarán probabilidades menores que los flujos más cortos, y por tanto estos últimos se verán favorecidos.

Una posible solución es evaluar cada flujo de manera independiente y dividir la probabilidad acumulada entre el número de palabras de dicho flujo. Este valor resulta comparable de un flujo a otro, ya que las probabilidades logarítmicas constituyen una magnitud aditiva de la cual se puede extraer la media aritmética. No obstante, si en una frase aparecen varias de estas palabras problemáticas, el número de flujos a evaluar viene dado por el producto cruzado de las diferentes posibilidades de segmentación de dichas palabras. Por tanto, estamos ante un problema de excesiva complejidad temporal. En este punto, nuestro reto para el futuro es diseñar una extensión del formalismo HMM que permita evaluar flujos de palabras de distinta longitud sobre un único enrejado.

10.2 Análisis sintáctico robusto, etiquetación y recuperación de información

Los sistemas de etiquetación puramente estocásticos suelen etiquetar erróneamente alrededor del 2 ó 3% de las palabras de un texto dado. Para poder enfrentarse con garantías a este pequeño porcentaje de errores, es necesario incorporar alguna fuente de información de más alto nivel. Por tanto, hemos querido estudiar qué efecto produce el uso combinado de los sistemas de etiquetación tradicionales con información sintáctica procedente de una gramática. Éste es precisamente el segundo gran objetivo que perseguíamos en el presente trabajo.

Ante la dificultad de disponer de gramáticas de cobertura total para un idioma dado, se ha desarrollado un entorno de análisis sintáctico estocástico que permite experimentar cómodamente con técnicas de análisis sintáctico robusto orientadas específicamente al problema

de la etiquetación [Barcala y Graña 1999]. Tras recuperar las etiquetas que los subárboles de análisis asignan a las palabras, se han diseñado estrategias que combinan dichas subsecuencias de etiquetación parcial, y producen secuencias de cobertura total para las frases no completamente analizables [Graña *et al.* 1999]. La etiquetación final se obtiene mediante un HMM que opera como un filtro estadístico sobre dichas secuencias y selecciona la más probable.

Tras el análisis de los experimentos realizados, hemos podido observar que el uso de las restricciones sintácticas impuestas por una gramática puede constituir una gran ayuda a la hora de etiquetar un texto en lenguaje natural. Las causas hay que atribuirlas a que una gramática estocástica puede verse como un modelo de lenguaje restrictivo, capaz de detectar y formalizar las dependencias de larga distancia y las estructuras recursivas que escapan a los modelos lineales.

Es importante destacar que, hasta donde llega nuestro conocimiento, la idea de utilizar en este contexto una gramática estocástica en combinación con un HMM tradicional constituye una aportación totalmente nueva y original. Tan sólo hemos encontrado una línea similar en [Sawaf *et al.* 2000], un trabajo surgido durante la realización de la presente memoria, y cuya aplicación final se centra realmente en la traducción automática, lo que conlleva de manera inherente un proceso previo de etiquetación.

No obstante, hay que destacar que una gramática es un recurso lingüístico muy costoso y complejo de diseñar, y podríamos estar infrautilizándolo si solamente se aprovecha para la etiquetación de textos. Por tanto, la integración de estas nuevas técnicas en sistemas de recuperación de información se perfila como una de las principales líneas de trabajo futuro. Aunque todavía está sin resolver la discusión de si las técnicas de NLP producen mejoras reales de rendimiento o no en este tipo de sistemas, cada vez son más numerosos los proyectos que se orientan en este sentido.

Bajo esta realidad, es necesario recordar que la integración de técnicas de NLP en este contexto pasa necesariamente por la comparación de los análisis sintácticos parciales de las consultas con los de las frases que conforman el conjunto de documentos disponibles, en lugar de buscar la simple coincidencia individual de las palabras. Teniendo en cuenta que los algoritmos de comparación de estructuras arborescentes presentan una complejidad temporal alta, se hace necesaria una reducción del conjunto total de árboles a comparar, y resulta evidente que el filtrado estadístico realizado sobre las subsecuencias de etiquetas implica automáticamente un filtrado de los subárboles de análisis parcial.

Por supuesto, existe la alternativa de realizar un paso de etiquetación previo al análisis sintáctico, lo cual también elimina radicalmente muchos de los subárboles que se hubieran generado trabajando sólo con la gramática. Pero tal y como hemos visto, esto no siempre es una buena idea y el uso de técnicas combinadas como las que hemos presentado aquí puede resultar más apropiado en determinadas condiciones, y puede desembocar en una mejora de la viabilidad del procesamiento automático de consultas en lenguaje natural.

En definitiva, la disponibilidad de herramientas eficientes de análisis léxico y sintáctico, capaces de enfrentarse a diccionarios y gramáticas incompletos con la ayuda del marco estadístico, abre perspectivas de aplicación inmediata en sistemas de tratamiento de información a alto nivel, y más concretamente en los sistemas de recuperación de información.

10.3 Reflexiones finales

Antes de finalizar, es necesario recordar que uno de los principales inconvenientes de cualquier sistema de etiquetación es que los resultados obtenidos dependen en exceso del estilo de los textos de entrenamiento. Actualmente, los esfuerzos de investigación se dirigen hacia el diseño de técnicas híbridas capaces de paliar este problema. Sin embargo, la mayoría de ellas precisan

la incorporación manual por parte del usuario de información capaz de enfrentarse a los errores sistemáticos que cometen los etiquetadores, y a los fenómenos lingüísticos no correctamente formalizados. A lo largo de todo el presente trabajo, hemos querido cubrir las aproximaciones que utilizan únicamente la información que se puede extraer de los recursos lingüísticos de manera automática. Bajo esta hipótesis de trabajo, creemos que no se puede ir más allá.

Por otra parte, a pesar de los grandes avances que se han realizado en este campo de trabajo, podemos afirmar que las técnicas de NLP no han alcanzado todavía el grado de madurez deseable. A medida que existan disponibles más recursos lingüísticos de calidad, las técnicas de NLP experimentarán mejoras sucesivas de manera inherente. Por tanto, estamos en un momento en el que todavía es necesario que el diseño de aplicaciones finales conviva paralelamente con el diseño de aplicaciones que permitan a los expertos lingüistas construir más recursos, y cubrir así cada vez más idiomas y estilos.

Además, las aplicaciones finales a las que ahora mismo tenemos acceso mediante técnicas de lenguaje natural son simplemente pinceladas en comparación con lo que nos depara la próxima generación. La explosión de las técnicas de NLP a nivel industrial está todavía por llegar. Pueden ser más o menos años, pero la gran revolución está cercana y trae consigo una especialización urgente. Podemos y debemos dar rienda suelta a la creatividad para adaptarnos a las nuevas exigencias, pero sin olvidar nunca los recursos fundamentales que constituyen la base para que cualquier sistema de NLP funcione adecuadamente. De esta manera, tendremos garantizada también la correcta formación de los ingenieros destinados a cubrir esta necesidad social que día a día se hace más inminente.

Atestigua una torcida noción de un lenguaje cualquiera el estimarlo constituido por un número dado de voces, ni una más, ni una menos. A una lengua, como a cualquier otra vía de comunicación humana, si ha de vivir vida exuberante, le es forzoso ser, más que rica, fecunda.

Miguel de Unamuno (Bilbao 1864 - Salamanca 1936)

Parte IV

Apéndices, bibliografía e índice de materias

Apéndice A

Juegos de etiquetas

A.1 Juego de etiquetas del proyecto CRATER

La siguiente lista constituye el juego de etiquetas utilizado en el proyecto CRATER [Sánchez 1994]. La lista consta de dos columnas: la primera columna es la etiqueta y la segunda columna contiene la descripción de la etiqueta y a veces alguna palabra de ejemplo. El cardinal de este conjunto es de 475 etiquetas.

ACRNM	Sigla (ADN)
ADJCP	Adjetivo comparativo general plural (mayores, menores)
ADJCS	Adjetivo comparativo general singular (mayor, menor)
ADJGFP	Adjetivo positivo general femenino plural
ADJGFS	Adjetivo positivo general femenino singular
ADJGMP	Adjetivo positivo general masculino plural
ADJGMS	Adjetivo positivo general masculino singular
ADJSFP	Adjetivo superlativo general femenino plural (máximas, mínimas)
ADJSFS	Adjetivo superlativo general femenino singular (máxima, mínima)
ADJSMP	Adjetivo superlativo general masculino plural (máximos, mínimos)
ADJSMS	Adjetivo superlativo general masculino singular (máximo, mínimo, grandísimo)
ADVGR	Adverbio con grado positivo (muy, demasiado, mucho)
ADVGRC	Adverbio con grado comparativo (más, menos)
ADVGRS	Adverbio con grado superlativo (abundantísimamente)
ADVINT	Adverbio interrogativo (cómo)
ADVL	Adverbio locativo direccional (abajo)
ADVLD	Adverbio locativo dinámico (adelante)
ADVLE	Adverbio locativo estático (dentro)
ADVLID	Adverbio locativo interrogativo dinámico (adónde)
ADVLIN	Adverbio locativo interrogativo (dónde)
ADVLP	Adverbio locativo con deixis próxima (aquí)
ADVLR	Adverbio locativo con deixis remota (allí)
ADVLRD	Adverbio locativo relativo dinámico (adonde)
ADVLRE	Adverbio locativo relativo direccional (donde)
ADVMRE	Adverbio relativo modal (como)
ADVN	Adverbio general (salvajemente, bien, probablemente)
ADVNEG	Adverbio general negativo (tampoco)
ADVT	Adverbio temporal (ahora, ayer)
ADVTIN	Adverbio temporal interrogativo (cuándo)

ADVTNE	Adverbio temporal negativo (nunca)
ADVTRE	Adverbio temporal relativo (cuando)
ALFP	Letra del alfabeto en plural (As/Aes, bes)
ALFS	Letra del alfabeto en singular (A, b)
ARCAFS	Artículo indefinido femenino singular y cardinal con función pronominal (una)
ARCAMS	Artículo indefinido masculino singular y cardinal no pronominal (un)
ARQUFP	Artículo indefinido femenino plural y cuantificador con función pronominal (unas)
ARQUMP	Artículo indefinido masculino plural y cuantificador con función pronominal (unos)
ARTDFP	Artículo definido femenino plural (las)
ARTDFS	Artículo definido femenino singular (la)
ARTDMP	Artículo definido masculino plural (los)
ARTDMS	Artículo definido masculino singular (el)
ARTDNS	Artículo definido neutro singular (lo)
CARDFP	Cardinal femenino plural con función pronominal (doscientas)
CARDGU	Cardinales con guión (40-50, 1850-1990)
CARDMP	Cardinal masculino plural con función pronominal (doscientos)
CARDPS	Cardinal pronominal singular (uno)
CARDXP	Cardinal plural neutro para el género (dos, tres, mil)
CARDXS	Cardinal singular como dígito (1)
CARNMP	Cardinal no pronominal masculino plural (veintiún)
CC	Conjunción coordinada (y, o)
CCAD	Conjunción coordinada adversativa (pero)
CCNEG	Conjunción coordinada negativa (ni)
CODE	Código alfanumérico
CQUE	que (como conjunción)
CSUBF	Conjunción subordinada que introduce cláusulas finitas (apenas)
CSUBI	Conjunción subordinada que introduce cláusulas no finitas (al)
CSUBX	Conjunción subordinada para tipos subordinados (aunque)
DMDPFP	Pronombre demostrativo femenino plural con deixis distal (ésas)
DMDPFS	Pronombre demostrativo femenino singular con deixis distal (ésa)
DMDPMP	Pronombre demostrativo masculino plural con deixis distal (ésos)
DMDPMS	Pronombre demostrativo masculino singular con deixis distal (ése)
DMDPNS	Pronombre demostrativo neutro singular con deixis distal (eso)
DMDXFP	Demostrativo femenino plural (con función pronominal) con deixis distal (esas)
DMDXFS	Demostrativo femenino singular (con función pronominal) con deixis distal (esa)
DMDXMP	Demostrativo masculino plural (con función pronominal) con deixis distal (esos)
DMDXMS	Demostrativo masculino singular (con función pronominal) con deixis distal (ese)
DMPPFP	Pronombre demostrativo femenino plural con deixis próxima (éstas)
DMPPFS	Pronombre demostrativo femenino singular con deixis próxima (ésta)
DMPPMP	Pronombre demostrativo masculino plural con deixis próxima (éstos)
DMPPMS	Pronombre demostrativo masculino singular con deixis próxima (éste)
DMPPNS	Pronombre demostrativo neutro singular con deixis próxima (esto)
DMPXFP	Demostrativo femenino plural (con función pronominal) con deixis próxima (estas)
DMPXFS	Demostrativo femenino singular (con función pronominal) con deixis próxima (esta)
DMPXMP	Demostrativo masculino plural (con función pronominal) con deixis próxima (estos)

DMPXMS	Demostrativo masculino singular (con función pronominal) con deixis próxima (este)
DMRPFP	Pronombre demostrativo femenino plural con deixis remota (aquéllas)
DMRPFS	Pronombre demostrativo femenino singular con deixis remota (aquélla)
DMRPMP	Pronombre demostrativo masculino plural con deixis remota (aquéllos)
DMRPMS	Pronombre demostrativo masculino singular con deixis remota (aquél)
DMRPNS	Pronombre demostrativo neutro singular con deixis remota (aquello)
DMRXFP	Demostrativo femenino plural (con función pronominal) con deixis remota (aquellas)
DMRXFS	Demostrativo femenino singular (con función pronominal) con deixis remota (aquella)
DMRXMP	Demostrativo masculino plural (con función pronominal) con deixis remota (aquellos)
DMRXMS	Demostrativo masculino singular (con función pronominal) con deixis remota (aquel)
FO	Fórmula
INTPXP	Pronombre interrogativo plural para personas neutro para el género (quiénes)
INTPXS	Pronombre interrogativo singular para personas neutro para el género (quién)
INTXFP	Interrogativo femenino plural con función pronominal (cuántas)
INTXFS	Interrogativo femenino singular con función pronominal para inanimados (cuánta)
INTXMP	Interrogativo masculino plural con función pronominal (cuántos)
INTXMS	Interrogativo masculino y neutro singular con función pronominal para inanimados (cuánto)
INTXXP	Interrogativo plural neutro para el género con función pronominal (cuáles)
INTXXS	Interrogativo singular neutro para el género con función pronominal (cuál)
INTXXX	Interrogativo con función pronominal neutro para género y número (qué)
ITJN	Interjección (oh, ja)
NCFP	Nombre común femenino plural (mesas, manos)
NCFS	Nombre común femenino singular (mesa, mano)
NCMP	Nombre común masculino plural (libros, ordenadores)
NCMS	Nombre común masculino singular (libro, ordenador)
NEG	Negación
NMEAFP	Nombre unidad de medida femenino plural (hectáreas, micras)
NMEAFS	Nombre unidad de medida femenino singular (hectárea, micra)
NMEAMP	Nombre unidad de medida masculino plural (metros, litros)
NMEAMS	Nombre unidad de medida masculino singular (metro, litro)
NMON	Nombres de mes (singular y plural) (diciembre(s))
NPAFP	Nombre propio antropónimo femenino plural (Marías)
NPAFS	Nombre propio antropónimo femenino singular (María)
NPAMP	Nombre propio antropónimo masculino plural (Juanes)
NPAMS	Nombre propio antropónimo masculino singular (Juan)
NPAXX	Nombre propio antropónimo neutro para género y número (Rodríguez)
NPGP	Otros nombres propios en plural (Aberri Egunas)
NPGS	Otros nombres propios en singular (Eutelsat)
NPGX	Otros nombres propios (etiqueta por defecto para nombres propios desconocidos)
NPOS	Nombres propios colectivos en singular (Iberia)
NPTOP	Nombres propios colectivos o topónimos en plural (Coreas)
NPTOS	Nombres propios colectivos o topónimos en singular (Madrid)
NPTP	Nombres propios topónimos en plural (Pirineos)

NPTS	Nombres propios topónimos en singular (Guadalquivir)
NWEE	Nombres de días de la semana (singular y plural) (sábado(s))
ORDNMS	Ordinal no pronominal masculino singular (primer, tercer)
ORDXFP	Ordinal femenino plural con función pronominal (primeras, segundas)
ORDXFS	Ordinal femenino singular con función pronominal (primera, segunda)
ORDXMP	Ordinal masculino plural con función pronominal (primeros, segundos)
ORDXMS	Ordinal masculino singular con función pronominal (primero, segundo)
PAL	Contracción formada por a y el
PDEL	Contracción formada por de y el
PE	Palabra extranjera
PNC	Palabra no clasificada
PPC1P	Pronombre personal enclítico, primera persona plural OD/OI (nos)
PPC1S	Pronombre personal enclítico, primera persona singular OD/OI (me)
PPC2P	Pronombre personal enclítico, segunda persona plural OD/OI (os)
PPC2S	Pronombre personal enclítico, segunda persona singular OD/OI (te)
PPC3P	Pronombre personal enclítico, tercera persona plural OD/OI (les)
PPC3S	Pronombre personal enclítico, tercera persona singular OD/OI (le)
PPN1S	Pronombre personal nominativo, primera persona singular (yo)
PPN2S	Pronombre personal nominativo, segunda persona singular (tú)
PPO3FP	Pronombre personal enclítico, femenino tercera persona plural OD (las)
PPO3FS	Pronombre personal enclítico, femenino tercera persona singular OD (la)
PPO3MP	Pronombre personal enclítico, masculino tercera persona plural OD (los)
PPO3XS	Pronombre personal enclítico, masculino o neutro tercera persona singular OD (lo)
PPOSFP	Pronombre posesivo femenino plural (tuyas, suyas)
PPOSFS	Pronombre posesivo femenino singular (mía, tuya)
PPOSMP	Pronombre posesivo masculino plural (míos, tuyos)
PPOSMS	Pronombre posesivo masculino singular (tuyo, suyo)
PPOSPP	Pronombre posesivo prenominal plural (mis, tus, sus)
PPOSPS	Pronombre posesivo prenominal singular (mi, tu, su)
PPP1S	Pronombre personal oblicuo, primera persona singular (mí)
PPP2S	Pronombre personal oblicuo, segunda persona singular (ti)
PPP3X	Pronombre personal oblicuo, tercera persona neutro para el número (sí)
PPX1FP	Pronombre personal nominativo u oblicuo, femenino primera persona plural (nosotras)
PPX1MP	Pronombre personal nominativo u oblicuo, masculino primera persona plural (nosotros)
PPX2FP	Pronombre personal nominativo u oblicuo, femenino segunda persona plural (vosotras)
PPX2MP	Pronombre personal nominativo u oblicuo, masculino segunda persona plural (vosotros)
PPX3FP	Pronombre personal nominativo u oblicuo, femenino tercera persona plural (ellas)
PPX3FS	Pronombre personal nominativo u oblicuo, femenino tercera persona singular (ella)
PPX3MP	Pronombre personal nominativo u oblicuo, masculino tercera persona plural (ellos)
PPX3MS	Pronombre personal nominativo u oblicuo, masculino tercera persona singular (él)
PPX3NS	Pronombre personal nominativo u oblicuo, neutro tercera persona singular (ello)

PPXT2P	Pronombre personal nominativo u oblicuo, segunda persona plural de cortesía (ustedes)
PPXT2S	Pronombre personal nominativo u oblicuo, segunda persona singular de cortesía (usted)
PREP	Preposición
PREPN	Preposición negativa (sin)
QUDF	Cuantificador distributivo femenino plural (sendas)
QUDM	Cuantificador distributivo masculino plural (sendos)
QUDX	Cuantificador distributivo neutro para el género (cada)
QUNFP	Cuantificador no pronominal femenino plural (ciertas)
QUNFS	Cuantificador no pronominal femenino singular (cualquier)
QUNMP	Cuantificador no pronominal masculino plural (ciertos)
QUNMS	Cuantificador no pronominal masculino singular (algún)
QUNNMS	Cuantificador no pronominal con polaridad negativa masculino singular (ningún)
QUPA	Cuantificador pronominal para personas singular (alguien)
QUPI	Cuantificador pronominal para inanimados singular (algo)
QUPMUL	Cuantificador pronominal que indica múltiplos singular (doble, triple)
QUPNA	Cuantificador pronominal con polaridad negativa para personas singular (nadie)
QUPNI	Cuantificador pronominal con polaridad negativa para inanimados singular (nada)
QUPNX	Cuantificador pronominal con polaridad negativa para animados e inanimados masculino singular (ninguno)
QUXFP	Cuantificador con función pronominal feminino plural (todas, algunas, cualesquiera)
QUXFS	Cuantificador con función pronominal feminino singular (toda, alguna, cualquiera)
QUXMP	Cuantificador con función pronominal masculino plural (todos, algunos, cualesquiera)
QUXMS	Cuantificador con función pronominal masculino singular (todo, alguno, cualquiera)
QUXNFP	Cuantificador con función pronominal con polaridad negativa femenino plural (ningunas)
QUXNFS	Cuantificador con función pronominal con polaridad negativa femenino singular (ninguna)
QUXNMP	Cuantificador con función pronominal con polaridad negativa masculino plural (ningunos)
RELPFP	Pronombre relativo posesivo femenino plural (cuyas)
RELPFS	Pronombre relativo posesivo femenino singular (cuya)
RELPMP	Pronombre relativo posesivo masculino plural (cuyos)
RELPMS	Pronombre relativo posesivo masculino singular (cuyo)
RELPXP	Pronombre relativo para animados pural, neutro para el género (quienes)
RELPXS	Pronombre relativo para animados singular, neutro para el género (quien)
RELXFP	Pronombre relativo con función pronominal femenino plural (cuantas)
RELXFS	Pronombre relativo con función pronominal femenino singular (cuanta)
RELXMP	Pronombre relativo con función pronominal masculino plural (cuantos)
RELXMS	Pronombre relativo con función pronominal masculino singular (cuanto)
ROMAN	Número romano (XI)
SE	se (como partícula)
TRATF	Nombre de título femenino (Sra., Dña., Exma.)
TRATM	Nombre de título masculino (Sr., D., Prof., Exmo.)

UMFX	Unidad de medida femenino, neutro para el número (pta.)
UMMX	Unidad de medida masculino, neutro para el número (cm.)
VECI1P	Verbo estar - Indicativo tiempo condicional primera persona plural
VECI1S	Verbo estar - Indicativo tiempo condicional primera persona singular
VECI2P	Verbo estar - Indicativo tiempo condicional segunda persona plural
VECI2S	Verbo estar - Indicativo tiempo condicional segunda persona singular
VECI3P	Verbo estar - Indicativo tiempo condicional tercera persona plural
VECI3S	Verbo estar - Indicativo tiempo condicional tercera persona singular
VEFI1P	Verbo estar - Indicativo tiempo futuro primera persona plural
VEFI1S	Verbo estar - Indicativo tiempo futuro primera persona singular
VEFI2P	Verbo estar - Indicativo tiempo futuro segunda persona plural
VEFI2S	Verbo estar - Indicativo tiempo futuro segunda persona singular
VEFI3P	Verbo estar - Indicativo tiempo futuro tercera persona plural
VEFI3S	Verbo estar - Indicativo tiempo futuro tercera persona singular
VEFS1P	Verbo estar - Subjuntivo tiempo futuro primera persona plural
VEFS1S	Verbo estar - Subjuntivo tiempo futuro primera persona singular
VEFS2P	Verbo estar - Subjuntivo tiempo futuro segunda persona plural
VEFS2S	Verbo estar - Subjuntivo tiempo futuro segunda persona singular
VEFS3P	Verbo estar - Subjuntivo tiempo futuro tercera persona plural
VEFS3S	Verbo estar - Subjuntivo tiempo futuro tercera persona singular
VEGER	Verbo estar - Gerundio
VEII1P	Verbo estar - Indicativo tiempo imperfecto primera persona plural
VEII1S	Verbo estar - Indicativo tiempo imperfecto primera persona singular
VEII2P	Verbo estar - Indicativo tiempo imperfecto segunda persona plural
VEII2S	Verbo estar - Indicativo tiempo imperfecto segunda persona singular
VEII3P	Verbo estar - Indicativo tiempo imperfecto tercera persona plural
VEII3S	Verbo estar - Indicativo tiempo imperfecto tercera persona singular
VEINF	Verbo estar - Infinitivo
VEIS1P	Verbo estar - Subjuntivo tiempo imperfecto primera persona plural
VEIS1S	Verbo estar - Subjuntivo tiempo imperfecto primera persona singular
VEIS2P	Verbo estar - Subjuntivo tiempo imperfecto segunda persona plural
VEIS2S	Verbo estar - Subjuntivo tiempo imperfecto segunda persona singular
VEIS3P	Verbo estar - Subjuntivo tiempo imperfecto tercera persona plural
VEIS3S	Verbo estar - Subjuntivo tiempo imperfecto tercera persona singular
VEPI1P	Verbo estar - Indicativo tiempo presente primera persona plural
VEPI1S	Verbo estar - Indicativo tiempo presente primera persona singular
VEPI2P	Verbo estar - Indicativo tiempo presente segunda persona plural
VEPI2S	Verbo estar - Indicativo tiempo presente segunda persona singular
VEPI3P	Verbo estar - Indicativo tiempo presente tercera persona plural
VEPI3S	Verbo estar - Indicativo tiempo presente tercera persona singular
VEPM2P	Verbo estar - Imperativo segunda persona plural
VEPM2S	Verbo estar - Imperativo segunda persona singular
VEPS1P	Verbo estar - Subjuntivo tiempo presente primera persona plural
VEPS1S	Verbo estar - Subjuntivo tiempo presente primera persona singular
VEPS2P	Verbo estar - Subjuntivo tiempo presente segunda persona plural
VEPS2S	Verbo estar - Subjuntivo tiempo presente segunda persona singular
VEPS3P	Verbo estar - Subjuntivo tiempo presente tercera persona plural
VEPS3S	Verbo estar - Subjuntivo tiempo presente tercera persona singular
VEPX	Verbo estar - Participio pasado

VEXI1P	Verbo estar - Indicativo tiempo pretérito primera persona plural
VEXI1S	Verbo estar - Indicativo tiempo pretérito primera persona singular
VEXI2P	Verbo estar - Indicativo tiempo pretérito segunda persona plural
VEXI2S	Verbo estar - Indicativo tiempo pretérito segunda persona singular
VEXI3P	Verbo estar - Indicativo tiempo pretérito tercera persona plural
VEXI3S	Verbo estar - Indicativo tiempo pretérito tercera persona singular
VHCI1P	Verbo haber - Indicativo tiempo condicional primera persona plural
VHCI1S	Verbo haber - Indicativo tiempo condicional primera persona singular
VHCI2P	Verbo haber - Indicativo tiempo condicional segunda persona plural
VHCI2S	Verbo haber - Indicativo tiempo condicional segunda persona singular
VHCI3P	Verbo haber - Indicativo tiempo condicional tercera persona plural
VHCI3S	Verbo haber - Indicativo tiempo condicional tercera persona singular
VHFI1P	Verbo haber - Indicativo tiempo futuro primera persona plural
VHFI1S	Verbo haber - Indicativo tiempo futuro primera persona singular
VHFI2P	Verbo haber - Indicativo tiempo futuro segunda persona plural
VHFI2S	Verbo haber - Indicativo tiempo futuro segunda persona singular
VHFI3P	Verbo haber - Indicativo tiempo futuro tercera persona plural
VHFI3S	Verbo haber - Indicativo tiempo futuro tercera persona singular
VHFS1P	Verbo haber - Subjuntivo tiempo futuro primera persona plural
VHFS1S	Verbo haber - Subjuntivo tiempo futuro primera persona singular
VHFS2P	Verbo haber - Subjuntivo tiempo futuro segunda persona plural
VHFS2S	Verbo haber - Subjuntivo tiempo futuro segunda persona singular
VHFS3P	Verbo haber - Subjuntivo tiempo futuro tercera persona plural
VHFS3S	Verbo haber - Subjuntivo tiempo futuro tercera persona singular
VHGER	Verbo haber - Gerundio
VHII1P	Verbo haber - Indicativo tiempo imperfecto primera persona plural
VHII1S	Verbo haber - Indicativo tiempo imperfecto primera persona singular
VHII2P	Verbo haber - Indicativo tiempo imperfecto segunda persona plural
VHII2S	Verbo haber - Indicativo tiempo imperfecto segunda persona singular
VHII3P	Verbo haber - Indicativo tiempo imperfecto tercera persona plural
VHII3S	Verbo haber - Indicativo tiempo imperfecto tercera persona singular
VHINF	Verbo haber - Infinitivo
VHIS1P	Verbo haber - Subjuntivo tiempo imperfecto primera persona plural
VHIS1S	Verbo haber - Subjuntivo tiempo imperfecto primera persona singular
VHIS2P	Verbo haber - Subjuntivo tiempo imperfecto segunda persona plural
VHIS2S	Verbo haber - Subjuntivo tiempo imperfecto segunda persona singular
VHIS3P	Verbo haber - Subjuntivo tiempo imperfecto tercera persona plural
VHIS3S	Verbo haber - Subjuntivo tiempo imperfecto tercera persona singular
VHPI1P	Verbo haber - Indicativo tiempo presente primera persona plural
VHPI1S	Verbo haber - Indicativo tiempo presente primera persona singular
VHPI2P	Verbo haber - Indicativo tiempo presente segunda persona plural
VHPI2S	Verbo haber - Indicativo tiempo presente segunda persona singular
VHPI3E	Verbo haber - Indicativo tiempo presente tercera persona singular existencial
VHPI3P	Verbo haber - Indicativo tiempo presente tercera persona plural
VHPI3S	Verbo haber - Indicativo tiempo presente tercera persona singular
VHPM2P	Verbo haber - Imperativo segunda persona plural
VHPM2S	Verbo haber - Imperativo segunda persona singular
VHPS1P	Verbo haber - Subjuntivo tiempo presente primera persona plural
VHPS1S	Verbo haber - Subjuntivo tiempo presente primera persona singular

VHPS2P	Verbo **haber** - Subjuntivo tiempo presente segunda persona plural
VHPS2S	Verbo **haber** - Subjuntivo tiempo presente segunda persona singular
VHPS3P	Verbo **haber** - Subjuntivo tiempo presente tercera persona plural
VHPS3S	Verbo **haber** - Subjuntivo tiempo presente tercera persona singular
VHPXFP	Verbo **haber** - Participio pasado femenino plural
VHPXFS	Verbo **haber** - Participio pasado femenino singular
VHPXMP	Verbo **haber** - Participio pasado masculino plural
VHPXMS	Verbo **haber** - Participio pasado masculino singular
VHXI1P	Verbo **haber** - Indicativo tiempo pretérito primera persona plural
VHXI1S	Verbo **haber** - Indicativo tiempo pretérito primera persona singular
VHXI2P	Verbo **haber** - Indicativo tiempo pretérito segunda persona plural
VHXI2S	Verbo **haber** - Indicativo tiempo pretérito segunda persona singular
VHXI3P	Verbo **haber** - Indicativo tiempo pretérito tercera persona plural
VHXI3S	Verbo **haber** - Indicativo tiempo pretérito tercera persona singular
VLCI1P	Verbo lexical - Indicativo tiempo condicional primera persona plural
VLCI1S	Verbo lexical - Indicativo tiempo condicional primera persona singular
VLCI2P	Verbo lexical - Indicativo tiempo condicional segunda persona plural
VLCI2S	Verbo lexical - Indicativo tiempo condicional segunda persona singular
VLCI3P	Verbo lexical - Indicativo tiempo condicional tercera persona plural
VLCI3S	Verbo lexical - Indicativo tiempo condicional tercera persona singular
VLFI1P	Verbo lexical - Indicativo tiempo futuro primera persona plural
VLFI1S	Verbo lexical - Indicativo tiempo futuro primera persona singular
VLFI2P	Verbo lexical - Indicativo tiempo futuro segunda persona plural
VLFI2S	Verbo lexical - Indicativo tiempo futuro segunda persona singular
VLFI3P	Verbo lexical - Indicativo tiempo futuro tercera persona plural
VLFI3S	Verbo lexical - Indicativo tiempo futuro tercera persona singular
VLFS1P	Verbo lexical - Subjuntivo tiempo futuro primera persona plural
VLFS1S	Verbo lexical - Subjuntivo tiempo futuro primera persona singular
VLFS2P	Verbo lexical - Subjuntivo tiempo futuro segunda persona plural
VLFS2S	Verbo lexical - Subjuntivo tiempo futuro segunda persona singular
VLFS3P	Verbo lexical - Subjuntivo tiempo futuro tercera persona plural
VLFS3S	Verbo lexical - Subjuntivo tiempo futuro tercera persona singular
VLGER	Verbo lexical - Gerundio
VLII1P	Verbo lexical - Indicativo tiempo imperfecto primera persona plural
VLII1S	Verbo lexical - Indicativo tiempo imperfecto primera persona singular
VLII2P	Verbo lexical - Indicativo tiempo imperfecto segunda persona plural
VLII2S	Verbo lexical - Indicativo tiempo imperfecto segunda persona singular
VLII3P	Verbo lexical - Indicativo tiempo imperfecto tercera persona plural
VLII3S	Verbo lexical - Indicativo tiempo imperfecto tercera persona singular
VLINF	Verbo lexical - Infinitivo
VLIS1P	Verbo lexical - Subjuntivo tiempo imperfecto primera persona plural
VLIS1S	Verbo lexical - Subjuntivo tiempo imperfecto primera persona singular
VLIS2P	Verbo lexical - Subjuntivo tiempo imperfecto segunda persona plural
VLIS2S	Verbo lexical - Subjuntivo tiempo imperfecto segunda persona singular
VLIS3P	Verbo lexical - Subjuntivo tiempo imperfecto tercera persona plural
VLIS3S	Verbo lexical - Subjuntivo tiempo imperfecto tercera persona singular
VLPI1P	Verbo lexical - Indicativo tiempo presente primera persona plural
VLPI1S	Verbo lexical - Indicativo tiempo presente primera persona singular
VLPI2P	Verbo lexical - Indicativo tiempo presente segunda persona plural

VLPI2S	Verbo lexical - Indicativo tiempo presente segunda persona singular
VLPI3P	Verbo lexical - Indicativo tiempo presente tercera persona plural
VLPI3S	Verbo lexical - Indicativo tiempo presente tercera persona singular
VLPM2P	Verbo lexical - Imperativo segunda persona plural
VLPM2S	Verbo lexical - Imperativo segunda persona singular
VLPPFP	Verbo lexical - Participio presente femenino plural
VLPPFS	Verbo lexical - Participio presente femenino singular
VLPPMP	Verbo lexical - Participio presente masculino plural
VLPPMS	Verbo lexical - Participio presente masculino singular
VLPS1P	Verbo lexical - Subjuntivo tiempo presente primera persona plural
VLPS1S	Verbo lexical - Subjuntivo tiempo presente primera persona singular
VLPS2P	Verbo lexical - Subjuntivo tiempo presente segunda persona plural
VLPS2S	Verbo lexical - Subjuntivo tiempo presente segunda persona singular
VLPS3P	Verbo lexical - Subjuntivo tiempo presente tercera persona plural
VLPS3S	Verbo lexical - Subjuntivo tiempo presente tercera persona singular
VLPXFP	Verbo lexical - Participio pasado femenino plural
VLPXFS	Verbo lexical - Participio pasado femenino singular
VLPXMP	Verbo lexical - Participio pasado masculino plural
VLPXMS	Verbo lexical - Participio pasado masculino singular
VLXI1P	Verbo lexical - Indicativo tiempo pretérito primera persona plural
VLXI1S	Verbo lexical - Indicativo tiempo pretérito primera persona singular
VLXI2P	Verbo lexical - Indicativo tiempo pretérito segunda persona plural
VLXI2S	Verbo lexical - Indicativo tiempo pretérito segunda persona singular
VLXI3P	Verbo lexical - Indicativo tiempo pretérito tercera persona plural
VLXI3S	Verbo lexical - Indicativo tiempo pretérito tercera persona singular
VMCI1P	Verbo modal - Indicativo tiempo condicional primera persona plural
VMCI1S	Verbo modal - Indicativo tiempo condicional primera persona singular
VMCI2P	Verbo modal - Indicativo tiempo condicional segunda persona plural
VMCI2S	Verbo modal - Indicativo tiempo condicional segunda persona singular
VMCI3P	Verbo modal - Indicativo tiempo condicional tercera persona plural
VMCI3S	Verbo modal - Indicativo tiempo condicional tercera persona singular
VMFI1P	Verbo modal - Indicativo tiempo futuro primera persona plural
VMFI1S	Verbo modal - Indicativo tiempo futuro primera persona singular
VMFI2P	Verbo modal - Indicativo tiempo futuro segunda persona plural
VMFI2S	Verbo modal - Indicativo tiempo futuro segunda persona singular
VMFI3P	Verbo modal - Indicativo tiempo futuro tercera persona plural
VMFI3S	Verbo modal - Indicativo tiempo futuro tercera persona singular
VMFS1P	Verbo modal - Subjuntivo tiempo futuro primera persona plural
VMFS1S	Verbo modal - Subjuntivo tiempo futuro primera persona singular
VMFS2P	Verbo modal - Subjuntivo tiempo futuro segunda persona plural
VMFS2S	Verbo modal - Subjuntivo tiempo futuro segunda persona singular
VMFS3P	Verbo modal - Subjuntivo tiempo futuro tercera persona plural
VMFS3S	Verbo modal - Subjuntivo tiempo futuro tercera persona singular
VMGER	Verbo modal - Gerundio
VMII1P	Verbo modal - Indicativo tiempo imperfecto primera persona plural
VMII1S	Verbo modal - Indicativo tiempo imperfecto primera persona singular
VMII2P	Verbo modal - Indicativo tiempo imperfecto segunda persona plural
VMII2S	Verbo modal - Indicativo tiempo imperfecto segunda persona singular
VMII3P	Verbo modal - Indicativo tiempo imperfecto tercera persona plural

VMII3S	Verbo modal - Indicativo tiempo imperfecto tercera persona singular
VMINF	Verbo modal - Infinitivo
VMIS1P	Verbo modal - Subjuntivo tiempo imperfecto primera persona plural
VMIS1S	Verbo modal - Subjuntivo tiempo imperfecto primera persona singular
VMIS2P	Verbo modal - Subjuntivo tiempo imperfecto segunda persona plural
VMIS2S	Verbo modal - Subjuntivo tiempo imperfecto segunda persona singular
VMIS3P	Verbo modal - Subjuntivo tiempo imperfecto tercera persona plural
VMIS3S	Verbo modal - Subjuntivo tiempo imperfecto tercera persona singular
VMPI1P	Verbo modal - Indicativo tiempo presente primera persona plural
VMPI1S	Verbo modal - Indicativo tiempo presente primera persona singular
VMPI2P	Verbo modal - Indicativo tiempo presente segunda persona plural
VMPI2S	Verbo modal - Indicativo tiempo presente segunda persona singular
VMPI3P	Verbo modal - Indicativo tiempo presente tercera persona plural
VMPI3S	Verbo modal - Indicativo tiempo presente tercera persona singular
VMPM2P	Verbo modal - Imperativo segunda persona plural
VMPM2S	Verbo modal - Imperativo segunda persona singular
VMPS1P	Verbo modal - Subjuntivo tiempo presente primera persona plural
VMPS1S	Verbo modal - Subjuntivo tiempo presente primera persona singular
VMPS2P	Verbo modal - Subjuntivo tiempo presente segunda persona plural
VMPS2S	Verbo modal - Subjuntivo tiempo presente segunda persona singular
VMPS3P	Verbo modal - Subjuntivo tiempo presente tercera persona plural
VMPS3S	Verbo modal - Subjuntivo tiempo presente tercera persona singular
VMPX	Verbo modal - Participio pasado
VMXI1P	Verbo modal - Indicativo tiempo pretérito primera persona plural
VMXI1S	Verbo modal - Indicativo tiempo pretérito primera persona singular
VMXI2P	Verbo modal - Indicativo tiempo pretérito segunda persona plural
VMXI2S	Verbo modal - Indicativo tiempo pretérito segunda persona singular
VMXI3P	Verbo modal - Indicativo tiempo pretérito tercera persona plural
VMXI3S	Verbo modal - Indicativo tiempo pretérito tercera persona singular
VSCI1P	Verbo ser - Indicativo tiempo condicional primera persona plural
VSCI1S	Verbo ser - Indicativo tiempo condicional primera persona singular
VSCI2P	Verbo ser - Indicativo tiempo condicional segunda persona plural
VSCI2S	Verbo ser - Indicativo tiempo condicional segunda persona singular
VSCI3P	Verbo ser - Indicativo tiempo condicional tercera persona plural
VSCI3S	Verbo ser - Indicativo tiempo condicional tercera persona singular
VSFI1P	Verbo ser - Indicativo tiempo futuro primera persona plural
VSFI1S	Verbo ser - Indicativo tiempo futuro primera persona singular
VSFI2P	Verbo ser - Indicativo tiempo futuro segunda persona plural
VSFI2S	Verbo ser - Indicativo tiempo futuro segunda persona singular
VSFI3P	Verbo ser - Indicativo tiempo futuro tercera persona plural
VSFI3S	Verbo ser - Indicativo tiempo futuro tercera persona singular
VSFS1P	Verbo ser - Subjuntivo tiempo futuro primera persona plural
VSFS1S	Verbo ser - Subjuntivo tiempo futuro primera persona singular
VSFS2P	Verbo ser - Subjuntivo tiempo futuro segunda persona plural
VSFS2S	Verbo ser - Subjuntivo tiempo futuro segunda persona singular
VSFS3P	Verbo ser - Subjuntivo tiempo futuro tercera persona plural
VSFS3S	Verbo ser - Subjuntivo tiempo futuro tercera persona singular
VSGER	Verbo ser - Gerundio
VSII1P	Verbo ser - Indicativo tiempo imperfecto primera persona plural

VSII1S	Verbo ser - Indicativo tiempo imperfecto primera persona singular
VSII2P	Verbo ser - Indicativo tiempo imperfecto segunda persona plural
VSII2S	Verbo ser - Indicativo tiempo imperfecto segunda persona singular
VSII3P	Verbo ser - Indicativo tiempo imperfecto tercera persona plural
VSII3S	Verbo ser - Indicativo tiempo imperfecto tercera persona singular
VSINF	Verbo ser - Infinitivo
VSIS1P	Verbo ser - Subjuntivo tiempo imperfecto primera persona plural
VSIS1S	Verbo ser - Subjuntivo tiempo imperfecto primera persona singular
VSIS2P	Verbo ser - Subjuntivo tiempo imperfecto segunda persona plural
VSIS2S	Verbo ser - Subjuntivo tiempo imperfecto segunda persona singular
VSIS3P	Verbo ser - Subjuntivo tiempo imperfecto tercera persona plural
VSIS3S	Verbo ser - Subjuntivo tiempo imperfecto tercera persona singular
VSPI1P	Verbo ser - Indicativo tiempo presente primera persona plural
VSPI1S	Verbo ser - Indicativo tiempo presente primera persona singular
VSPI2P	Verbo ser - Indicativo tiempo presente segunda persona plural
VSPI2S	Verbo ser - Indicativo tiempo presente segunda persona singular
VSPI3P	Verbo ser - Indicativo tiempo presente tercera persona plural
VSPI3S	Verbo ser - Indicativo tiempo presente tercera persona singular
VSPM2P	Verbo ser - Imperativo segunda persona plural
VSPM2S	Verbo ser - Imperativo segunda persona singular
VSPS1P	Verbo ser - Subjuntivo tiempo presente primera persona plural
VSPS1S	Verbo ser - Subjuntivo tiempo presente primera persona singular
VSPS2P	Verbo ser - Subjuntivo tiempo presente segunda persona plural
VSPS2S	Verbo ser - Subjuntivo tiempo presente segunda persona singular
VSPS3P	Verbo ser - Subjuntivo tiempo presente tercera persona plural
VSPS3S	Verbo ser - Subjuntivo tiempo presente tercera persona singular
VSPX	Verbo ser - Participio pasado
VSXI1P	Verbo ser - Indicativo tiempo pretérito primera persona plural
VSXI1S	Verbo ser - Indicativo tiempo pretérito primera persona singular
VSXI2P	Verbo ser - Indicativo tiempo pretérito segunda persona plural
VSXI2S	Verbo ser - Indicativo tiempo pretérito segunda persona singular
VSXI3P	Verbo ser - Indicativo tiempo pretérito tercera persona plural
VSXI3S	Verbo ser - Indicativo tiempo pretérito tercera persona singular

A.2 Juego de etiquetas extendido del sistema GALENA

La siguiente lista representa el juego de etiquetas utilizado en el experimento con el corpus ITU. Se trata simplemente de la enumeración de las etiquetas que utiliza el sistema GALENA, extendidas con los indicadores que marcan explícitamente las formas verbales compuestas, las formas verbales en voz pasiva y las formas verbales con pronombres clíticos. La descripción de cada etiqueta se puede obtener a partir de la tabla 2.1. El cardinal de este conjunto es de 373 etiquetas.

Afp0	Afpc	Afs0	Afsc	Afy0
Amp0	Ampc	Ams0	Amsc	Amy0
Ayp0	Aypc	Ays0	Aysc	Ayy0
Cc	Cs	Dfp	Dfs	Dmp
Dms	Dns	Enfp	Enfs	Enmp

Enms	Enns	Eyfp	Eyfs	Eymp
Eyms	Eyns	Eyyp	Eyys	Gnyp
Gnys	Gyfp	Gyfs	Gymp	Gyms
Gyyp	Gyys	Gyyy	Idfp	Idfs
Idmp	Idms	Idyp	Idys	Infp
Infs	Inmp	Inms	Inns	Inyp
Inys	Inyy	Iyfp	Iyfs	Iymp
Iyms	Iyyp	Iyys	Iyyy	Mdyp1s
Mdyp2s	Mdyp3p	Mdyp3s	Mdys1s	Mdys2s
Mdys3s	Mnfp1s	Mnfp2s	Mnfp3p	Mnfs1s
Mnfs2s	Mnfs3s	Mnmp1s	Mnmp2s	Mnmp3p
Mnms1s	Mnms2s	Mnms3s	Myfp1p	Myfp2p
Myfs1p	Myfs2p	Mymp1p	Mymp2p	Myms1p
Myms2p	Ncdms	Ncnms	Ncyfp	Ncyfs
Ncymp	Ncyyp	Ncyys	Nmnyp	Nmnys
Nonfp	Nonfs	Nonmp	Nonms	Noyfp
Noyfs	Noymp	Noyms	Npnfp	Npnfs
Npnmp	Npnms	P	Q!	Q¡
Q”	Q(	Q)	Q,	Q-
Q.	Q...	Q:Q;	Q;	Q?
Q¿	Rp1pyy	Rp1syy	Rp2pyy	Rp2syy
Rp3paf	Rp3pam	Rp3pdy	Rp3saf	Rp3sam
Rp3sdy	Rp3yyy	Rt1pqf	Rt1pqm	Rt1sny
Rt1spy	Rt2pqf	Rt2pqm	Rt2sny	Rt2spy
Rt3pqf	Rt3pqm	Rt3sqf	Rt3sqm	Rt3sqy
Rt3ypy	Scfp	Scfs	Scfy	Scmp
Scms	Scmy	Scyp	Scys	Scyy
Spfp	Spfs	Spfy	Spmp	Spms
Spyp	Spys	Spyy	Tdfp	Tdfs
Tdmp	Tdms	Tnyp	Tnys	Tnyy
Tyfp	Tyfs	Tymp	Tyms	Tyyp
Tyys	V000f0	V000f0TC1	V000f0PE1	V000f0PE1TC1
V000f0PE1TCP1	V000f0TCP1	V000g0	V000g0TC1	V000g0PE1
V000g0PE1TC1	V000g0PE1TCP1	V000g0TCP1	V0p0pf	V0p0pfTCP2
V0p0pfTCP3	V0p0pm	V0p0pmTCP2	V0p0pmTCP3	V0s0pf
V0s0pfTCP2	V0s0pfTCP3	V0s0pm	V0s0pmTC2	V0s0pmTCP2
V0s0pmTCP3	V1pci0	V1pci0TC1	V1pci0TCP1	V1pei0
V1pei0TC1	V1pei0TCP1	V1pfi0	V1pfi0TC1	V1pfi0TCP1
V1pfs0	V1pfs0TC1	V1pfs0TCP1	V1pii0	V1pii0TC1
V1pii0TCP1	V1pis0	V1pis0TC1	V1pis0TCP1	V1ppi0
V1ppi0TC1	V1ppi0TCP1	V1pps0	V1pps0TC1	V1pps0TCP1
V1pss0	V1pss0TC1	V1pss0TCP1	V1sei0	V1sei0TC1
V1sei0TCP1	V1sfi0	V1sfi0TC1	V1sfi0TCP1	V1spi0
V1spi0TC1	V1spi0TCP1	V2pci0	V2pci0TC1	V2pci0TCP1
V2pei0	V2pei0TC1	V2pei0TCP1	V2pfi0	V2pfi0TC1
V2pfi0TCP1	V2pfs0	V2pfs0TC1	V2pfs0TCP1	V2pii0
V2pii0TC1	V2pii0TCP1	V2pis0	V2pis0TC1	V2pis0TCP1
V2ppi0	V2ppi0TC1	V2ppi0TCP1	V2ppm0	V2ppm0PE1
V2pps0	V2pps0TC1	V2pps0TCP1	V2pss0	V2pss0TC1

V2pss0TCP1	V2sci0	V2sci0TC1	V2sci0TCP1	V2sei0
V2sei0TC1	V2sei0TCP1	V2sfi0	V2sfi0TC1	V2sfi0TCP1
V2sfs0	V2sfs0TC1	V2sfs0TCP1	V2sii0	V2sii0TC1
V2sii0TCP1	V2sis0	V2sis0TC1	V2sis0TCP1	V2spi0
V2spi0TC1	V2spi0TCP1	V2spm0	V2spm0PE1	V2sps0
V2sps0TC1	V2sps0TCP1	V2sss0	V2sss0TC1	V2sss0TCP1
V3pci0	V3pci0TC1	V3pci0TCP1	V3pei0	V3pei0TC1
V3pei0TCP1	V3pfi0	V3pfi0TC1	V3pfi0TCP1	V3pfs0
V3pfs0TC1	V3pfs0TCP1	V3pii0	V3pii0TC1	V3pii0TCP1
V3pis0	V3pis0TC1	V3pis0TCP1	V3ppi0	V3ppi0TC1
V3ppi0TCP1	V3pps0	V3pps0TC1	V3pps0TCP1	V3pss0
V3pss0TC1	V3pss0TCP1	V3sei0	V3sei0TC1	V3sei0TCP1
V3sfi0	V3sfi0TC1	V3sfi0TCP1	V3spi0	V3spi0TC1
V3spi0TCP1	V3sps0	Vysci0	Vysci0TC1	Vysci0TCP1
Vysfs0	Vysfs0TC1	Vysfs0TCP1	Vysii0	Vysii0TC1
Vysii0TCP1	Vysis0	Vysis0TC1	Vysis0TCP1	Vysps0
Vysps0TC1	Vysps0TCP1	Vysss0	Vysss0TC1	Vysss0TCP1
Wi	Wm	Wn	Wr	Wv
Wy	Y	Ze00	Zefs	Zemp
Zems	Zeys	Zeyy	Zgfp	Zgfs
Zgms	Zo00	Zoms		

A.3 Correspondencia de etiquetas CRATER ⟼ GALENA

La siguiente lista de pares de etiquetas representa la función de correspondencia o *mapping* entre el juego de etiquetas del proyecto CRATER y el juego de etiquetas del sistema GALENA. Este es, por tanto, el *mapping* que fue aplicado al corpus ITU para obtener el corpus de referencia del experimento.

ACRNM → Zgfs	ADJCP → Aypc	ADJCS → Aysc	ADJGFP → Afp0
ADJGFS → Afs0	ADJGMP → Amp0	ADJGMS → Ams0	ADJSFP → Afpc
ADJSFS → Afsc	ADJSMP → Ampc	ADJSMS → Amsc	ADVGR → Wm
ADVGRC → Wy	ADVGRS → Wn	ADVINT → Wv	ADVL → Wn
ADVLD → Wn	ADVLE → Wn	ADVLID → Wn	ADVLIN → Wv
ADVLP → Wn	ADVLR → Wn	ADVLRD → Wr	ADVLRE → Wr
ADVMRE → Wr	ADVN → Wy	ADVNEG → Wn	ADVT → Wn
ADVTIN → Wv	ADVTNE → Wn	ADVTRE → Wr	ALFP → Scfp
ALFS → Scfs	ARCAFS → Iyfs	ARCAMS → Idms	ARQUFP → Iyfp
ARQUMP → Iymp	ARTDFP → Dfp	ARTDFS → Dfs	ARTDMP → Dmp
ARTDMS → Dms	ARTDNS → Dns	CARDFP → Ncyfp	CARDGU → Ays0
CARDMP → Ncymp	CARDPS → Ncnms	CARDXP → Ncyyp	CARDXS → Ncyys
CARNMP → Ncdmp	CC → Cc	CCAD → Cs	CCNEG → Cc
CODE → Scms	CQUE → Cs	CSUBF → Cs	CSUBI → Cs
CSUBX → Cs	DMDPFP → Enfp	DMDPFS → Enfs	DMDPMP → Enmp
DMDPMS → Enms	DMDPNS → Eyns	DMDXFP → Eyfp	DMDXFS → Eyfs
DMDXMP → Eymp	DMDXMS → Eyms	DMPPFP → Enfp	DMPPFS → Enfs
DMPPMP → Enmp	DMPPMS → Enms	DMPPNS → Enns	DMPXFP → Eyfp
DMPXFS → Eyfs	DMPXMP → Eymp	DMPXMS → Eyms	DMRPFP → Enfp

DMRPFS → Enfs
DMRPMP → Enmp
DMRPMS → Enms
DMRPNS → Enns
DMRXFP → Eyfp
DMRXFS → Eyfs
DMRXMP → Eymp
DMRXMS → Eyms
FO → Zf
INTPXP → Gnyp
INTPXS → Gnys
INTXFP → Gyfp
INTXFS → Gyfs
INTXMP → Gymp
INTXMS → Gyms
INTXXP → Gyyp
INTXXS → Gyys
INTXXX → Gyyy
ITJN → Y
NCFP → Scfp
NCFS → Scfs
NCMP → Scmp
NCMS → Scms
NEG → Wn
NMEAFP → Scfp
NMEAFS → Scfs
NMEAMP → Scmp
NMEAMS → Scms
NMON → Scmy
NPAFP → Spfp
NPAFS → Spfs
NPAMP → Spmp
NPAMS → Spms
NPAXX → Spyy
NPGP → Spyp
NPGS → Spys
NPGX → Spyy
NPOS → Spys
NPTOP → Spyp
NPTOS → Spys
NPTP → Spyp
NPTS → Spys
NWEE → Scmy
ORDNMS → Nonms
ORDXFP → Nonfp
ORDXFS → Nonfs
ORDXMP → Nonmp
ORDXMS → Nonms
PAL → P Dms
PDEL → P Dms
PE → Ze00
PNC → U
PPC1P → Re1pyy
PPC1S → Re1syy
PPC2P → Re2pyy
PPC2S → Re2syy
PPC3P → Re3pdy
PPC3S → Re3sdy
PPN1S → Rt1sny
PPN2S → Rt2sny
PPO3FP → Re3paf
PPO3FS → Re3saf
PPO3MP → Re3pam
PPO3XS → Re3sam
PPOSFP → Mnfp2s
PPOSFS → Mnfs2s
PPOSMP → Mnmp2s
PPOSMS → Mnms3y
PPOSPP → Mdyp3s
PPOSPS → Mdys3s
PPP1S → Rt1spy
PPP2S → Rt2spy
PPP3X → Rt3ypy
PPX1FP → Rt1pqf
PPX1MP → Rt1pqm
PPX2FP → Rt2pqf
PPX2MP → Rt2pqm
PPX3FP → Rt3pqf
PPX3FS → Rt3sqf
PPX3MP → Rt3pqm
PPX3MS → Rt3sqm
PPX3NS → Rt3sqn
PPXT2P → Rt300y
PPXT2S → Rt300y
PREP → P
PREPN → P
QUDF → Idfp
QUDM → Idmp
QUDX → Iyys
QUNFP → Idfp
QUNFS → Idys
QUNMP → Idmp
QUNMS → Idms
QUNNMS → Idms
QUPA → Inms
QUPI → Inns
QUPMUL → Nmnys
QUPNA → Inms
QUPNI → Inns
QUPNX → Inms
QUXFP → Iyfp
QUXFS → Iyfs
QUXMP → Iymp
QUXMS → Iyms
QUXNFP → Iyfp
QUXNFS → Iyfs
QUXNMP → Iymp
RELPFP → Tdfp
RELPFS → Tdfs
RELPMP → Tdmp
RELPMS → Tdms
RELPXP → Tnyp
RELPXS → Tnys
RELXFP → Tyfp
RELXFS → Tyfs
RELXMP → Tymp
RELXMS → Tyms
ROMAN → Ncyyp
SE → Rp3yyy
TRATF → Afs0
TRATM → Ams0
UMFX → Scfy
UMMX → Scmy
VECI1P → V1pci0
VECI1S → V1sci0
VECI2P → V2pci0
VECI2S → V2sci0
VECI3P → V3pci0
VECI3S → V3sci0
VEFI1P → V1pfi0
VEFI1S → V1sfi0
VEFI2P → V2pfi0
VEFI2S → V2sfi0
VEFI3P → V3pfi0
VEFI3S → V3sfi0
VEFS1P → V1pfs0
VEFS1S → V1sfs0
VEFS2P → V2pfs0
VEFS2S → V2sfs0
VEFS3P → V3pfs0
VEFS3S → V3sfs0
VEGER → V000g0
VEII1P → V1pii0
VEII1S → V1sii0
VEII2P → V2pii0
VEII2S → V2sii0
VEII3P → V3pii0
VEII3S → V3sii0
VEINF → V000f0
VEIS1P → V1pss0
VEIS1S → V1sss0
VEIS2P → V2pss0
VEIS2S → V2sss0
VEIS3P → V3pss0
VEIS3S → V3sss0
VEPI1P → V1ppi0
VEPI1S → V1spi0
VEPI2P → V2ppi0
VEPI2S → V2spi0
VEPI3P → V3ppi0
VEPI3S → V3spi0
VEPM2P → V2ppm0
VEPM2S → V2spm0
VEPS1P → V1pps0
VEPS1S → V1sps0
VEPS2P → V2pps0
VEPS2S → V2sps0
VEPS3P → V3pps0
VEPS3S → V3sps0
VEPX → V0s0pm
VEXI1P → V1pei0
VEXI1S → V1sei0
VEXI2P → V2pei0
VEXI2S → V2sei0
VEXI3P → V3pei0
VEXI3S → V3sei0
VHCI1P → V1pci0
VHCI1S → V1sci0
VHCI2P → V2pci0
VHCI2S → V2sci0
VHCI3P → V3pci0
VHCI3S → V3sci0
VHFI1P → V1pfi0
VHFI1S → V1sfi0
VHFI2P → V2pfi0
VHFI2S → V2sfi0
VHFI3P → V3pfi0
VHFI3S → V3sfi0
VHFS1P → V1pfs0
VHFS1S → V1sfs0
VHFS2P → V2pfs0
VHFS2S → V2sfs0
VHFS3P → V3pfs0
VHFS3S → V3sfs0
VHGER → V000g0
VHII1P → V1pii0

VHII1S → V1sii0 VHII2P → V2pii0 VHII2S → V2sii0 VHII3P → V3pii0
VHII3S → V3sii0 VHINF → V000f0 VHIS1P → V1pss0 VHIS1S → V1sss0
VHIS2P → V2pss0 VHIS2S → V2sss0 VHIS3P → V3pss0 VHIS3S → V3sss0
VHPI1P → V1ppi0 VHPI1S → V1spi0 VHPI2P → V2ppi0 VHPI2S → V2spi0
VHPI3E → V3spi0 VHPI3P → V3ppi0 VHPI3S → V3spi0 VHPM2P → V2ppm0
VHPM2S → V2spm0 VHPS1P → V1pps0 VHPS1S → V1sps0 VHPS2P → V2pps0
VHPS2S → V2sps0 VHPS3P → V3pps0 VHPS3S → V3sps0 VHPXFP → V0p0pf
VHPXFS → V0s0pf VHPXMP → V0p0pm VHPXMS → V0s0pm VHXI1P → V1pei0
VHXI1S → V1sei0 VHXI2P → V2pei0 VHXI2S → V2sei0 VHXI3P → V3pei0
VHXI3S → V3sei0 VLCI1P → V1pci0 VLCI1S → V1sci0 VLCI2P → V2pci0
VLCI2S → V2sci0 VLCI3P → V3pci0 VLCI3S → V3sci0 VLFI1P → V1pfi0
VLFI1S → V1sfi0 VLFI2P → V2pfi0 VLFI2S → V2sfi0 VLFI3P → V3pfi0
VLFI3S → V3sfi0 VLFS1P → V1pfs0 VLFS1S → V1sfs0 VLFS2P → V2pfs0
VLFS2S → V2sfs0 VLFS3P → V3pfs0 VLFS3S → V3sfs0 VLGER → V000g0
VLII1P → V1pii0 VLII1S → V1sii0 VLII2P → V2pii0 VLII2S → V2sii0
VLII3P → V3pii0 VLII3S → V3sii0 VLINF → V000f0 VLIS1P → V1pss0
VLIS1S → V1sss0 VLIS2P → V2pss0 VLIS2S → V2sss0 VLIS3P → V3pss0
VLIS3S → V3sss0 VLPI1P → V1ppi0 VLPI1S → V1spi0 VLPI2P → V2ppi0
VLPI2S → V2spi0 VLPI3P → V3ppi0 VLPI3S → V3spi0 VLPM2P → V2ppm0
VLPM2S → V2spm0 VLPPFP → V0p0pf VLPPFS → V0s0pf VLPPMP → V0p0pm
VLPPMS → V0s0pm VLPS1P → V1pps0 VLPS1S → V1sps0 VLPS2P → V2pps0
VLPS2S → V2sps0 VLPS3P → V3pps0 VLPS3S → V3sps0 VLPXFP → V0p0pf
VLPXFS → V0s0pf VLPXMP → V0p0pm VLPXMS → V0s0pm VLXI1P → V1pei0
VLXI1S → V1sei0 VLXI2P → V2pei0 VLXI2S → V2sei0 VLXI3P → V3pei0
VLXI3S → V3sei0 VMCI1P → V1pci0 VMCI1S → V1sci0 VMCI2P → V2pci0
VMCI2S → V2sci0 VMCI3P → V3pci0 VMCI3S → V3sci0 VMFI1P → V1pfi0
VMFI1S → V1sfi0 VMFI2P → V2pfi0 VMFI2S → V2sfi0 VMFI3P → V3pfi0
VMFI3S → V3sfi0 VMFS1P → V1pfs0 VMFS1S → V1sfs0 VMFS2P → V2pfs0
VMFS2S → V2sfs0 VMFS3P → V3pfs0 VMFS3S → V3sfs0 VMGER → V000g0
VMII1P → V1pii0 VMII1S → V1sii0 VMII2P → V2pii0 VMII2S → V2sii0
VMII3P → V3pii0 VMII3S → V3sii0 VMINF → V000f0 VMIS1P → V1pss0
VMIS1S → V1sss0 VMIS2P → V2pss0 VMIS2S → V2sss0 VMIS3P → V3pss0
VMIS3S → V3sss0 VMPI1P → V1ppi0 VMPI1S → V1spi0 VMPI2P → V2ppi0
VMPI2S → V2spi0 VMPI3P → V3ppi0 VMPI3S → V3spi0 VMPM2P → V2ppm0
VMPM2S → V2spm0 VMPS1P → V1pps0 VMPS1S → V1sps0 VMPS2P → V2pps0
VMPS2S → V2sps0 VMPS3P → V3pps0 VMPS3S → V3sps0 VMPX → V0s0pm
VMXI1P → V1pei0 VMXI1S → V1sei0 VMXI2P → V2pei0 VMXI2S → V2sei0
VMXI3P → V3pei0 VMXI3S → V3sei0 VSCI1P → V1pci0 VSCI1S → V1sci0
VSCI2P → V2pci0 VSCI2S → V2sci0 VSCI3P → V3pci0 VSCI3S → V3sci0
VSFI1P → V1pfi0 VSFI1S → V1sfi0 VSFI2P → V2pfi0 VSFI2S → V2sfi0
VSFI3P → V3pfi0 VSFI3S → V3sfi0 VSFS1P → V1pfs0 VSFS1S → V1sfs0
VSFS2P → V2pfs0 VSFS2S → V2sfs0 VSFS3P → V3pfs0 VSFS3S → V3sfs0
VSGER → V000g0 VSII1P → V1pii0 VSII1S → V1sii0 VSII2P → V2pii0
VSII2S → V2sii0 VSII3P → V3pii0 VSII3S → V3sii0 VSINF → V000f0
VSIS1P → V1pss0 VSIS1S → V1sss0 VSIS2P → V2pss0 VSIS2S → V2sss0
VSIS3P → V3pss0 VSIS3S → V3sss0 VSPI1P → V1ppi0 VSPI1S → V1spi0
VSPI2P → V2ppi0 VSPI2S → V2spi0 VSPI3P → V3ppi0 VSPI3S → V3spi0
VSPM2P → V2ppm0 VSPM2S → V2spm0 VSPS1P → V1pps0 VSPS1S → V1sps0
VSPS2P → V2pps0 VSPS2S → V2sps0 VSPS3P → V3pps0 VSPS3S → V3sps0

VSPX → V0s0pm VSXI1P → V1pei0 VSXI1S → V1sei0 VSXI2P → V2pei0
VSXI2S → V2sei0 VSXI3P → V3pei0 VSXI3S → V3sei0

A.4 Juego de etiquetas del corpus SUSANNE

La siguiente lista constituye el juego de etiquetas utilizado en el corpus SUSANNE. La lista consta de tres columnas. La primera columna es el número de la etiqueta. Este número ha sido asignado simplemente de manera ascendente sobre el orden alfabético de las etiquetas. Por tanto, su única utilidad radica en poder identificar las etiquetas a lo largo de los ejemplos que aparecen en el texto, donde por razones de ahorro de espacio se suele utilizar dicho número en lugar del nombre completo de la etiqueta. La segunda columna es la etiqueta en sí y la tercera columna contiene la descripción de la etiqueta tomada directamente de [Sampson 1994b] y [Garside *et al.* 1987], y a veces alguna palabra de ejemplo. El cardinal de este conjunto es de 425 etiquetas.

1)	APPGf	her como posesivo ≠ PPHO1f
2)	APPGh1	its
3)	APPGh2	their
4)	APPGi1	my como posesivo
5)	APPGi2	our
6)	APPGm	his excepto como pronombre ≠ PPGm
7)	APPGy	your
8)	AT	the como determinante
9)	AT1	Artículo indefinido a, an
10)	AT1e	every
11)	ATn	no como determinante o calificador ≠ UH
12)	BTO21	in en la secuencia in order + infinitivo
13)	BTO22	order en la secuencia in order + infinitivo
14)	CC	Conjunción coordinada
15)	CC31	Conjunción coordinada de 3 palabras (1a. palabra), e.g. as well as
16)	CC32	Conjunción coordinada de 3 palabras (2a. palabra), e.g. as well as
17)	CC33	Conjunción coordinada de 3 palabras (3a. palabra), e.g. as well as
18)	CCB	but como conjunción coordinada ≠ ICSx RR
19)	CCn	nor
20)	CCr	or
21)	CS	Conjunción subordinada
22)	CS21	Conjunción subordinada de 2 palabras (1a. palabra), e.g. even though
23)	CS22	Conjunción subordinada de 2 palabras (2a. palabra), e.g. even though
24)	CS31	Conjunción subordinada de 3 palabras (1a. palabra), e.g. as long as
25)	CS32	Conjunción subordinada de 3 palabras (2a. palabra), e.g. as long as
26)	CS33	Conjunción subordinada de 3 palabras (3a. palabra), e.g. as long as
27)	CSA	as como conjunción subordinada o como preposición en sentido comparativo ≠ IIa RGa
28)	CSN	than en todos los usos
29)	CST	that como conjunción subordinada ≠ DD1a
30)	CST21	as how
31)	CST22	as how
32)	CSW	whether en todos los usos

33)	CSf	for como conjunción ≠ IF
34)	CSg	though como conjunción subordinada ≠ RR
35)	CSi	if
36)	CSk21	as if, as though
37)	CSk22	as if, as though
38)	CSn	when como conjunción subordinada ≠ RRQq RRQr
39)	CSr	where como conjunción subordinada ≠ RRQq RRQr
40)	DA1	much, little ≠ little JJ
41)	DA2	many, few, en todos los usos
42)	DA2R	fewer
43)	DA2q	several
44)	DAR	more, less, en todos los usos excpeto less II
45)	DAT	most, least, en todos los usos
46)	DAg	own como parte de una construcción de genitivo ≠ VV0v
47)	DAr	former, latter, en todos los usos
48)	DAy	same, selfsame
49)	DAz	such en todos los usos
50)	DB2	both como determinante o pronombre ≠ LE RR
51)	DBa	all como determinante o pronombre ≠ NN1c RR FB
52)	DBh	half como determinante o pronombre ≠ NN1c RR
53)	DD1a	that como determinante, pronombre demostrativo o calificador ≠ CST
54)	DD1b21	a bit
55)	DD1b22	a bit
56)	DD1e	either como determinante o pronombre ≠ LEe RR
57)	DD1i	this en todos los usos incluido como calificador, e.g. this big
58)	DD1n	neither como determinante o pronombre ≠ LEn RR
59)	DD1q	another, each, como determinante o pronombre ≠ each RAq
60)	DD1q41	one and the same
61)	DD1q42	one and the same
62)	DD1q43	one and the same
63)	DD1q44	one and the same
64)	DD1t21	a little
65)	DD1t22	a little
66)	DD21	the rest
67)	DD22	the rest
68)	DD221	a few
69)	DD222	a few
70)	DD231	a good few, a good many, a great many
71)	DD232	a good few, a good many, a great many
72)	DD233	a good few, a good many, a great many
73)	DD2a	those
74)	DD2i	these
75)	DDQ	what
76)	DDQGq	whose en usos interrogativos ≠ DDQGr
77)	DDQGr	whose en usos relativos ≠ DDQGq
78)	DDQV	whichever, whatever, whichsoever, whatsoever ≠ whatever RAn whatsoever RAn
79)	DDQV31	no matter which, no matter what
80)	DDQV32	no matter which, no matter what

81)	DDQV33	`no matter which, no matter what`
82)	DDQq	`which` en usos interrogativos ≠ DDQr
83)	DDQr	`which` en usos relativos ≠ DDQq
84)	DDf	`enough` como pronombre o pre/postmodificando un nombre ≠ RGAf RRe
85)	DDi	`some` como determinante o pronombre ≠ RGi
86)	DDo21	`a lot`
87)	DDo22	`a lot`
88)	DDy	`any` como determinante o pronombre
89)	EX	`there` existencial ≠ RLh UH
90)	FA	Sufijo (si se ha etiquetado de manera separada, e.g. porque va unido a la raíz mediante un guión)
91)	FB	Prefijo (si se ha etiquetado de manera separada, e.g. porque va unido a la raíz mediante un guión)
92)	FO	Fórmula indeterminada
93)	FOc	Fórmula o sigla para una substancia química, molécula o partícula subatómica, e.g. `TNT`, `C-14`, H_2O
94)	FOqx	Ecuación algebraica, cuando se analiza como una sola palabra
95)	FOs	Número de registro/referencia/serie/modelo
96)	FOx	Expresión algebraica nominal, e.g. a, π, dy/dx
97)	FW	Palabra extranjera imposible de etiquetar de manera más específica haciendo referencia a su contexto en inglés, e.g. porque aparece en una frase que está toda en otro idioma, o en un título gramatical independiente del contexto
98)	FWg	Nombre biológico en Latín de género o de orden más alto que las especies, e.g. `Equus Umbelliferae`
99)	FWs	Nombre de especie biológica en Latín, e.g. `sapiens`
100)	GG	Inflexión de genitivo germánico: `'s` o sólo `'` después de un plural o de determinadas terminaciones en `s`
101)	ICS	`considering`, `notwithstanding` ≠ `considering` VVGt RR, `notwithstanding` RR
102)	ICSk	`like` como preposición o conjunción subordinada ≠ FA JB NN1c VV0t
103)	ICSt	`after`, `before`, `ere`, `since`, `until`, `till` como preposición (con complemento) o conjunción subordinada ≠ `after` RR FB, `before` RR, `since` RAa RR
104)	ICSx	`but`, `except`, `save` como preposición o conjunción subordinada ≠ `but` CCB RR, `except` VV0t, `save` NN1c VV0v
105)	IF	`for` como preposición ≠ CSf
106)	II	Preposición, incluido el uso preposicional de las palabras que pueden funcionar como preposición o como adverbio
107)	II21	Locución prepositiva de 2 palabras (1a. palabra)
108)	II22	Locución prepositiva de 2 palabras (2a. palabra)
109)	II31	Locución prepositiva de 3 palabras (1a. palabra)
110)	II32	Locución prepositiva de 3 palabras (2a. palabra)
111)	II33	Locución prepositiva de 3 palabras (3a. palabra)
112)	II41	Locución prepositiva de 4 palabras (1a. palabra)
113)	II42	Locución prepositiva de 4 palabras (2a. palabra)
114)	II43	Locución prepositiva de 4 palabras (3a. palabra)
115)	II44	Locución prepositiva de 4 palabras (4a. palabra)
116)	IIa	`as` usado no comparativamente como preposición ≠ CSA RGa

117)	IIb	by como preposición ≠ RL
118)	IIp	per
119)	IIt	to como preposición ≠ TO RL
120)	IIx	Operador matemático infijo
121)	IO	of
122)	IW	with en todos los usos, without como preposición ≠ RR
123)	IW21	what with
124)	IW22	what with
125)	JA	Adjetivo usado sólo de forma predicativa, e.g. alone, unable
126)	JA21	Adjetivo predicativo de 2 palabras (1a. palabra)
127)	JA22	Adjetivo predicativo de 2 palabras (2a. palabra)
128)	JB	Adjetivo usado sólo de forma atributiva, e.g. chief, inverse
129)	JBR	inner, lesser, nether, outer, upper ≠ upper NN1c
130)	JBT	utmost, uttermost ≠ NN1c en ambos casos
131)	JBo	other en todos los usos
132)	JBy	only como adjetivo (e.g. the only thing) ≠ RRx
133)	JJ	Adjetivo general, e.g. blue, Jewish
134)	JJ21	Adjetivo general de 2 palabras (1a. palabra)
135)	JJ22	Adjetivo general de 2 palabras (2a. palabra)
136)	JJR	Adjetivo comparativo incluidos elder, further
137)	JJT	Adjetivo superlativo incluidos eldest, furthest
138)	JJh	Pseudo-adjetivo formado mediante el sufijo ed en la última palabra de un nombre compuesto
139)	JJj	Adjetivo abreviado incorporado a un nombre de organización para indicar su estado legal, e.g. Ltd, Inc, Pty
140)	JJs	Adjetivo terminado en most excepto utmost, uttermost
141)	LE	both como precoordinador ≠ DB2
142)	LE21	not only
143)	LE22	not only
144)	LEe	either como precoordinador ≠ DD1e RR
145)	LEn	neither como precoordinador ≠ DD1n RR
146)	MC	Cardinal numeral escrito con letras desde zero en adelante, incluyendo umpteen
147)	MC1	one como numeral escrito con letras, incluyendo usos como en one of the major items, o como lugar de colocación del nombre de una frase nominal (e.g. a large one) ≠ PN1o
148)	MC1n	1 escrito como dígito, incluyendo su uso como ordinal, e.g. February 1
149)	MC2	Cardinal en plural escrito con dígitos, e.g. 1s, 2s, 10s, 10's
150)	MC2y	Nombre de año en plural, e.g. 1960s
151)	MCb	Etiquetas utilizadas para hacer referencias cruzadas entre distintos elementos del texto, que consisten en combinaciones de dígitos, letras y/o caracteres no alfanuméricos, e.g. (a), 1990b, 1a, IIc, (1990)
152)	MCd	Numeral con punto decimal
153)	MCe	Número que contiene un separador no decimal
154)	MCn	Cardinal numeral escrito con dígitos, incluyendo su uso como ordinal aunque no esté marcado como tal (e.g. February 28), incluyendo fracciones, e incluyendo números negativos
155)	MCo	0 escrito como dígito
156)	MCr	Número romano (desde I en adelante)

157) MCy Nombre de año escrito con dígitos, completo o con apóstrofe, e.g. 1987 o '91
158) MD Ordinal (e.g. third, fourth) si se usa como adjetivo o adverbio ordinal o como fracción, incluyendo umpteenth y nth, pero no half ni quarter, que son fracciones pero no ordinales (half es DBh NN1c RR, y quarter es NNL1c NNT1c NNU1c)
159) MDn Número ordinal escrito con dígitos, e.g. 1st, 2nd, 100th
160) MDo first, second ≠ second NNT1c VV0t
161) MDt next, last ≠ last NN1c VV0i
162) MFn Fracción escrita con dígitos, e.g. 2/3
163) ND1 Dirección, e.g. north, N, southeast
164) NN1c Nombre contable singular, e.g. clock, crowd, Buddhist
165) NN1c21 Nombre contable singular de 2 palabras (1a. palabra)
166) NN1c22 Nombre contable singular de 2 palabras (2a. palabra)
167) NN1n Nombre incontable o contable singular, e.g. illness, German
168) NN1u Nombre incontable, e.g. snow, Buddhism
169) NN1u21 Nombre incontable de 2 palabras (1a. palabra)
170) NN1u22 Nombre incontable de 2 palabras (2a. palabra)
171) NN1ux Nombre incontable terminado en ics, e.g. athletics, thermodynamics
172) NN2 Nombre plural, e.g. clocks, trousers, Germans, police
173) NN221 Nombre plural de 2 palabras (1a. palabra)
174) NN222 Nombre plural de 2 palabras (2a. palabra)
175) NNJ Nombre descriptor de organización (singular o plural), e.g. Corp, Bros
176) NNJ1c Nombre descriptor de organización, contable singular, e.g. club, committe, institute
177) NNJ1n Nombre descriptor de organización, incontable o contable singular, e.g. company, organization, society
178) NNJ2 Nombre descriptor de organización plural, e.g. associates
179) NNL Nombre descriptor de lugar, singular, plural o neutro, e.g. Rd, Is, Mts
180) NNL1c Nombre descriptor de lugar, contable singular, e.g. road, island, city
181) NNL1cb Nombre descriptor de lugar, contable singular, con tendencia a preceder al nombre específico, e.g. camp, lake, mount
182) NNL1n Nombre descriptor de lugar, incontable o contable singular, e.g. drive, water, green
183) NNL2 Nombre descriptor de lugar, plural, e.g. buildings, mountains, springs
184) NNOc dozen, score, gross, hundred, thousand, million, billion, trillion, etc., zillion
185) NNS Nombre de estilo o título, e.g. Mr, Mrs, Miss, Dr, Dom, Mme, Gen, Sen, Sir, Missy, Ma'am, Sire, Mister
186) NNS1c Nombre de estilo o título, contable singular, e.g. doctor, miss, madam, minister, prince, queen
187) NNS1c21 Nombre de estilo o título, contable singular, de 2 palabras (1a. palabra)
188) NNS1c22 Nombre de estilo o título, contable singular, de 2 palabras (2a. palabra)
189) NNS1n Nombre de estilo o título, incontable o contable singular, e.g. justice
190) NNS2 Nombre de estilo o título, plural, e.g. doctors, madams, ministers, princes, queens, justices
191) NNS21 Nombre de estilo o título, plural, de 2 palabras (1a. palabra)
192) NNS22 Nombre de estilo o título, plural, de 2 palabras (2a. palabra)
193) NNSA Nombre de estilo o título después de un nombre, contable o no, e.g. Jr,

		Sr, Jun, Sen, Esq, Bart, BA, PhD, MP, QC, VC, Dem, Lab
194)	NNSS	Nombre de estilo o título, plural, e.g. Messrs, Mmes, Mesdames, Sirs
195)	NNSj	Nombre de estilo o título de uso adjetival
196)	NNT1c	Nombre de tiempo en singular que puede encabezar una frase nominal funcionando adverbialmente (distinto de NNT1h), e.g. hour, day, night, morning, afternoon, evening, week, year, century, time
197)	NNT1h	Nombre de estación o periodo vacacional, e.g. Christmas, Boxing Day, Hallowe'en summer
198)	NNT1h21	Nombre de estación o periodo vacacional, de 2 palabras (1a. palabra)
199)	NNT1h22	Nombre de estación o periodo vacacional, de 2 palabras (2a. palabra)
200)	NNT1m	Nombre de momento temporal que no puede encabezar una frase nominal funcionando adverbialmente, e.g. noon, midnight, midyear
201)	NNT2	Plural de cualquier nombre NNT1
202)	NNU	Nombre de unidad, singular, plural o neutro, e.g. in, ins
203)	NNU1c	Nombre de unidad, contable singular, e.g. inch, kilogram
204)	NNU1n	Nombre de unidad, incontable o contable singular, e.g. metre
205)	NNU2	Nombre de unidad, plural, e.g. inches
206)	NNU21	Nombre de unidad, plural, de 2 palabras (1a. palabra)
207)	NNU22	Nombre de unidad, plural, de 2 palabras (2a. palabra)
208)	NNUb	Símbolo de unidad que precede a un numeral, e.g. $
209)	NNUc	Nombre de unidad, contable plural, e.g. hertz, yen
210)	NNUp	%, percent
211)	NNUp21	per cent
212)	NNUp22	per cent
213)	NNa	Hora del día escrita con dígitos, e.g. 10:30, 10.30, 1030
214)	NNb	Nombre común atributivo, e.g. scissor, trouser, pincer
215)	NNc	Nombre contable plural, e.g. sheep, species, people, Swiss
216)	NNm	Abreviatura singular normalmente seguida de un numeral funcionando como nombre propio, e.g. Fig., No., p., #
217)	NNmm	NNm en plural, e.g. Figs., Nos., pp.
218)	NNn	Nombre incontable o contable, singular o plural, e.g. fish, Chinese
219)	NNu	Nombre incontable plural, e.g. data, measles, bowls, French
220)	NNux	Nombre incontable plural terminado en ics, e.g. mechanics
221)	NP1c	Nombre propio de país
222)	NP1c21	Nombre propio de país de 2 palabras (1a. palabra)
223)	NP1c22	Nombre propio de país de 2 palabras (2a. palabra)
224)	NP1f	Nombre propio de pila femenino
225)	NP1g	Nombre propio geográfico, por ejemplo Adriatic America Sherwood
226)	NP1i	Nombre propio inicial de un nombre personal, como por ejemplo W. o G. en W.G. Grace
227)	NP1j	Nombre propio de organización, por ejemplo Unilever Kiwanis CIA
228)	NP1j31	Nombre propio de organización de 3 palabras (1a. palabra)
229)	NP1j32	Nombre propio de organización de 3 palabras (2a. palabra)
230)	NP1j33	Nombre propio de organización de 3 palabras (3a. palabra)
231)	NP1m	Nombre propio de pila masculino
232)	NP1p	Nombre propio de *provincia*, por ejemplo un estado americano o un condado inglés
233)	NP1p21	Nombre propio de *provincia* de 2 palabras (1a. palabra)
234)	NP1p22	Nombre propio de *provincia* de 2 palabras (2a. palabra)

235)	NP1s	Nombre propio apellido singular
236)	NP1t	Nombre propio de ciudad, suburbio o pueblo
237)	NP1t21	Nombre propio de ciudad, suburbio o pueblo de 2 palabras (1a. palabra)
238)	NP1t22	Nombre propio de ciudad, suburbio o pueblo de 2 palabras (2a. palabra)
239)	NP1x	Nombre propio singular misceláneo, por ejemplo `Parthenon Persil`
240)	NP1z	Nombre propio en código funcionando como un nombre contable, por ejemplo `GTi` o `PDP11/70`
241)	NP2c	Nombre propio de país en plural
242)	NP2g	Nombre propio geográfico en plural
243)	NP2p	Nombre propio de *provincia* en plural
244)	NP2s	Nombre propio apellido plural
245)	NP2z	Nombre propio en código funcionando como un nombre contable en plural
246)	NPD1	Nombre propio de día de la semana, por ejemplo `Sunday`
247)	NPD2	Nombre propio de día de la semana en plural
248)	NPM	Nombre propio de mes abreviado, por ejemplo `Oct.`
249)	NPM1	Nombre propio de mes, por ejemplo `October`
250)	PN	`none` en todos los usos
251)	PN1	`anybody`, `anyone`, `anything`, `everybody`, `everyone`, `everything`, `naught`, `nobody`, `nothing`, `somebody`, `someone`, `something` ≠ `nobody` NN1c
252)	PN121	`no` one
253)	PN122	`no` one
254)	PN1o	`one` como pronombre impersonal ≠ MC1
255)	PN1z	`so` como proforma, e.g. en construcciones como `do so` ≠ RGz RRz
256)	PNQOr	`whom` relativo
257)	PNQSq	`who` interrogativo ≠ PNQSr
258)	PNQSr	`who` relativo ≠ PNQSq
259)	PNQVS	`whoever`, `whosoever`
260)	PPGf	`hers`
261)	PPGh2	`theirs`
262)	PPGi1	`mine` como pronombre
263)	PPGi2	`ours`
264)	PPGm	`his` como pronombre ≠ APPGm
265)	PPGy	`yours`
266)	PPH1	`it`
267)	PPHO1f	`her` como pronombre ≠ APPGf
268)	PPHO1m	`him`
269)	PPHO2	`them`
270)	PPHS1f	`she`
271)	PPHS1m	`he`
272)	PPHS2	`they`
273)	PPIO1	`me`
274)	PPIO2	`us`
275)	PPIS1	`I` como prononbre personal
276)	PPIS2	`we`
277)	PPX1f	`herself`
278)	PPX1h	`itself`
279)	PPX1i	`myself`
280)	PPX1m	`himself`
281)	PPX1y	`yourself`

282)	PPX2h	themselves
283)	PPX2h21	each other, one another
284)	PPX2h22	each other, one another
285)	PPX2i	ourselves, ourself de tratamiento real
286)	PPX2y	yourselves
287)	PPY	you
288)	RAa	ago, since como sinónimo de ago ≠ since ICSt RR
289)	RAc	Elemento de cierre de coordinación, e.g. etc, respectively
290)	RAc21	et al
291)	RAc22	et al
292)	RAc31	and so on
293)	RAc32	and so on
294)	RAc33	and so on
295)	RAe	else en todos los usos
296)	RAh	am (*ante meridiem*), pm, o'clock
297)	RAj	Adjetivo posnominal, e.g. designate, centigrade, elect
298)	RAn	whatever, whatsoever, después de un encabezamiento nominal negativo o no afirmativo (e.g. any ...) ≠ DDQV
299)	RAp21	per diem
300)	RAp22	per diem
301)	RAq	apiece, each en usos distributivos ≠ each DD1q
302)	REX	Elemento introductorio, e.g. namely, eg, ie, viz
303)	REX21	for example, for instance, that is
304)	REX22	for example, for instance, that is
305)	RG	Calificador sin ningún otro uso adverbial, e.g. jolly, mighty, stark ≠ JJ
306)	RG21	far from ≠ II
307)	RG22	far from ≠ II
308)	RGA	indeed como calificador ≠ RR
309)	RGAf	enough como calificador ≠ DDf RRe
310)	RGQV	however como calificador ≠ RR RRQV
311)	RGQV31	no matter how como calificador ≠ RRQV
312)	RGQV32	no matter how como calificador ≠ RRQV
313)	RGQV33	no matter how como calificador ≠ RRQV
314)	RGQq	how como calificador ≠ RRQq RRQr
315)	RGa	as como calificador ≠ CSA IIa
316)	RGb	quite como calificador o antes de artículo (e.g. quite a good idea) ≠ RR
317)	RGf	too como calificador ≠ RR
318)	RGi	about, around, circa, over, some, under, utilizados con cantidad o número ≠ about II RPK, around II RL, over FB II RP, some DDi, under FB II RL
319)	RGi21	up to ≠ II
320)	RGi22	up to ≠ II
321)	RGr	rather como calificador o antes de artículo (e.g. rather a good idea) ≠ RRR
322)	RGr21	more of antes de artículo (e.g. more of an agricultural nation)
323)	RGr22	more of
324)	RGz	so como calificador ≠ RRz PN1z
325)	RL	Adverbio de lugar o dirección

326)	RL21	Adverbio de lugar o dirección de 2 palabras (1a. palabra)
327)	RL22	Adverbio de lugar o dirección de 2 palabras (2a. palabra)
328)	RLe	elsewhere
329)	RLh	here, there como adverbio de lugar ≠ there EX UH
330)	RLn	downstairs, upstairs ≠ JJ NN1u en ambos casos
331)	RLw	somewhere, someplace, anywhere, anyplace, everywhere, nowhere
332)	RP	Usos adberviales de across, down, in, off, on, out, over, through, up ≠ II en cada caso y ≠ down JB NNL1n VV0t, off FB JB, out JB VV0t, over FB JB NN1c RGi, through JB, up VV0v
333)	RPK	about en uso adverbial y encadenado (e.g. about to) ≠ II RGi
334)	RR	Adverbio general, e.g. fast, skilfully
335)	RR21	Adverbio general de 2 palabras (1a. palabra)
336)	RR22	Adverbio general de 2 palabras (2a. palabra)
337)	RR31	Adverbio general de 3 palabras (1a. palabra)
338)	RR32	Adverbio general de 3 palabras (2a. palabra)
339)	RR33	Adverbio general de 3 palabras (3a. palabra)
340)	RR41	Adverbio general de 4 palabras (1a. palabra)
341)	RR42	Adverbio general de 4 palabras (2a. palabra)
342)	RR43	Adverbio general de 4 palabras (3a. palabra)
343)	RR44	Adverbio general de 4 palabras (4a. palabra)
344)	RRQV	Adverbio wh...ever, e.g. however, whencesoever, whenever, wheresoever, wherever, withersoever ≠ however RGQV RR
345)	RRQV31	no matter how, no matter when, no matter where, no matter why ≠ no matter how RGQV
346)	RRQV32	no matter how, no matter when, no matter where, no matter why
347)	RRQV33	no matter how, no matter when, no matter where, no matter why
348)	RRQq	Adverbio interrogativo que comienza por wh, e.g when, whence, where, whereabouts, whither, why, how ≠ when CSn RRQr, whence RRQr, where CSr RRQr, whereabouts NN2, whither RRQr, why RRQr UH, how RGQq RRQr
349)	RRQr	Adverbio relativo que comienza por wh, e.g when, whence, where, whereat, whereby, wherein, whereoff, whereon, whereupon, wherewith, whither, why, how ≠ when CSn RRQq, whence RRQq, where CSr RRQq, whereat CS, whereupon CS, whither RRQq, why RRQq, how RGQq RRQq
350)	RRR	Adverbio comparativo de una sola palabra, excepto more, less, e.g. better, closer, deeper, earlier, farther, harder, higher, later, longer, louder, lower, nearer, nigher, oftener, quicker, rather, safer, slower, sooner, wider, worse ≠ JJR para todos excepto oftener, rather, sooner y ≠ nearer II, nigher II, rather RGr, worse NN1u
351)	RRT	Adverbio superlativo de una sola palabra, excepto most, least, e.g. best, brightest, closest, deepest, earliest, farthest, fastest, furthest, hardest, highest, latest, longest, loudest, lowest, nearest, nighest, soonest, widest, worst ≠ JJT para todos excepto para soonest y ≠ nearest II, nighest II, worst NN1u
352)	RRe	enough como cláusula adverbial ≠ DDf RGAf
353)	RRf	far como adverbio ≠ JJ
354)	RRg	long como adverbio ≠ JJ

355)	RRs	otherwise, yet
356)	RRx	only como adverbio ≠ JBy
357)	RRz	so introduciendo una cláusula principal o una cláusula de propósito o resultado, o como adverbio de modo o grado ≠ RGz PN1z
358)	RT	again, hereafter, overnight ≠ hereafter NN1c, overnight JB
359)	RTn	then
360)	RTo	now ≠ CS
361)	RTt	today, tomorrow, tonight, yesterday
362)	TO	to infinitivo ≠ IIt RL
363)	UH	Interjección, e.g. hello, please, well, yes
364)	UH21	Interjección de 2 palabras (1a. palabra)
365)	UH22	Interjección de 2 palabras (2a. palabra)
366)	VB0	be
367)	VBDR	were
368)	VBDZ	was
369)	VBG	being ≠ NN1n
370)	VBM	am
371)	VBN	been
372)	VBR	are
373)	VBZ	is
374)	VD0	do
375)	VDD	did
376)	VDG	doing
377)	VDN	done
378)	VDZ	does
379)	VH0	have
380)	VHD	had como tiempo pasado ≠ VHN
381)	VHG	having
382)	VHN	had como participio pasado ≠ VHD
383)	VHZ	has
384)	VMK	ought, used como verbo modal encadenado ≠ used JJ VVDt VVNt
385)	VMd	Verbo modal en pasado, e.g. could, might, should, would
386)	VMo	Verbo modal en presente, e.g. can, dare, may, must, need, shall, will ≠ dare VV0v, need VV0t
387)	VV0i	Verbo intransitivo en infinitivo
388)	VV0t	Verbo transitivo en infinitivo
389)	VV0v	Verbo de uso transitivo o intransitivo en infinitivo
390)	VVDi	Verbo intransitivo en tiempo pasado
391)	VVDt	Verbo transitivo en tiempo pasado
392)	VVDv	Verbo de uso transitivo o intransitivo en tiempo pasado
393)	VVGK	going de uso encadenado (e.g. be going to) ≠ JJ NN1n VVGi
394)	VVGi	Verbo intransitivo en participio presente
395)	VVGt	Verbo transitivo en participio presente
396)	VVGv	Verbo de uso transitivo o intransitivo en participio presente
397)	VVNK	bound de uso encadenado (e.g. be bound to) ≠ JJ NN1c VV0v VVDv VVNv
398)	VVNi	Verbo intransitivo en participio pasado
399)	VVNt	Verbo transitivo en participio pasado
400)	VVNv	Verbo de uso transitivo o intransitivo en participio pasado

401)	VVZi	Verbo intransitivo en tercera persona del singular
402)	VVZt	Verbo transitivo en tercera persona del singular
403)	VVZv	Verbo de uso transitivo o intransitivo en tercera persona del singular
404)	XX	not
405)	YB	División de texto sin cabecera (un párrafo o una unidad de orden mayor)
406)	YBL	Comienzo de cabecera
407)	YBR	Fin de cabecera
408)	YC	,
409)	YD	Guión de rotura
410)	YE	...
411)	YF	.
412)	YG	Marca o traza de la posición lógica de un elemento movido o eliminado
413)	YH	Guión de separación
414)	YIL	Comillas simples o dobles de apertura
415)	YIR	Comillas simples o dobles de cierre
416)	YN	:
417)	YO	:-
418)	YPL	(, [, {, ...
419)	YPR	),], }, ...
420)	YQ	?
421)	YS	;
422)	YTL	Comienzo de cursiva, negrita, ...
423)	YTR	Final de cursiva, negrita, ...
424)	YX	!
425)	ZZ1	Nombre de letra en singular

Apéndice B

Notación de los nodos no terminales del corpus SUSANNE

Las etiquetas de los nodos no terminales del esquema SUSANNE pueden contener hasta tres tipos de información: una *etiqueta de forma*, una *etiqueta de función*, y un *índice*, en este orden. Cuando la etiqueta contiene una etiqueta de forma y uno o varios de los otros dos elementos, la etiqueta de forma se separa de ellos mediante el caracter de dos puntos. Una etiqueta de función es siempre un único caracter alfabético, y un índice es una secuencia de tres dígitos, de tal manera que cualquier combinación válida de elementos dentro de una etiqueta de nodo siempre se puede descomponer sin ambigüedad en todos sus campos.

B.1 Clasificación de los constituyentes

Excepto las etiquetas de los nodos que dominan directamente los elementos *ghost* o elementos *fantasma* correspondientes a las trazas, todas las etiquetas de nodo incluyen etiquetas de forma, las cuales identifican las propiedades internas de la palabra o secuencia de palabras dominadas por el nodo. La forma de un árbol de análisis se define en términos de una jerarquía de clasificación de las etiquetas de forma:

1. Etiquetas de palabra: comienzan con dos letras mayúsculas; las restantes etiquetas de forma comienzan con una sola letra mayúscula y no contienen ninguna mayúscula más.
2. Etiquetas de frase: comienzan con alguna de las siguientes letras: `N`, `V`, `J`, `R`, `P`, `D`, `M`, `G`.
3. Etiquetas de cláusula: comienzan con alguna de las siguientes letras: `S`, `F`, `T`, `Z`, `L`, `A`, `W`.
4. Etiquetas de raíz: comienzan con alguna de las siguientes letras: `O`, `Q`, `I`.

A cada cláusula gramatical, si consta de una o más palabras, se le da un nodo etiquetado con una etiqueta de cláusula. A cada constituyente inmediato de una cláusula, si existen uno o más constituyentes y si cada constityente consta de una o más palabras, se le da un nodo etiquetado con una etiqueta de frase, a menos que el constituyente pertenezca a una categoría de palabra que no tiene su correspondiente categoría de frase (por ejemplo las marcas de puntuación o las conjunciones), o a una categoría raíz (por ejemplo una cita, con etiqueta de forma `Q`). Por tanto, a una cláusula que conste de un verbo se le asignará una etiqueta de cláusula (por ejemplo `Tg`, para una cláusula de participio presente), la cual dominará única y directamente una etiqueta de frase (por ejemplo `Vg`, para un grupo verbal que comience con un participio presente), la cual a su vez dominará única y directamente una etiqueta de palabra (por ejemplo `VVGi`, para un participio presente de un verbo intransitivo).

Se puede insertar un nodo intermedio de frase entre un nodo de frase más alto y una secuencia de palabras dominada por él, pero sólo si dos o más de esas palabras forman un constituyente coherente con esa frase más alta. Una cláusula que llena un hueco ocupado de manera estándar por una frase (por ejemplo una cláusula nominal, como sujeto u objeto) no tendrá un nodo de frase por debajo del nodo de la cláusula, a menos que la propia cláusula esté precedida y/o seguida por elementos modificadores que no sean parte de la cláusula.

B.2 Etiquetas de función e índices

Las etiquetas de función, que identifican papeles tales como sujeto superficial, objeto lógico, complemento temporal, etc., se asignan a todos los constituyentes de cláusula intermedios, excepto para el grupo verbal que va en cabeza y para otros constituyentes determinados en los que la función etiquetada es inapropiada.

Los índices se asignan a pares de nodos para mostrar identidades referenciales entre ítems que presentan determinadas relaciones gramaticales entre ellos. Por ejemplo, a una frase que aparece por encima de una cláusula más baja para actuar como objeto de una cláusula más alta, como en `John expected Mary to admit it`, se le asigna un índice idéntico al del elemento *fantasma* que marca la posición lógica del ítem de la cláusula más baja. El ejemplo (artificial) mencionado se representaría como sigue:

```
[Nns:s John] expected [Nns:0999 Mary] [Ti:o [s999 GHOST] to admit [Ni:o it]]
```

donde el índice `999` muestra que el elemento *fantasma* que actúa como sujeto lógico de la cláusula del `admit` (simbolizado por `s`) es correferencial con `Mary`, que actúa como objeto superficial de la cláusula del `expected` (`0`), siendo el objeto lógico de la cláusula del `expected` (`o`) la cláusula subordinada de infinitivo (`Ti`).

En algunos casos, el movimiento de reglas desplaza un constituyente hacia el interior de un sintagma que no tiene ningún papel gramatical (por ejemplo, un adverbio, que es lógicamente un constituyente de cláusula, puede interrumpir el grupo verbal – la secuencia de verbos auxiliares y verbo principal – de la cláusula): en tales casos la etiqueta de función es `G` (*guest*, invitado). A los constituyentes que no pertenecen de manera lógica al nodo que los domina inmediatamente en la estructura superficial se les asigna siempre etiquetas de función `G` e índices que los ligan a sus posiciones lógicas. Con esa excepción (y con otra que no discutiremos aquí relativa a la coordinación), el etiquetado de función se usa sólo para los constituyentes inmediatos de las cláusulas.

B.3 Las etiquetas de forma

Las etiquetas de forma en el corpus SUSANNE son como sigue:

Etiquetas de raíz	
`0`	párrafo
`0h`	cabecera
`0t`	título (por ejemplo de un libro)
`Q`	cita
`I`	interpolación
`Iq`	cláusula interrogativa
`Iu`	referencia técnica

Etiquetas de cláusula

`S`	cláusula principal
`Ss`	cláusula de cita embebida
`Fa`	cláusula adverbial
`Fn`	cláusula nominal
`Fr`	cláusula relativa
`Ff`	relativo fundido
`Fc`	cláusula comparativa
`Tg`	cláusula de participio presente
`Tn`	cláusula de participio pasado
`Ti`	cláusula de infinitivo
`Tf`	cláusula `for-to`
`Tb`	cláusula descubierta no finita
`Tq`	cláusula relativa de infinitivo
`W`	cláusula `with`
`A`	cláusula especial `as`
`Z`	relativo reducido (*whiz-deleted*)
`L`	cláusula miscelánea sin verbo

Etiquetas de frase

`V`	grupo verbal
`N`	frase nominal
`J`	frase adjetiva
`R`	frase adverbial
`P`	frase preposicional
`D`	frase determinante
`M`	frase numeral
`G`	frase genitiva

Las diferentes categorías de frase añaden símbolos de subcategoría en minúsculas, que se pueden combinar de cualquier manera para conseguir el significado deseado (por ejemplo, el grupo verbal `must have been noticed` tendría la etiqueta de forma `Vcfp`). Las subcategorías de frase son:

`Vo`	operador de sección de un grupo verbal, cuando va separado del resto de `V` (por ejemplo, mediante la inversión de sujeto y auxiliar)
`Vr`	resto de `V` a partir del cual `Vo` ha sido separado
`Vm`	`V` comenzando con `am`
`Va`	`V` comenzando con `are`
`Vs`	`V` comenzando con `was`
`Vz`	`V` comenzando con otro verbo en tercera persona del singular
`Vw`	`V` comenzando con `were`
`Vj`	`V` comenzando con `be`
`Vd`	`V` comenzando con un tiempo verbal en pasado
`Vi`	`V` infinitivo
`Vg`	`V` comenzando con participio presente
`Vn`	`V` comenzando con participio pasado
`Vc`	`V` comenzando con verbo modal
`Vk`	`V` que contiene un `DO` de énfasis
`Ve`	`V` negativo

Vf	V perfectivo
Vu	V progresivo
Vp	V pasivo
Vb	V terminando con BE
Vx	V con pérdida del verbo principal
Vt	V encadenado
Nq	N de la forma wh-
Nv	N de la forma wh...ever
Ne	N siendo o comenzando por I/me
Ny	N siendo o comenzando por you
Ni	N siendo o comenzando por it
Nj	N comenzando por adjetivo
Nn	nombre propio
Nu	unidad de medida
Na	marca de sujeto
No	marca de no sujeto
Ns	marca de singular
Np	marca de plural
Jq	J de la forma wh-
Jv	J de la forma wh...ever
Jx	J de medida absoluta
Jr	J de medida comparativa
Jh	heavy
Rq	R de la forma wh-
Rv	R de la forma wh...ever
Rx	R de medida absoluta
Rr	R de medida comparativa
Rs	adverbio que conduce a un asíndeton
Rw	adverbio casi nominal
Po	frase of
Pb	frase by
Pq	P de la forma wh-
Pv	P de la forma wh...ever
Dq	D de la forma wh-
Dv	D de la forma wh...ever
Ds	marca de singular
Dp	marca de plural
Ms	M comenzando con one
Gq	G de la forma wh-
Gv	G de la forma wh...ever

B.4 Sufijos de etiqueta de forma no alfanuméricos

Las etiquetas de forma pueden contener también símbolos no alfanuméricos como los siguientes:

?	cláusula interrogativa
*	cláusula imperativa
%	cláusula subjuntiva
!	cláusula exclamatoria u otro ítem
"	elemento vocativo

Otros símbolos no alfanuméricos representan estructuras de coordinación. Bajo el esquema SUSANNE, el segundo y los subsiguientes elementos conjuntivos de una coordinación se analizan como subordinados del primer elemento conjuntivo. Por tanto, a una coordinación de la forma:

```
chi, psi, and omega
```

cualquiera que sea la clasificación gramatical de la secuencia de palabras `chi`, `psi`, etc., se le asignaría una estructura de la forma:

```
[chi, [psi], [and omega]]
```

La etiqueta de forma de toda la coordinación se determina a partir de las propiedades del primer elemento conjuntivo, excepto para las subcategorías singular/plural, en el caso de aquellas categorías de frase donde éstas son aplicables. Los posteriores elementos conjuntivos, que a menudo aparecerán reducidos por algún tipo de transformación, tienen nodos propios cuyas etiquetas de forma los marcan como elementos conjuntivos subordinados. Los símbolos relacionados con las estructuras de coordinación son los siguientes:

+	elemento subordinado introducido mediante conjunción
-	elemento subordinado no introducido mediante conjunción
@	elemento aposicional
&	estructura coordinada actuando como primer elemento dentro de una coordinación más alta (marcada sólo en ciertos casos)

La coordinación se reconoce cuando ocurre entre palabras, pero también entre sintagmas de clasificación más alta. Por tanto, los nodos no terminales pueden tener etiquetas de forma que constan de etiquetas de palabra seguidas de símbolos de coordinación. Así pues, siendo `WT` una etiqueta de palabra arbitraria, son combinaciones válidas:

`WT&`	coordinación de palabras
`WT+`	elemento dentro de la coordinación de nivel de palabra introducido mediante una conjunción
`WT-`	elemento dentro de la coordinación de nivel de palabra no introducido mediante una conjunción

Una coordinación de nivel de palabra siempre usa el símbolo `&` en su etiqueta de forma; las coordinaciones de frase o de cláusula lo usan sólo en circunstancias muy restringidas.

Puede ocurrir también que ciertas secuencias de palabras ortográficas, en determinados usos, funcionen gramaticalmente como palabras individuales (modismos y frases hechas). Por ejemplo, la secuencia de palabras `none the less` sería tratada normalmente como un modismo

gramatical equivalente a un adverbio, para el cual la etiqueta de palabra es RR. En tales casos, el nodo no terminal que domina la secuencia tiene una etiqueta de forma que consta de un signo igual añadido como sufijo a la correspondiente etiqueta de palabra, y las palabras individuales que forman el modismo gramatical incorporan en sus etiquetas un sufijo numérico que refleja su pertenencia al modismo. La secuencia **none the less** tendría la etiqueta de forma RR=, y las palabras **none**, **the** y **less** llevarían en este contexto las etiquetas de palabra RR31, RR32 y RR33.

Es importante señalar que las etiquetas de forma WT&, WT+, WT- y WT= figuran en la clasificación como etiquetas de palabra para el propósito de determinar la estructura de árbol, tal y como se discutió anteriormente.

B.5 Las etiquetas de función

Las etiquetas de función se dividen en etiquetas de complemento y etiquetas de adjunto: básicamente, una etiqueta de complemento dada puede aparecer como máximo una vez en una cláusula, mientras que una cláusula puede contener múltiples adjuntos del mismo tipo.

Etiquetas de función de complemento

0	párrafo
s	sujeto lógico
o	objeto directo lógico
i	objecto indirecto
u	objeto preposicional
e	complemento predicado de sujeto
j	complemento predicado de objeto
a	agente de pasiva
S	sujeto superficial (y no lógico)
O	objeto directo superficial (y no lógico)
G	invitado (*guest*) sin papel gramatical dentro de este sintagma

Etiquetas de función de adjunto

p	lugar
q	dirección
t	tiempo
h	manera o grado
m	modalidad
c	contingencia
r	respecto
w	comitativo
k	benefactivo
b	absoluto

Otras etiquetas de función

n	participio de frase verbal
x	cláusula relativa que tiene otra cláusula mayor como antecedente
z	complemento de encadenación

Apéndice C

Transiciones de palabras al integrar diccionarios externos

Tal y como discutimos en la sección 7.2.2, el proceso de integración de diccionarios externos da lugar a una serie de transiciones de palabras entre los distintos contadores de acierto y fallo. Una forma de representar estos cambios es mediante los diagramas de transición de palabras que describimos a continuación.

C.1 Diagramas de transición de palabras

Un diagrama de transición de palabras es un grafo dirigido donde cada uno de los nodos contiene, en este orden, lo siguiente:

1. El nombre de uno de los contadores.
2. El número de palabras en este contador con el lexicón de entrenamiento.
3. El nuevo número de palabras en el contador al integrar el lexicón externo.

Las aristas del grafo representan las transiciones de las palabras. Las etiquetas de las aristas indican cuántas palabras pasan del contador del nodo origen al contador del nodo destino.

Por ejemplo, la figura C.1 muestra las transiciones de palabras en el test11 sobre el corpus ITU con el sistema BRILL al integrar el lexicón del sistema GALENA, y la figura C.2 muestra las transiciones del mismo experimento al integrar todo el lexicón del corpus ITU. En ambas figuras hemos marcado lo siguiente:

- Con flechas más gruesas, las transiciones que van desde los nodos correspondientes a los fallos a los nodos correspondientes a los aciertos.
- Con flechas discontinuas, lo contrario, es decir, las transiciones que van desde los nodos correspondientes a los aciertos a los nodos correspondientes a los fallos.
- Con flechas normales, las transiciones restantes, es decir, las que van de nodos de acierto a nodos de acierto (palabras que se siguen etiquetando correctamente), y las que van de nodos de fallo a nodos de fallo (errores que persisten).

Además, se hace mención explícita de la diferencia entre las etiquetas de las flechas más gruesas y las etiquetas de las flechas discontinuas, es decir, de la ganancia absoluta de nuevas palabras acertadas que se produce en el proceso de etiquetación tras la integración del diccionario externo, y de cómo esta ganancia afecta a los índices S1 y S2, para los cuales se muestra el valor previo a la integración, el valor posterior y el incremento.

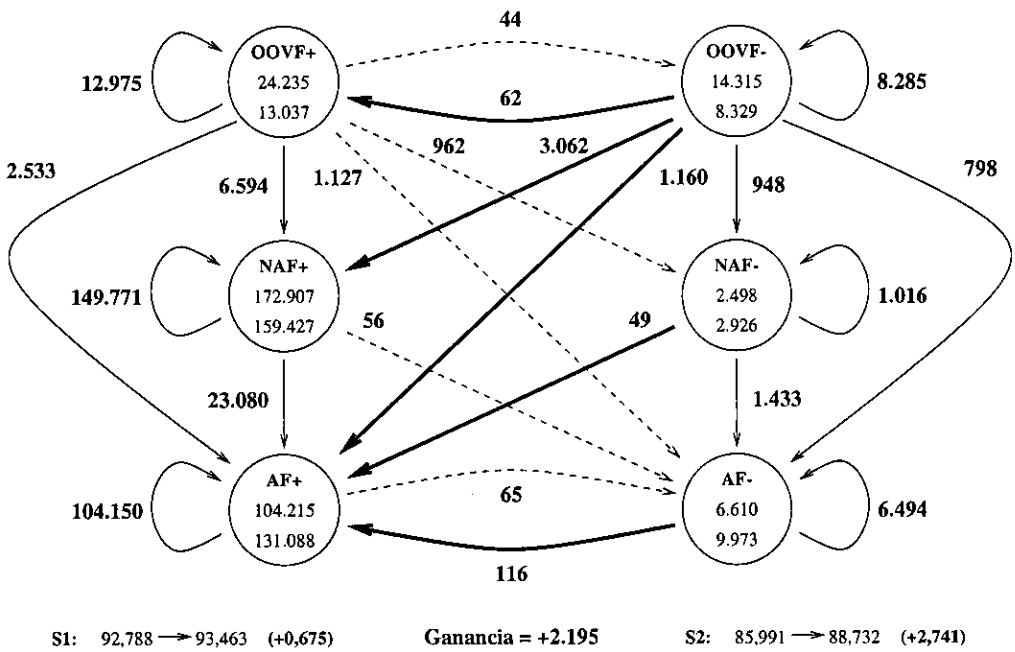


Figura C.1: Corpus: ITU - Sistema: BRILL - Test: test11 - Transiciones de palabras desde el lexicón de Entrenamiento al lexicón Entrenamiento+GALENA

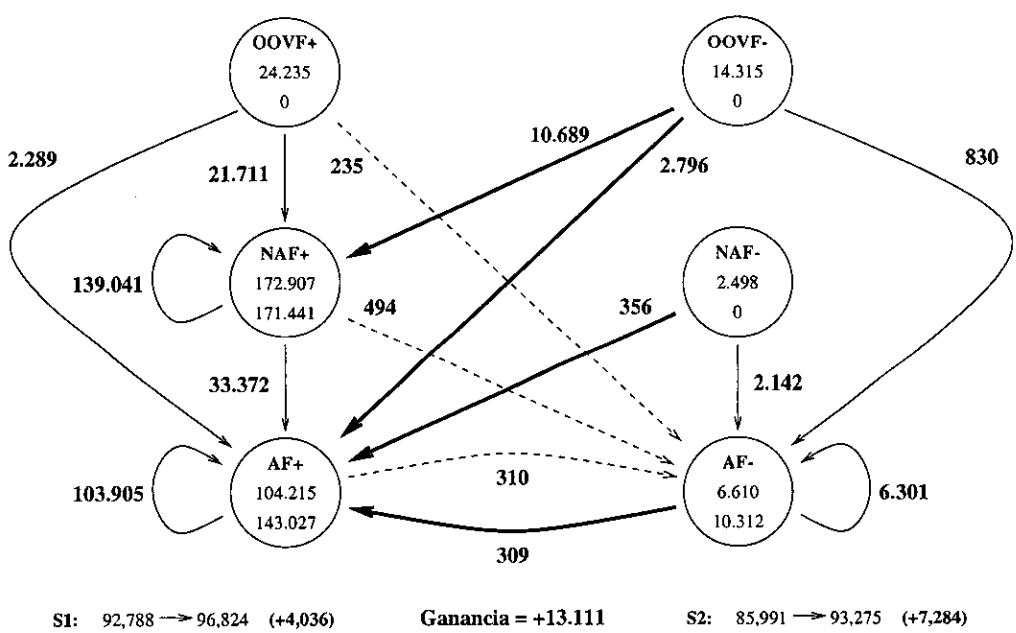


Figura C.2: Corpus: ITU - Sistema: BRILL - Test: test11 - Transiciones de palabras desde el lexicón de Entrenamiento al lexicón Entrenamiento+ITU

C.2 Transiciones de palabras en los sistemas BRILL y GALENA

Nuestro interés se centra en las transiciones de palabras de los experimentos realizados sobre el corpus ITU con los etiquetadores que permiten integrar diccionarios externos: BRILL y GALENA. Pero un total de 16 experimentos, para dos etiquetadores distintos, y al integrar dos lexicones distintos, daría lugar a 64 diagramas de transición. Por lo tanto, hemos preferido plasmar sólo la información relativa a la ganancia absoluta de palabras y al incremento de los índices S1 y S2. Dicha información se muestra en las tablas C.1 y C.2.

Lexicón: Entr. → Entr.+GALENA			
Test	Ganancia	Δ S1	Δ S2
test11	+2.195	+0,675	+2,741
test12	+623	+0,192	+1,466
test13	+426	+0,131	+1,054
test14	+524	+0,161	+1,001
test15	+240	+0,074	+0,602
test21	+2.347	+0,730	+3,152
test22	+865	+0,270	+1,567
test23	+462	+0,144	+1,078
test24	+359	+0,112	+0,874
test25	+123	+0,038	+0,634
test31	+1.892	+0,583	+2,660
test32	+1.213	+0,373	+1,835
test33	+586	+0,181	+1,163
test34	+314	+0,097	+0,745
test35	+84	+0,026	+0,523
testSP	+65	+0,040	+0,407
Lexicón: Entr. → Entr.+ITU			
Test	Ganancia	Δ S1	Δ S2
test11	+13.111	+4,036	+7,284
test12	+7.519	+2,315	+4,259
test13	+4.498	+1,385	+2,642
test14	+3.278	+1,009	+2,007
test15	+2.290	+0,705	+0,985
test21	+14.032	+4,368	+7,901
test22	+7.811	+2,432	+4,312
test23	+3.547	+1,104	+1,950
test24	+3.224	+1,004	+1,919
test25	+2.526	+0,786	+1,571
test31	+13.197	+4,067	+7,530
test32	+8.096	+2,495	+4,437
test33	+4.705	+1,450	+2,565
test34	+3.013	+0,929	+1,575
test35	+2.463	+0,759	+1,336
testSP	+779	+0,478	+0,527

Tabla C.1: Corpus: ITU - Sistema: BRILL - Transiciones de palabras al integrar un lexicón externo

Lexicón: Entr. → Entr.+GALENA			
Test	Ganancia	△ S1	△ S2
test11	+3.823	+1,177	+3,516
test12	+1.923	+0,592	+2,050
test13	+1.214	+0,374	+1,458
test14	+708	+0,218	+1,041
test15	+935	+0,177	+0,780
test21	+3.372	+1,021	+3,441
test22	+1.757	+0,547	+1,926
test23	+1.226	+0,381	+1,468
test24	+767	+0,258	+1,069
test25	+457	+0,143	+0,799
test31	+3.713	+1,144	+3,605
test32	+1.830	+0,564	+2,014
test33	+1.118	+0,344	+1,404
test34	+813	+0,251	+1,001
test35	+517	+0,160	+0,749
testSP	+126	+0,077	+0,443

Lexicón: Entr. → Entr.+ITU			
Test	Ganancia	△ S1	△ S2
test11	+13.066	+4,023	+7,181
test12	+7.872	+2,424	+4,345
test13	+5.817	+1,791	+3,429
test14	+4.187	+1,289	+2,536
test15	+3.625	+1,006	+1,617
test21	+12.748	+3,939	+6,940
test22	+7.641	+2,379	+4,087
test23	+5.515	+1,716	+3,181
test24	+3.942	+1,227	+2,343
test25	+2.920	+0,909	+1,797
test31	+13.123	+4,044	+7,423
test32	+8.176	+2,520	+4,385
test33	+5.529	+1,704	+3,047
test34	+4.076	+1,256	+2,232
test35	+3.128	+0,964	+1,743
testSP	+1.127	+0,691	+0,984

Tabla C.2: Corpus: ITU - Sistema: GALENA - Transiciones de palabras al integrar un lexïcón externo

Estas tablas no hacen más que confirmar que el método de integración de diccionarios externos que implementa el sistema GALENA ofrece unos resultados ligeramente superiores a los del sistema BRILL. Esto puede observarse también comparando la figura 7.3 con la 7.5 para el índice S1, y la 7.4 con la 7.6 para el índice S2. En ambos casos se observa que la separación entre curvas es mayor en las figuras correspondientes al sistema GALENA.

Apéndice D

Análisis sintáctico estocástico y paralelismo

La complejidad temporal del algoritmo CYK extendido, medida en términos de n, la longitud de la frase de entrada, es claramente de orden $\mathcal{O}(n^3)$. Sin embargo, la constante multiplicativa asociada a n^3, en este caso, juega un importante papel. Se podría calcular una complejidad más detallada para el peor caso de análisis, en función del cardinal de los conjuntos de no terminales y de reglas, y veríamos que es menor que la de otros algoritmos de análisis sintáctico similares. No vamos a desarrollar este cálculo aquí[1], pero la explicación intuitiva es que la ganancia que se produce, respecto a otros algoritmos del tipo Earley o del tipo CYK, es debida al hecho de que el algoritmo CYK extendido realiza una mejor factorización de las reglas punteadas[2], y procesa menos elementos[3].

Aún con todo esto, dicha complejidad se puede rebajar, debido a la naturaleza intrínsecamente paralela de los algoritmos tipo CYK: existen muchas celdas de la tabla de análisis que pueden ser calculadas simultáneamente. Para la paralelización del algoritmo CYK extendido [Barcala *et al.* 2000], se han planteado dos alternativas diferentes, barajando las dos tendencias principales que existen actualmente en el procesamiento paralelo:

1. Memoria distribuida, en este caso, mediante paso de mensajes entre distintos computadores.

2. Memoria compartida, es decir, ejecución de varios procesos que acceden concurrentemente a la misma memoria física.

A continuación se exponen con más detalle ambas aproximaciones.

D.1 Memoria distribuida

En esta alternativa se plantea la posibilidad de que cada máquina calcule un fragmento de la tabla de análisis, de manera que ésta quede repartida por la memoria de todos los computadores utilizados. Una posibilidad es que cada máquina calcule una o varias columnas de la tabla, de manera que el paralelismo se afronta a nivel de columna. Este reparto presenta la ventaja de que cada procesador siempre tiene una celda de las que le hacen falta para calcular la siguiente. Un inconveniente consiste en que una vez que un procesador termina con sus columnas, no ayuda a

[1] Puede verse en detalle en [Chappelier y Rajman 1998].
[2] Las reglas que aparecen en los ítems de las listas de tipo 2.
[3] Los ítems aparecen sólo una vez y no se hacen predicciones.

realizar el trabajo de los demás. Por otra parte, si existen más procesadores que palabras en la frase, los procesadores sobrantes no realizan ningún trabajo.

Bajo esta perspectiva, sólo queda por determinar cómo se hace el reparto de las columnas entre las máquinas y cómo es el intercambio de mensajes entre los procesadores. Con respecto al reparto, lo más razonable es efectuar una distribución cíclica invertida, de tal manera que si se tienen, por ejemplo, 10 columnas y 3 procesadores, se asignan las columnas 1, 6, y 7 al procesador 1, las columnas 2, 5, y 8 al procesador 2, y las columnas 3, 4, 9 y 10 al procesador 3, tal y como se muestra en la parte izquierda de la figura D.1. De esta forma, se obtiene una distribución balanceada de la carga de procesamiento entre las máquinas. La parte derecha de la figura muestra las subtablas que mantendría en memoria cada procesador.

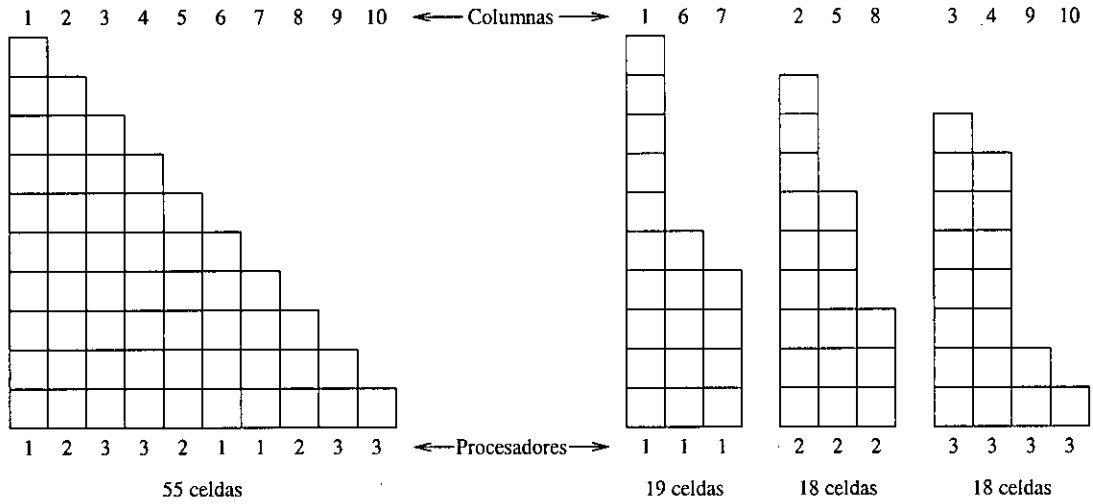


Figura D.1: Reparto de columnas de la tabla de análisis en una paralelización del algoritmo CYK extendido mediante memoria distribuida

En cuanto a las comunicaciones, si los procesadores envían cada celda, una vez calculada, a los demás a los que les hará falta para calcular celdas posteriores, se elimina la necesidad de establecer una comunicación bidireccional de petición y envío de celdas entre los procesadores. No habrá peticiones, sino sólo envíos. Además, dichos envíos se pueden realizar una sola vez a cada procesador, de manera que el destino, cuando recibe una celda, debe determinar si le hará falta otra vez para calcular otra celda de otra columna, en cuyo caso debe envíarse a sí mismo la celda nuevamente.

Sin embargo, esta alternativa daría más prioridad a la rapidez de cálculo que a la fiabilidad. Efectivamente, con la supresión de las comunicaciones bidireccionales se produce un aumento considerable de la velocidad de procesamiento. Cada procesador busca las celdas que necesita en su cola de entrada. Si no encuentra una determinada celda, será porque el procesador responsable de calcularla aún no ha terminado su tarea y aún no se la ha enviado, con lo cual se producirá una espera momentánea hasta que dicho cálculo termine. Pero existe el problema de que un procesador puede agotar su capacidad de almacenamiento de celdas. Dicho problema se acentúa cuando se utilizan pocos procesadores, pudiendo producirse un bloqueo general del algoritmo.

Incluso con esta optimización del intercambio de mensajes y aunque no se produzcan bloqueos, el gran problema de la aproximación con memoria distribuida es la *granularidad* del proceso que se quiere paralelizar. En nuestro caso, si la cantidad de información que reside en las celdas es muy grande, incluso aunque no existan demasiadas esperas, es muy probable que los procesadores dediquen más tiempo al envío y recepción de los datos, que al cálculo de los mismos. Puede ocurrir que la ganancia producida por la paralelización no sea suficientemente significativa con respecto a los recursos computacionales utilizados, o incluso que el mismo proceso se ejecute

más rápido utilizando una sola máquina secuencial. Todas estas reflexiones nos llevan al estudio de la alternativa con memoria compartida.

D.2 Memoria compartida

En una paralelización mediante memoria compartida, todos los procesadores comparten la misma tabla de análisis, y deben ir rellenando sus celdas simultáneamente. Al contrario que en el caso anterior, las celdas se van calculando de izquierda a derecha y de abajo a arriba, independientemente de los procesadores y a medida que éstos van quedando libres, hasta rellenar toda la tabla. De esta manera no queda ningún procesador libre antes de tiempo y hay un mayor aprovechamiento de la capacidad de procesamiento de cada uno de ellos. Esta alternativa no resulta viable con memoria distribuida, ya que la sincronización de todos los procesadores supone un coste de comunicación muy elevado.

Una característica importante inherente al paradigma de paralelismo con memoria compartida es que no hay transferencia de datos entre los procesadores. Esto supone una gran ventaja en casos como el que nos ocupa, donde la cantidad de información que se desea compartir es elevada. Sin embargo, hay que tener en cuenta que los procesadores deben acceder concurrentemente a las mismas zonas de memoria, por lo que hay que determinar las secciones críticas y realizar una protección adecuada sobre ellas. En nuestro caso, es suficiente con definir una variable asociada a cada celda de la tabla, que indique si está ya calculada o no, y un semáforo asociado a dicha variable.

Hay que destacar que el orden de los lazos que determinan qué celda se procesa es importante para el adecuado comportamiento del analizador, y debe ajustarse perfectamente a los requerimientos de esta alternativa. Tal y como hemos visto, el lazo externo debe ser el de las filas, y el interno el de las columnas. De esta manera, no se producirán situaciones de interbloqueo, es decir, situaciones donde un proceso necesite una casilla que está calculando otro, y viceversa, que el otro necesite la casilla que está calculando el primero. Cuando un procesador comienza a calcular una celda, las celdas que va a necesitar, o ya están calculadas, o se están calculando, y ningún procesador que las esté calculando va a necesitar la celda del primero, ya que ésta está por encima o como mucho al mismo nivel. Por tanto, está garantizado que esta alternativa de paralelización no producirá abrazos mortales.

De todo esto se puede deducir que el algoritmo de análisis sintáctico CYK extendido presenta un esquema de paralelización natural. Aunque como veremos más adelante, no es apropiado implementarlo sobre un multicomputador de memoria distribuida, ya que la penalización en las comunicaciones afecta mucho a su eficiencia, la implementación en un multiprocesador considerando varios hilos de ejecución[4] aporta unos resultados sorprendentes.

D.3 Análisis de resultados

Los experimentos de paralelismo se han realizado sobre un computador con sistema operativo Linux, formado por los siguientes componentes:

- 1 nodo principal con 2 procesadores Pentium II a 350 MHz, y con 384 MB de memoria RAM (multiprocesador de memoria compartida).
- 23 nodos adicionales AMD K6 a 300 MHz, con 96 MB de memoria RAM, agrupados mediante un *switch fast ethernet* (multicomputador de memoria distribuida).

[4]También denominados *threads*.

Las versiones secuencial y multihilo se ejecutan en el nodo central. La versión de memoria distribuida se ejecuta repartidamente por todos los nodos, incluido el nodo principal, e implementa la comunicación entre los procesadores mediante la librería estándar MPI[5].

Los datos sobre los que se ha evaluado el algoritmo han sido tomados del banco de árboles SUSANNE [Sampson 1994a, Sampson 1994b], el cual ha sido dividido en dos partes: la primera formada por las frases sin trazas (4.292 frases, 77.275 palabras), y la segunda formada por las frases con trazas (2.188 frases, 60.759 palabras). La primera parte ha sido utilizada para extraer una gramática (compuesta por 17.669 reglas y 1.525 símbolos no terminales) [Graña y Rajman 1999], y la segunda para medir los tiempos de análisis de las diferentes versiones del algoritmo.

Los resultados de la tabla D.1 presentan tiempos relacionados con: secuencial, 1 hilo, 2 hilos, 2 MPI, 4 MPI, etc. Cada uno de estos descriptores se corresponde, respectivamente, con la versión secuencial, la versión multihilo con un solo hilo de ejecución[6], la versión multihilo con 2 hilos de ejecución, y las versiones de memoria distribuida con 2 procesadores, 4 procesadores, etc. Los tiempos se muestran en formato `mm:ss.mmm` (minutos, segundos y milésimas de segundo). Los tiempos de usuario para 2 hilos representan la suma de los tiempos de ejecución de los dos procesadores. Por tanto, en este caso, aunque sea menos fiable, es más representativo el tiempo real.

Análisis de 2.188 frases							
Tiempos	Secuencial	1 hilo	2 hilos	2 MPI	4 MPI	8 MPI	16 MPI
Real	03:45.259	03:52.648	02:24.311	10:47.108	09:28.020	07:57.424	07:40.391
Usuario	03:44.810	03:50.580	03:53.040	05:08.900	03:55.730	02:36.370	02:07.190
Sistema	00:00.036	00:00.530	00:01.370	05:16.200	05:26.370	05:17.350	05:28.990

Tabla D.1: Tiempos de ejecución del algoritmo CYK extendido sobre el corpus SUSANNE

A través de estos datos se observa que existe un coste adicional por el manejo de semáforos, pero que aún así la versión multihilo con 2 hilos de ejecución es considerablemente más rápida que la versión secuencial. Además, la implementación evaluada considera el paralelismo a nivel de frase, por lo que las estructuras de los hilos de ejecución se crean y destruyen para cada una de las frases, con el coste que ello supone, pudiendo optimizarse aún más si los hilos se mantienen entre frase y frase.

Por otra parte, la versión con memoria distribuida parece la peor en cualquier caso, y esto se debe a tres razones principales, alguna de las cuales ya ha sido comentada anteriormente:

1. Para frases pequeñas, el coste de comunicación de los datos entre los diferentes procesadores es elevado.

2. Cuando la frase es grande, debería existir una repartición equilibrada de datos entre las diferentes columnas, quedando equilibrado el cálculo entre los procesadores. Pero esto no es lo usual debido a la asimetría de las tablas de análisis, que típicamente contienen gran cantidad de celdas vacías.

3. Cuando la frase es compleja, las celdas suelen contener demasiada información de manera que resulta muy cara la comunicación entre los diferentes procesadores. Puede observarse que al aumentar el número de procesadores, disminuye el tiempo de ejecución, pero

[5] *Message Passing Interface.*

[6] De esta forma se deduce el coste de inicialización y manejo de los semáforos.

aumenta el tiempo del sistema, que en este caso incluye, fundamentalmente, el tiempo de comunicación entre los diferentes procesadores.

Existe una ganancia obvia de la implementación con 2 hilos de ejecución: un tiempo de análisis de 2:24.311 sobre 2.188 frases implica una media aproximada de 0,065 segundos por frase. Pero este dato tampoco es representativo en absoluto. Para demostrarlo, basta echar un vistazo a la tabla D.2, en la cual se estudian los tiempos de ejecución de 5 frases de manera independiente. Hay frases que se analizan en tiempos muy superiores a la media, que oscilan entre 318 milésimas de segundo y casi dos minutos. Algo a destacar también es el comportamiento sobre la frase 5, donde la aproximación multihilo no produce una mejora significativa del rendimiento. Esto se debe a que muy probablemente en la tabla de análisis de esta frase haya muchas celdas vacías, y de hecho es así, ya que el tiempo de análisis es realmente bajo para tratarse de una frase de 214 palabras (23.005 celdas). Pero la existencia de pocas celdas con ítems hace que las esperas entre los procesadores sean muy frecuentes. Es decir, el paralelismo se explota mucho mejor cuanto más compleja sea la frase a analizar.

	Frase 1	Frase 2	Frase 3	Frase 4	Frase 5
Palabras	30	35	55	150	214
Tiempos en Secuencial					
Real	00:00.351	00:04.646	02:40.821	00:00.750	00:22.268
Usuario	00:00.340	00:04.630	02:40.600	00:00.740	00:22.160
Sistema	00:00.010	00:00.020	00:00.220	00:00.010	00:00.100
Tiempos en 2 hilos					
Real	00:00.318	00:02.735	01:57.759	00:00.623	00:21.004
Usuario	00:00.460	00:04.810	02:39.800	00:00.950	00:23.480
Sistema	00:00.001	00:00.020	00:00.140	00:00.030	00:00.200

Tabla D.2: Tiempos de ejecución independientes del algoritmo CYK extendido sobre 5 frases

Lo que sí resulta también evidente es que la complejidad de una frase no es proporcional al número de palabras de la misma, lo que queda claro comparando los tiempos de análisis de las frases 3 y 4, y que, en todo caso, la mejora de la versión de 2 hilos de ejecución respecto a la secuencial sigue presente.

Por último, es importante destacar que los tiempos mostrados no han sido medidos sobre una implementación estándar del algoritmo CYK extendido, sino sobre una versión que realiza tareas adicionales específicamente orientadas al problema de la etiquetación, tal y como se explica en la sección 8.5.

En resumen, la paralelización del algoritmo CYK extendido mediante memoria compartida lo convierte en una alternativa que podría competir con otras técnicas de análisis sintáctico mucho más eficientes. Si bien esas otras técnicas también podrían ser paralelizables, esta paralelización, en caso de que sea posible, no será en absoluto inherente al propio algoritmo, y dará lugar a una complejidad de diseño e implementación mucho mayor.

Apéndice E

Experimentos de análisis sintáctico robusto sobre el corpus SUSANNE

Las siguientes secciones muestran los datos obtenidos en los experimentos que se mencionan en el capítulo 9, y por tanto constituyen las cifras concretas a partir de las cuales se han construido las gráficas que aparecen en dicho capítulo.

E.1 Ambigüedades de la gramática SUSANNE sobre las frases sin trazas

Las siguientes filas de números constituyen los datos sobre las ambigüedades de la gramática SUSANNE al analizar todas las frases de la parte sin trazas. Estas cifras se resumen gráficamente en la figura 9.1. Cada celda contiene cuatro cifras: número de ambigüedades, número de frases que presentan ese número de ambigüedades, porcentaje de dichas frases sobre el total, y porcentaje de frases acumulado.

1	846	19,711	19,711	2	497	11,580	31,291	3	198	4,613	35,904
4	225	5,242	41,146	5	133	3,099	44,245	6	149	3,472	47,717
7	42	0,979	48,695	8	136	3,169	51,864	9	55	1,281	53,145
10	87	2,027	55,172	11	13	0,303	55,475	12	76	1,771	57,246
13	16	0,373	57,619	14	25	0,582	58,201	15	31	0,722	58,924
16	62	1,445	60,368	17	7	0,163	60,531	18	42	0,979	61,510
19	5	0,116	61,626	20	68	1,584	63,211	21	15	0,349	63,560
22	19	0,443	64,003	23	4	0,093	64,096	24	56	1,305	65,401
25	19	0,443	65,843	26	11	0,256	66,100	27	13	0,303	66,403
28	14	0,326	66,729	29	5	0,116	66,845	30	35	0,815	67,661
31	3	0,070	67,731	32	39	0,909	68,639	33	6	0,140	68,779
34	6	0,140	68,919	35	2	0,047	68,966	36	28	0,652	69,618
37	2	0,047	69,664	38	7	0,163	69,828	39	3	0,070	69,897
40	32	0,746	70,643	41	3	0,070	70,713	42	21	0,489	71,202
43	2	0,047	71,249	44	10	0,233	71,482	45	11	0,256	71,738
46	6	0,140	71,878	48	30	0,699	72,577	49	2	0,047	72,623
50	14	0,326	72,950	51	2	0,047	72,996	52	6	0,140	73,136
53	1	0,023	73,159	54	15	0,349	73,509	55	2	0,047	73,555
56	11	0,256	73,812	57	4	0,093	73,905	58	3	0,070	73,975
59	3	0,070	74,045	60	23	0,536	74,581	61	1	0,023	74,604
62	3	0,070	74,674	63	2	0,047	74,720	64	17	0,396	75,116
65	2	0,047	75,163	66	6	0,140	75,303	67	1	0,023	75,326
68	4	0,093	75,419	69	2	0,047	75,466	70	4	0,093	75,559
72	15	0,349	75,909	74	1	0,023	75,932	75	3	0,070	76,002
76	4	0,093	76,095	77	2	0,047	76,142	78	4	0,093	76,235
79	3	0,070	76,305	80	20	0,466	76,771	82	2	0,047	76,817
83	1	0,023	76,841	84	11	0,256	77,097	85	1	0,023	77,120
86	1	0,023	77,144	87	1	0,023	77,167	88	7	0,163	77,330
89	2	0,047	77,377	90	17	0,396	77,773	91	1	0,023	77,796
92	1	0,023	77,819	93	1	0,023	77,842	94	3	0,070	77,912
96	16	0,373	78,285	99	3	0,070	78,355	100	12	0,280	78,635

101	1	0,023	78,658	102	8	0,186	78,844	104	5	0,116	78,961
105	6	0,140	79,101	106	1	0,023	79,124	107	1	0,023	79,147
108	3	0,070	79,217	110	4	0,093	79,310	111	1	0,023	79,334
112	6	0,140	79,473	114	1	0,023	79,497	116	2	0,047	79,543
118	1	0,023	79,567	120	21	0,489	80,056	121	1	0,023	80,079
123	1	0,023	80,103	124	1	0,023	80,126	126	3	0,070	80,196
128	4	0,093	80,289	129	1	0,023	80,312	130	3	0,070	80,382
132	1	0,023	80,405	134	1	0,023	80,429	135	7	0,163	80,592
136	2	0,047	80,638	140	2	0,047	80,685	141	2	0,047	80,732
142	2	0,047	80,778	144	12	0,280	81,058	146	1	0,023	81,081
147	1	0,023	81,104	148	2	0,047	81,151	149	1	0,023	81,174
150	11	0,256	81,431	152	2	0,047	81,477	154	1	0,023	81,500
156	7	0,163	81,664	158	1	0,023	81,687	159	1	0,023	81,710
160	8	0,186	81,897	162	4	0,093	81,990	163	1	0,023	82,013
164	3	0,070	82,083	165	2	0,047	82,130	166	1	0,023	82,153
168	6	0,140	82,293	170	1	0,023	82,316	171	1	0,023	82,339
175	1	0,023	82,363	176	5	0,116	82,479	177	1	0,023	82,502
180	8	0,186	82,689	181	1	0,023	82,712	182	1	0,023	82,735
184	2	0,047	82,782	186	1	0,023	82,805	187	1	0,023	82,829
188	1	0,023	82,852	189	3	0,070	82,922	191	1	0,023	82,945
192	7	0,163	83,108	194	1	0,023	83,131	196	2	0,047	83,178
197	1	0,023	83,201	198	3	0,070	83,271	200	8	0,186	83,458
201	1	0,023	83,481	204	3	0,070	83,551	208	3	0,070	83,621
210	6	0,140	83,760	214	1	0,023	83,784	215	1	0,023	83,807
216	3	0,070	83,877	220	2	0,047	83,924	222	1	0,023	83,947
224	3	0,070	84,017	225	4	0,093	84,110	227	1	0,023	84,133
230	3	0,070	84,203	231	1	0,023	84,226	232	1	0,023	84,250
233	1	0,023	84,273	234	1	0,023	84,296	235	1	0,023	84,320
240	13	0,303	84,623	242	1	0,023	84,646	245	1	0,023	84,669
248	2	0,047	84,716	250	1	0,023	84,739	252	3	0,070	84,809
253	1	0,023	84,832	256	4	0,093	84,925	258	1	0,023	84,949
260	2	0,047	84,995	264	2	0,047	85,042	270	5	0,116	85,158
272	2	0,047	85,205	274	1	0,023	85,228	275	1	0,023	85,252
280	8	0,186	85,438	281	1	0,023	85,461	285	2	0,047	85,508
288	2	0,047	85,555	294	1	0,023	85,578	300	5	0,116	85,694
305	1	0,023	85,718	308	1	0,023	85,741	312	1	0,023	85,764
315	2	0,047	85,811	316	1	0,023	85,834	320	8	0,186	86,021
322	1	0,023	86,044	324	1	0,023	86,067	326	1	0,023	86,090
330	3	0,070	86,160	336	2	0,047	86,207	340	2	0,047	86,253
344	2	0,047	86,300	346	1	0,023	86,323	348	1	0,023	86,347
349	1	0,023	86,370	350	1	0,023	86,393	351	1	0,023	86,417
354	1	0,023	86,440	357	1	0,023	86,463	358	1	0,023	86,486
360	5	0,116	86,603	364	1	0,023	86,626	368	1	0,023	86,650
372	1	0,023	86,673	375	1	0,023	86,696	378	1	0,023	86,719
380	1	0,023	86,743	382	1	0,023	86,766	384	3	0,070	86,836
385	1	0,023	86,859	392	3	0,070	86,929	393	1	0,023	86,952
396	2	0,047	86,999	400	2	0,047	87,046	412	1	0,023	87,069
414	1	0,023	87,092	416	2	0,047	87,139	420	1	0,023	87,162
430	2	0,047	87,209	432	4	0,093	87,302	436	1	0,023	87,325
440	1	0,023	87,349	448	2	0,047	87,395	450	1	0,023	87,418
455	1	0,023	87,442	476	1	0,023	87,465	480	10	0,233	87,698
489	1	0,023	87,721	490	2	0,047	87,768	496	1	0,023	87,791
500	6	0,140	87,931	501	1	0,023	87,954	504	1	0,023	87,978
510	1	0,023	88,001	512	1	0,023	88,024	514	1	0,023	88,048
516	2	0,047	88,094	519	1	0,023	88,117	521	1	0,023	88,141
525	1	0,023	88,164	528	3	0,070	88,234	530	1	0,023	88,257
538	1	0,023	88,281	539	1	0,023	88,304	540	2	0,047	88,350
544	1	0,023	88,374	546	1	0,023	88,397	548	1	0,023	88,420
550	1	0,023	88,444	560	2	0,047	88,490	567	1	0,023	88,514
572	1	0,023	88,537	575	1	0,023	88,560	576	5	0,116	88,677
580	1	0,023	88,700	586	1	0,023	88,723	590	1	0,023	88,747
594	1	0,023	88,770	600	4	0,093	88,863	615	1	0,023	88,886
630	2	0,047	88,933	640	1	0,023	88,956	644	1	0,023	88,979
645	1	0,023	89,003	648	1	0,023	89,026	650	1	0,023	89,049
652	1	0,023	89,073	654	1	0,023	89,096	660	1	0,023	89,119
666	1	0,023	89,143	672	1	0,023	89,166	680	1	0,023	89,189
683	1	0,023	89,212	690	1	0,023	89,236	693	1	0,023	89,259
700	2	0,047	89,306	702	3	0,070	89,376	708	1	0,023	89,399
712	2	0,047	89,445	716	1	0,023	89,469	720	3	0,070	89,539
722	1	0,023	89,562	748	1	0,023	89,585	756	3	0,070	89,655

765	1	0,023	89,678	768	5	0,116	89,795	780	1	0,023	89,818
800	4	0,093	89,911	810	1	0,023	89,935	812	1	0,023	89,958
832	3	0,070	90,028	835	1	0,023	90,051	840	2	0,047	90,098
842	1	0,023	90,121	864	3	0,070	90,191	882	1	0,023	90,214
892	1	0,023	90,238	900	2	0,047	90,284	930	1	0,023	90,308
936	2	0,047	90,354	945	1	0,023	90,377	960	4	0,093	90,471
968	1	0,023	90,494	972	1	0,023	90,517	978	1	0,023	90,541
981	1	0,023	90,564	998	1	0,023	90,587	1000	2	0,047	90,634
1008	1	0,023	90,657	1024	1	0,023	90,680	1032	1	0,023	90,704
1050	1	0,023	90,727	1056	1	0,023	90,750	1064	1	0,023	90,774
1080	3	0,070	90,843	1092	1	0,023	90,867	1104	1	0,023	90,890
1111	1	0,023	90,913	1112	1	0,023	90,937	1120	2	0,047	90,983
1122	1	0,023	91,007	1128	1	0,023	91,030	1134	1	0,023	91,053
1144	1	0,023	91,076	1148	1	0,023	91,100	1152	2	0,047	91,146
1155	1	0,023	91,170	1170	1	0,023	91,193	1176	1	0,023	91,216
1184	1	0,023	91,240	1200	3	0,070	91,309	1244	1	0,023	91,333
1248	3	0,070	91,403	1260	3	0,070	91,473	1272	1	0,023	91,496
1280	2	0,047	91,542	1296	1	0,023	91,566	1300	1	0,023	91,589
1315	1	0,023	91,612	1320	1	0,023	91,636	1340	1	0,023	91,659
1350	1	0,023	91,682	1352	1	0,023	91,705	1355	1	0,023	91,729
1360	2	0,047	91,775	1368	1	0,023	91,799	1400	2	0,047	91,845
1404	2	0,047	91,892	1424	1	0,023	91,915	1440	4	0,093	92,008
1452	1	0,023	92,032	1470	1	0,023	92,055	1504	1	0,023	92,078
1512	1	0,023	92,102	1538	1	0,023	92,125	1550	1	0,023	92,148
1572	1	0,023	92,171	1584	2	0,047	92,218	1592	1	0,023	92,241
1596	1	0,023	92,265	1626	1	0,023	92,288	1632	1	0,023	92,311
1640	1	0,023	92,335	1680	2	0,047	92,381	1712	1	0,023	92,404
1736	1	0,023	92,428	1752	1	0,023	92,451	1800	2	0,047	92,498
1820	1	0,023	92,521	1848	1	0,023	92,544	1860	1	0,023	92,568
1872	1	0,023	92,591	1917	1	0,023	92,614	1920	2	0,047	92,661
1936	1	0,023	92,684	1944	1	0,023	92,707	1952	1	0,023	92,731
1972	1	0,023	92,754	1974	1	0,023	92,777	1980	1	0,023	92,801
2000	1	0,023	92,824	2016	2	0,047	92,870	2024	1	0,023	92,894
2025	1	0,023	92,917	2052	1	0,023	92,940	2088	1	0,023	92,964
2096	1	0,023	92,987	2128	1	0,023	93,010	2140	1	0,023	93,034
2160	2	0,047	93,080	2166	1	0,023	93,103	2176	2	0,047	93,150
2178	1	0,023	93,173	2190	1	0,023	93,197	2218	1	0,023	93,220
2250	1	0,023	93,243	2259	1	0,023	93,267	2274	1	0,023	93,290
2288	1	0,023	93,313	2320	2	0,047	93,360	2336	1	0,023	93,383
2340	1	0,023	93,406	2361	1	0,023	93,430	2400	3	0,070	93,500
2435	1	0,023	93,523	2440	1	0,023	93,546	2448	2	0,047	93,593
2480	1	0,023	93,616	2532	1	0,023	93,639	2560	1	0,023	93,663
2600	1	0,023	93,686	2640	1	0,023	93,709	2658	1	0,023	93,733
2680	1	0,023	93,756	2688	1	0,023	93,779	2704	1	0,023	93,802
2720	1	0,023	93,826	2756	1	0,023	93,849	2772	1	0,023	93,872
2854	1	0,023	93,896	2880	3	0,070	93,966	2916	1	0,023	93,989
2952	1	0,023	94,012	2970	2	0,047	94,059	3072	1	0,023	94,082
3076	1	0,023	94,105	3140	1	0,023	94,129	3150	1	0,023	94,152
3208	1	0,023	94,175	3234	1	0,023	94,199	3244	1	0,023	94,222
3264	1	0,023	94,245	3312	1	0,023	94,268	3360	1	0,023	94,292
3388	1	0,023	94,315	3456	1	0,023	94,338	3468	1	0,023	94,362
3472	1	0,023	94,385	3586	1	0,023	94,408	3630	1	0,023	94,432
3639	1	0,023	94,455	3756	1	0,023	94,478	3958	1	0,023	94,501
3960	1	0,023	94,525	3994	1	0,023	94,548	3996	1	0,023	94,571
4007	1	0,023	94,595	4032	1	0,023	94,618	4080	1	0,023	94,641
4087	1	0,023	94,664	4160	1	0,023	94,688	4320	1	0,023	94,711
4366	1	0,023	94,734	4400	1	0,023	94,758	4480	1	0,023	94,781
4485	1	0,023	94,804	4536	2	0,047	94,851	4600	1	0,023	94,874
4608	1	0,023	94,897	4644	1	0,023	94,921	4680	1	0,023	94,944
4704	1	0,023	94,967	4784	1	0,023	94,991	4800	2	0,047	95,037
4809	1	0,023	95,061	4815	1	0,023	95,084	4816	1	0,023	95,107
4832	1	0,023	95,130	4876	1	0,023	95,154	5040	1	0,023	95,177
5130	1	0,023	95,200	5280	1	0,023	95,224	5338	1	0,023	95,247
5568	1	0,023	95,270	5616	1	0,023	95,294	5670	1	0,023	95,317
5760	2	0,047	95,363	5796	1	0,023	95,387	6048	2	0,047	95,433
6336	1	0,023	95,457	6656	1	0,023	95,480	6672	1	0,023	95,503
6732	1	0,023	95,527	6750	1	0,023	95,550	6840	2	0,047	95,596
6912	1	0,023	95,620	7080	1	0,023	95,643	7096	1	0,023	95,666
7200	1	0,023	95,690	7260	1	0,023	95,713	7480	1	0,023	95,736
7560	2	0,047	95,783	7692	1	0,023	95,806	7696	1	0,023	95,829

7872	1	0,023	95,853	7875	1	0,023	95,876	7884	1	0,023	95,899
8064	1	0,023	95,923	8316	1	0,023	95,946	8580	1	0,023	95,969
8624	1	0,023	95,993	8643	1	0,023	96,016	8802	1	0,023	96,039
8925	1	0,023	96,062	8988	1	0,023	96,086	9000	1	0,023	96,109
9088	1	0,023	96,132	9360	1	0,023	96,156	9400	1	0,023	96,179
9520	1	0,023	96,202	9600	1	0,023	96,226	10100	1	0,023	96,249
10512	1	0,023	96,272	11696	1	0,023	96,295	12096	1	0,023	96,319
12600	2	0,047	96,365	12640	1	0,023	96,389	13048	1	0,023	96,412
13500	1	0,023	96,435	13598	1	0,023	96,459	13776	1	0,023	96,482
13824	1	0,023	96,505	14016	1	0,023	96,528	14112	1	0,023	96,552
14208	1	0,023	96,575	14480	1	0,023	96,598	14652	1	0,023	96,622
14751	1	0,023	96,645	14760	1	0,023	96,668	15120	1	0,023	96,692
15166	1	0,023	96,715	15360	1	0,023	96,738	15680	1	0,023	96,761
15960	1	0,023	96,785	16640	1	0,023	96,808	16800	1	0,023	96,831
17280	1	0,023	96,855	17490	1	0,023	96,878	17496	1	0,023	96,901
17640	1	0,023	96,925	17784	1	0,023	96,948	17850	1	0,023	96,971
17920	1	0,023	96,994	18000	1	0,023	97,018	19350	1	0,023	97,041
20560	1	0,023	97,064	20736	1	0,023	97,088	20964	1	0,023	97,111
21600	1	0,023	97,134	21808	1	0,023	97,158	21848	1	0,023	97,181
21950	1	0,023	97,204	22451	1	0,023	97,227	22488	1	0,023	97,251
22680	1	0,023	97,274	23816	1	0,023	97,297	25984	1	0,023	97,321
26218	1	0,023	97,344	26964	1	0,023	97,367	31542	1	0,023	97,390
31696	1	0,023	97,414	32080	1	0,023	97,437	35600	1	0,023	97,460
36000	1	0,023	97,484	36408	1	0,023	97,507	37440	1	0,023	97,530
37586	1	0,023	97,554	38112	1	0,023	97,577	39420	1	0,023	97,600
40950	1	0,023	97,623	41040	1	0,023	97,647	51450	1	0,023	97,670
53040	1	0,023	97,693	53100	1	0,023	97,717	54728	1	0,023	97,740
54800	1	0,023	97,763	56430	1	0,023	97,787	56704	1	0,023	97,810
58425	1	0,023	97,833	63288	1	0,023	97,856	63404	1	0,023	97,880
63468	1	0,023	97,903	65064	1	0,023	97,926	68880	1	0,023	97,950
71265	1	0,023	97,973	72576	1	0,023	97,996	74470	1	0,023	98,020
74520	1	0,023	98,043	76916	1	0,023	98,066	82800	1	0,023	98,089
86240	1	0,023	98,113	87175	1	0,023	98,136	87192	1	0,023	98,159
87360	1	0,023	98,183	88226	1	0,023	98,206	88704	1	0,023	98,229
101304	1	0,023	98,253	116640	1	0,023	98,276	118800	1	0,023	98,299
120428	1	0,023	98,322	123202	1	0,023	98,346	126345	1	0,023	98,369
136080	1	0,023	98,392	152568	1	0,023	98,416	153090	1	0,023	98,439
158340	1	0,023	98,462	163800	1	0,023	98,486	166360	1	0,023	98,509
168180	1	0,023	98,532	183804	1	0,023	98,555	184992	1	0,023	98,579
200000	1	0,023	98,602	205440	1	0,023	98,625	224000	1	0,023	98,649
225000	1	0,023	98,672	226842	1	0,023	98,695	226848	1	0,023	98,719
228096	1	0,023	98,742	243648	1	0,023	98,765	253368	1	0,023	98,788
262848	1	0,023	98,812	267264	1	0,023	98,835	285870	1	0,023	98,858
307704	1	0,023	98,882	310080	1	0,023	98,905	314496	1	0,023	98,928
340200	1	0,023	98,952	372240	1	0,023	98,975	392156	1	0,023	98,998
437904	1	0,023	99,021	471744	1	0,023	99,045	485184	1	0,023	99,068
595350	1	0,023	99,091	618240	1	0,023	99,115	656502	1	0,023	99,138
792480	1	0,023	99,161	793800	1	0,023	99,185	800208	1	0,023	99,208
870912	1	0,023	99,231	882792	1	0,023	99,254	1003520	1	0,023	99,278
1332918	1	0,023	99,301	1394800	1	0,023	99,324	1436400	1	0,023	99,348
1843200	1	0,023	99,371	1940400	1	0,023	99,394	1946112	1	0,023	99,418
1952588	1	0,023	99,441	2070474	1	0,023	99,464	2552760	1	0,023	99,487
2565600	1	0,023	99,511	2569728	1	0,023	99,534	2711376	1	0,023	99,557
3464208	1	0,023	99,581	3892328	1	0,023	99,604	3995708	1	0,023	99,627
15372816	1	0,023	99,651	16907403	1	0,023	99,674	27447552	1	0,023	99,697
36834000	1	0,023	99,720	38707200	1	0,023	99,744	39053720	1	0,023	99,767
50101216	1	0,023	99,790	66772134	1	0,023	99,814	419126400	1	0,023	99,837
1792391040	1	0,023	99,860	2190154400	1	0,023	99,884	overflow	5	0,116	100,000

La celda `[overflow 5 0,116 100,000]` indica que hay 5 frases para las cuales el número de ambigüedades es mayor que $2^{32} = 4{,}294967\text{e+}09$, el valor más alto que puede almacenar un entero largo en nuestro sistema. El número medio de ambigüedades por frase, sin considerar estas 5, viene dado por:

$$\#\text{ medio de ambigüedades por frase} = \frac{\#\text{ ambigüedades}}{\#\text{ frases}} = \frac{4.748.032.872}{4.292 - 5} \approx 1.107.542.$$

Sin embargo, resulta mucho más interesante calcular el valor de la mediana de esta distribución, es decir, el punto que deja la mitad de la población a su izquierda y la otra mitad a la derecha.

Las últimas cifras de las celdas [7 42 0.979 48.695] y [8 136 3.169 51.864] indican que la mediana se sitúa entre 7 y 8 ambigüedades por frase.

E.2 Nivel de acierto de la gramática SUSANNE sobre las frases sin trazas

Las siguientes filas de números constituyen los datos sobre el nivel de acierto de la gramática SUSANNE al analizar todas las frases de la parte sin trazas. Estas cifras se resumen gráficamente en la figura 9.2. Cada celda contiene cuatro cifras: posición del árbol de referencia dentro del conjunto de todos los árboles que proporciona el analizador para cada frase ordenados del más probable al menos probable, número de frases con esa posición, frases que presentan ese número de ambigüedades, porcentaje de dichas frases sobre el total, y porcentaje de frases acumulado.

1	3335	77,703	77,703	2	447	10,415	88,117	3	118	2,749	90,867	4	59	1,375	92,241
5	17	0,396	92,637	6	22	0,513	93,150	7	20	0,466	93,616	8	11	0,256	93,872
9	11	0,256	94,129	10	7	0,163	94,292	11	3	0,070	94,362	12	5	0,116	94,478
13	4	0,093	94,571	16	1	0,023	94,595	17	3	0,070	94,664	18	1	0,023	94,688
19	1	0,023	94,711	20	2	0,047	94,758	21	1	0,023	94,781	22	1	0,023	94,804
23	1	0,023	94,828	25	3	0,070	94,897	27	3	0,070	94,967	28	1	0,023	94,991
29	1	0,023	95,014	31	1	0,023	95,037	38	1	0,023	95,061	61	1	0,023	95,084
67	1	0,023	95,107	79	1	0,023	95,130	322	1	0,023	95,154	ne	208	4,846	100,000

La celda [ne 208 4,846 100,000] indica que existen 208 frases para las cuales el número de interpretaciones es mayor que 5.000, un límite artificial que hemos introducido en el analizador para reducir el tiempo de cálculo, y por tanto la posición del árbol de referencia es no encontrado. La posición media por frase, sin considerar estas 208, es:

$$\text{posición media} = \frac{\text{suma de las posiciones}}{\#\text{ frases}} = \frac{6.599}{4.292 - 208} = 1,61.$$

Además, la última cifra de la celda [1 3335 77,703 77,703] muestra que el análisis más probable coincide con el árbol de referencia para el 77,703% de las frases.

E.3 Ambigüedades de la gramática SUSANNE sobre las frases con trazas

Las siguientes filas de números constituyen los datos sobre las ambigüedades de la gramática SUSANNE al analizar todas las frases de la parte con trazas. Cada celda contiene cuatro cifras: número de ambigüedades, número de frases que presentan ese número de ambigüedades, porcentaje de dichas frases sobre el total, y porcentaje de frases acumulado.

0	1979	97,057	97,057	1	6	0,294	97,351	2	9	0,441	97,792
4	3	0,147	97,939	6	2	0,098	98,037	7	1	0,049	98,086
8	1	0,049	98,135	12	2	0,098	98,233	24	3	0,147	98,380
26	1	0,049	98,429	30	1	0,049	98,478	37	1	0,049	98,527
46	1	0,049	98,576	103	1	0,049	98,625	110	1	0,049	98,674
112	1	0,049	98,723	136	1	0,049	98,722	145	1	0,049	98,821
187	1	0,049	98,870	245	1	0,049	98,919	334	1	0,049	98,968
396	1	0,049	99,017	441	1	0,049	99,066	714	1	0,049	99,115
720	1	0,049	99,164	864	1	0,049	99,213	1238	1	0,049	99,262
1441	1	0,049	99,311	2448	1	0,049	99,360	2960	1	0,049	99,409
3885	1	0,049	99,458	6034	1	0,049	99,507	6163	1	0,049	99,556
11291	1	0,049	99,605	21754	1	0,049	99,654	25962	1	0,049	99,703
71500	1	0,049	99,752	76920	1	0,049	99,801	288192	1	0,049	99,850
504630	1	0,049	99,898	2291058	1	0,049	99,957	overflow	1	0,049	100,000

La celda [0 1979 97,057 97,057] indica que existen 1.979 frases no analizables, es decir, la gramática SUSANNE es capaz de analizar sólo 60 frases de la parte con trazas. La celda [overflow 1 0,049 100,000] indica que hay 1 frase para la cual el número de ambigüedades es mayor que $2^{32} = 4.294967e+09$, el valor más alto que puede almacenar un entero largo en nuestro sistema. El número medio de ambigüedades por frase, sin considerar esta última ni tampoco las frases no analizables, viene dado por:

$$\#\text{ medio de ambigüedades por frase} = \frac{\#\text{ ambigüedades}}{\#\text{ frases}} = \frac{3.320.281}{59} \approx 56.276.$$

Una vez más, resulta mucho más interesante calcular el valor de la mediana de esta distribución. Si sumamos el número de frases con 30 ambigüedades o menos, obtenemos 29 frases, lo cual es menor que la mitad de la población de frases analizables (29,5). Y si sumamos el número de frases con 37 ambigüedades (el siguiente valor para el número de ambigüedades en esta distribución) o más, obtenemos 30 frases, lo cual ya es mayor que la mitad de la población. Esto indica que la mediana se sitúa entre 30 y 37 ambigüedades por frase.

E.4 Nivel de acierto de la gramática SUSANNE sobre las frases con trazas

Solamente 60 de las 2.039 frases de la parte con trazas pueden ser analizadas con la gramática SUSANNE, y por supuesto ninguna de ellas presenta un árbol de análisis igual al de referencia. Esto es debido a que ninguno de los árboles obtenidos con la gramática SUSANNE presenta trazas, mientras que todos los árboles de referencia de las frases de esta parte del corpus sí presentan trazas.

E.5 Productos cruzados de etiquetas de las frases con trazas

Las siguientes filas de números constituyen los datos sobre el producto cruzado de las etiquetas de las palabras para cada frase de la parte con trazas del corpus SUSANNE. Este valor representa el número total de posibles etiquetaciones para cada frase. Estas cifras se resumen gráficamente en la figura 9.3. Cada celda contiene cinco cifras: número de posibles etiquetaciones, número de frases que presentan ese número de posibles etiquetaciones, número medio de palabras por frase, porcentaje de dichas frases sobre el total, y porcentaje de frases acumulado.

1e+0	1	6	0,049	0,049	1e+1	22	7	1,078	1,128	1e+2	91	9	4,462	5,590
1e+3	170	13	8,337	13,928	1e+4	196	16	9,612	23,541	1e+5	251	19	12,310	35,850
1e+6	217	22	10,642	46,493	1e+7	229	26	11,231	57,724	1e+8	209	29	10,250	67,974
1e+9	186	31	9,122	77,096	1e+10	154	33	7,552	84,649	1e+11	100	36	4,904	89,553
1e+12	78	40	3,825	93,379	1e+13	54	42	2,648	96,027	1e+14	38	44	1,863	97,891
1e+15	28	45	1,373	99,264	1e+16	9	45	0,441	99,705	1e+17	2	44	0,098	99,803
1e+18	1	50	0,049	99,852	1e+19	2	50	0,098	99,951	1e+20	1	60	0,049	100,000

El número medio de palabras por frase es 25, y la mediana es 23. El número medio de posibles etiquetaciones por frase es 10^{18}, y la mediana es 10^{6}.

Bibliografía

Abney, S. (1996). Part-of-speech tagging and partial parsing. In S. Young and G. Bloothooft (eds.), *Corpus-Based Methods in Language and Speech Processing*, pp. 118-136. Dordrecht: Kluwer Academic.

Aho, A.V.; Sethi, R.; Ullman, J.D. (1985). Compilers: principles, techniques and tools. *Addison-Wesley*, Reading, MA.

Alonso Pardo, M. (2000). Interpretación tabular de autómatas para lenguajes de adjunción de árboles. *Tesis Doctoral*, Departamento de Computación, Universidad de La Coruña.

Bahl, L.R.; Mercer, R.L. (1976). Part-of-speech assignement by a statistical decision algorithm. In *International Symposium on Information Theory*, Ronneby (Sweden).

Baldi, P.; Brunak, S. (1998). Bioinformatics: the machine learning approach. *The MIT Press*, Cambridge, MA.

Baker, J.K. (1975). Stochastic modeling for automatic speech undestanding. In D. Raj Reddy (ed.), *Speech Recognition: Invited papers presented at the 1974 IEEE symposium*, pp. 297-307. Academic Press, NY.

Barcala Rodríguez, M. (1999). Diseño e implementación de una herramienta de análisis sintáctico estocástico para el procesamiento de textos en lenguaje natural. *Proyecto Fin de Carrera en Ingeniería Informática*, dirigido por J. Graña Gil en el Departamento de Computación de la Universidad de La Coruña.

Barcala Rodríguez, M.; Sacristán Agulló, O.; Graña Gil, J. (2000). Análisis sintáctico estocástico y paralelismo. *Revista de la Sociedad Española para el Procesamiento del Lenguaje Natural*, vol. 26 (Septiembre), pp. 125-132.

Baum, L.E. (1972). An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, vol. 3, pp. 1-8.

Baum, L.E.; Petrie, L.; Soules, G.; Weiss, N. (1970). A maximization technique ocurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, vol. 41, pp. 164-171.

Benello, J.; Mackie, A.W.; Anderson, J.A. (1989). Syntactic category disambiguation with neural networks. *Computer Speech and Language*, vol. 3, pp. 203-217.

Berger, A.; Della Prieta, S.A.; Della Prieta, V.J.; (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, vol. 22(1), pp. 39-71.

Brants, T. (1996). TnT - A statistical part-of-speech tagger. *Technical Report*, Saarland University, Computational Linguistics. Saarbrücken (Germany).

Brants, T. (1998). Estimating hidden Markov model topologies. In J. Ginzburg, Z. Khasidashvili, C. Vogel, J.-J. Lévy and E. Vallduví (eds.), *The Tbilisi Symposium on Logic, Language and Computation: Selected Papers*, pp. 163-176. *CSLI Publications*, Stanford, CA.

Brants, T. (2000). TnT - A statistical part-of-speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP-2000)*, Seattle, WA.

Brill, E. (1992). A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, Trento (Italy), pp. 152-155.

Brill, E. (1993a). Automatic grammar induction and parsing free text: a transformation-based approach. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 259-265.

Brill, E. (1993b). A corpus-based approach to language learning. *PhD. Thesis*, University of Pennsylvania.

Brill, E. (1993c). Tansformation-based error-driven parsing. In *Proceedings of the Third International Workshop on Parsing Technologies*, Tilburg/Durbuy (The Netherlands/Belgium).

Brill, E. (1994). Some advances in rule-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, Seattle, WA.

Brill, E. (1995a). Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. *Computational Linguistics*, vol. 21, pp. 543-565.

Brill, E. (1995b). Unsupervised learning of disambiguation rules for part of speech tagging. In *Proceedings of the Third Workshop on Very Large Corpora*, pp. 1-13.

Brill, E.; Magerman, D.M.; Marcus, M.P.; Santorini, B. (1990). Deducing linguistic structure from the statistics of large corpora. In M. Kaufmann (ed.), *Proceedings of the DARPA Speech and Natural Language Workshop*, pp. 275-282, San Mateo, CA.

Brill, E.; Resnik, P. (1994). A transformation-based approach to prepositional phrase attachment disambiguation. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94)*, pp. 1198-1204.

Brown, P.F.; Della Prieta, S.A.; Della Prieta, V.J.; Mercer, R.L.; Resnik, P.S. (1991). Language modelling using decision trees. *IBM Research Report*, Yorktown Heights, NY.

Burke, R.; Hammond, K.; Kulyukin, V.; Lytinen, S.; Tomuro, N.; Schoenberg, S. (1997). Question answering form frequently asked files. *AI Magazine*, vol. 18, pp. 57-66.

Burnard, L. (1995). What it SGML and how does it help? *Computers and the Humanities*, vol. 29(1), pp. 41-50.

Calzolari, N.; McNaught, J. (1996). Eagles Editor's Introduction. Eagles Document EAG-EB-FR-1, Expert Advisory Group on Language Engineering Standars.

Cardie, C. (1997). Empirical methods in information extraction. *AI Magazine*, vol. 18, pp. 65-79.

Chanod, J.-P.; Tapanainen, P. (1995). Tagging French - comparing a statistical and a constraint-based method. In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 149-156.

Chappelier, J.-C.; Rajman, M. (1998). A practical bottom-up algorithm for on-line parsing with stochastic context-free grammars. *Rapport Technique 98/284*, Departement d'Informatique, Ecole Polytechnique Fédérale de Lausanne (Switzerland).

Charniak, E. (1993). Statistical language learning. *The MIT Press*, Cambridge, MA.

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI'97)*, pp. 598-603.

Charniak, E.; Hendrickson, C.; Jacobson, N.; Perkowitz, M. (1993). Equations for part-of-speech tagging. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp. 784-789. Menlo Park, CA.

Chi, Z.; Geman, S. (1998). Estimation of probabilistic context-free grammars. *Computational Linguistics*, vol. 24, pp. 299-305.

Chomsky, N. (1957). Syntactic structures. The Hague: Mouton.

Church, K.W. (1988). A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing*, pp. 136-143.

Church, K.W.; Gale, W.A. (1991). A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams. *Computer Speech and Language*, vol. 5, pp. 19-54.

Ciarlet, P.G. (1988). Introduction à l'analyse numérique matricielle et à l'optimisation. *Masson*, Paris.

Ciarlet, P.G.; Miara, B.; Thomas, J.M. (1991). Exercices d'analyse numérique matricielle et d'optimisation avec solutions. 2e. édition. *Masson*, Paris.

CRATER project. (1993). Corpus Resources And Terminology ExtRaction (MLAP-93/20), creation of a set of tools and resources for multilingual corpus linguistics work. Lancaster University, UK; Computers, Communications and Visions, France; Universidad Autónoma de Madrid, Spain. `http://www.lllf.uam.es/~fernando/projects/es_corpus.html`

Cutting, D.; Kupiec, J.; Pedersen, J.; Sibun, P. (1992). A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, Trento (Italy), pp. 133-140.

Daciuk, J.; Mihov, S.; Watson, B.W.; Watson, R.E. (2000). Incremental construction of minimal acyclic finite-state automata. *Computational Linguistics*, vol. 26(1), pp. 3-16.

Daelemans, W.; Zavrel, J.; Berck, P.; Gillis, S. (1996). MBT: A memory-based part of speech tagger generator. In *Proceedings of the Fourth Workshop on Very Large Corpora*, pp. 14-27.

Darroch, J.N.; Ratcliff, D. (1972). Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, vol. 43(5), pp. 1470-1480.

Della Prieta, S.A.; Della Prieta, V.J.; Lafferty, J. (1995). Inducing features of random fields. *Technical Report* CMU-CS95-144, School of Computer Science, Carnegie-Mellon University.

Dempster, A.P.; Laird, N.M.; Rubin, D.B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Stat. Soc.*, vol. 39(1), pp. 1-38.

Dermatas, E.; Kokkinakis, G. (1995). Automatic stochastic tagging of natural language texts. *Computational Linguistics*, vol. 21, pp. 137-164.

DeRose, S.J. (1988). Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, vol. 14, pp. 31-39.

Derouault, A.-M.; Merialdo, B. (1986). Natural language modeling for phoneme-to-text transcription. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, pp. 642-649.

Dini, L.; Di Tomaso, V.; Segond, F. (1998). Error-driven word sense disambiguation. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics/17th International Conference on Computational Linguistics (COLING-98)*, pp. 320-324.

Durbin, R.; Eddy, S.; Krogh, A.; Mitchison, G. (1998). Biological sequence analysis: probabilistic models of proteins and nucleic acids. *Cambridge University Press.*

Earley, J. (1970). An efficient context-free parsing algorithm. *Communications of the ACM*, vol. 13(2), pp. 94-102.

Elworthy, D. (1994). Does Baum-Welch re-estimation help taggers? In *Proceedings of the Fourth Conference on Applied Natural Language Processing*, pp. 53-58.

Erbach, G. (1994). Bottom-up Earley deduction. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-94)*, Kyoto (Japan).

Fagan, J.L. (1987). Automatic phrase indexing for document retrieval: an examination of syntactic and non-syntactic methods. In *Proceedings of the 10th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval.*

Forney, G.D. (1973). The Viterbi algorithm. In *Proceedings of the IEEE*, vol. 61 (March), pp. 268-278.

Foster, G.F. (1991). Statistical lexical disambiguation. *Master's thesis*, School of Computer Science, McGill University.

Francis, W.N.; Kučera, H. (1982). Frequency analysis of English usage. *Houghton Mifflin Company*, Boston, MA.

Franz, A. (1996). Automatic ambiguity resolution in natural language processing. *Lecture Notes in Artificial Intelligence*, vol. 1171, Springer Verlag, Berlin.

Franz, A. (1997). Independence assumptions considered harmful. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics/8th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 182-189.

Garside, R.G.; Leech, G.N.; Sampson, G.R. (eds.) (1987). The computational analysis of English: a corpus-based approach. *Longman*, London.

Gold, E.M. (1967). Language identification in the limit. *Information and Control*, vol. 10, pp. 447-474.

González Collar, A.L.; Goñi Menoyo, J.M.; González Cristóbal, J.C. (1995). Un analizador morfológico para el castellano basado en chart. En *Actas de la VI Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA'95)*, Alicante, pp. 343-352.

Graña Gil, J.; Alonso Pardo, M.A.; Valderruten Vidal, A. (1994). Análisis léxico no determinista: etiquetación eficiente del lenguaje natural. *Technical Report* 16, Departamento de Computación, Universidad de La Coruña.

Graña Gil, J.; Chappelier, J.-C.; Rajman, M. (1999). Using syntactic constraints in natural language disambiguation. *Rapport Technique 99/315*, Departement d'Informatique, Ecole Polytechnique Fédérale de Lausanne (Switzerland).

Graña Gil, J.; Rajman, M. (1999). Disambiguation experiment for Spanish. *Rapport Technique 99/314*, Departement d'Informatique, Ecole Polytechnique Fédérale de Lausanne (Switzerland).

Grassmann, W.K.; Tremblay, J.P. (1996). Matemática discreta y lógica: una perspectiva desde la ciencia de la computación. *Prentice-Hall*, Madrid.

Greene, B.B.; Rubin, G.M. (1971). Automatic grammatical tagging of English. *Technical Report*, Brown University, Providence, RI.

Hays, D.G. (1967). Introduction to computational linguistics. *American Elsevier*, NY.

Hopcroft, J.E.; Ullman, J.D. (1979). Introduction to automata theory, languages and computations. *Addison-Wesley*, Reading, MA.

Horning, J.J. (1969). A study of grammatical inference. *PhD. Thesis*, University of Stanford, CA.

Jacquemin, C.; Klavans, J.L.; Tzoukermann, E. (1997). Expansion of multi-words terms for indexing and retrieval using morphology and syntax. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics/8th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 24-31.

Jelinek, F. (1976). Continuous speech recognition by statistical methods. *IEEE*, vol. 64, pp. 532-556.

Jelinek, F. (1985). Markov source modeling of text generation. In J.K. Skwirzynski (ed.), *The Impact of Processing Techniques on Communications*, E91 of *NATO ASI series*, pp. 569-598. Dordrecht: M. Nijhoff.

Jelinek, F. (1997). Statistical methods for speech recognition. *The MIT Press*, Cambridge, MA.

Jelinek, F.; Bahl, L.R.; Mercer, R.L. (1975). Design of a linguistic statistical decoder for the recognition of continuous sppech. *IEEE Transactions on Information Theory*, vol. 21, pp. 250-256.

Jelinek, F.; Mercer, R.L. (1985). Probability distribution estimation from sparse data. *IBM Technical Disclosure Bulletin*, vol. 28, pp. 2591-2594.

Karlsson, F.; Voutilainen, A.; Heikkilä, J.; Anttila, A. (1995). Constraint grammar: a language-independent system for parsing unrestricted text. *Mouton de Gruyer*, Berlin.

Karttunen, L.; Beesley, K. (1992). Two-level rule compiler. *Technical Report* ISTL-92-2, Xerox, Palo Alto Research Center, CA.

Kasami, T. (1965). An efficient recognition and syntax analysis algorithm for context-free languages. *Technical Report*, AF CRL-65-758, Air Force Cambridge Research Laboratory, Bedford, MA.

Katz, S. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 35(3), pp. 400-401.

Kelley, D. (1998). Teoría de autómatas y lenguajes formales. *Prentice-Hall*, Madrid.

Kempe, A. (1997). Finite states transducers approximating hidden Markov models. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics/8th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 460-467.

Klein, S.; Simmons, R.F. (1963). A computational approach to grammatical coding of English words. *Journal of the Association for Computing Machinery*, vol. 10, pp. 334-347.

Koskennimi, K. (1983). Two-level morphology: A general computational model for word-form recognition and production. *Publications* 11, Department of General Linguistics, University of Helsinki.

Kupiec, J. (1992). Robust part-of-speech tagging using a Hidden Markov Model. *Computer Speech and Language*, vol. 6, pp. 225-242.

Kupiec, J. (1993). MURAX: A robust linguistic approach for question answering using an on-line encyclopedia. In *Proceedings of the 16th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pp. 181-190.

Lau, R.; Rosenfeld, R.; Roukos, S. (1993). Adaptative language modeling using the maximum entropy principle. In *Proceedings of the Human Language Technology Workshop (ARPA)*, pp. 108-113.

Levinson, S.E.; Rabiner, L.R.; Sondhi, M.M. (1983). An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech reconigtion. *Bell System Technical Journal*, vol. 62, pp. 1035-1074.

Lucchesi, C.L.; Kowaltowski, T. (1993). Applications of finite automata representing large vocabularies. *Software - Practice and Experience*, vol. 23(1), pp. 15-30.

Magerman, D.M. (1995). Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics.*

Manning, C.D.; Schütze, H. (1999). Foundations of statistical natural language processing. *The MIT Press*, Cambridge, MA.

Marcus, M.P.; Santorini, B.; Marcinkiewicz, M.A. (1994). Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, vol. 19(2), pp. 313-330.

Markov, A.A. (1913). An example of statistical investigation in the text of Eugene Onyegin illustrating coupling of tests in chains. In *Proceedings of the Academy of Sciences, St. Petersburg*, vol. 7 of VI, pp. 153-162.

Marques, N.C.; Pereira Lopes, G. (1996). A neural network approach to part-of-speech tagging. In *Proceedings of the XIII Simpósio Brasileiro de Inteligência Artificial (SBIA-96)*, Ciuritiba (Brasil).

Màrquez, L.; Padró, L. (1997). A flexible POS tagger using an automatically acquired language model. In *Proceedings of joint ACL/EACL-97*, Madrid.

Màrquez, L.; Rodríguez, H. (1997). Automatically acquiring a language model for POS tagging using decision trees. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-97)*, Tzigov Chark (Bulgaria).

Marshall, I. (1987). Tag selection using probabilistic methods. In R. Garside, G. Leech and G. Sampson (eds.), *The computational analysis of English: a corpus-based approach*, pp. 42-65. Longman, London.

McMahon, J.G.; Smith, F.J. (1996). Improving statistical language model performance with automatically generated word hierarchies. *Computational Linguistics*, vol. 22, pp. 217-247.

Merialdo, B. (1994). Tagging English text with a probabilistic model. *Computational Linguistics*, vol. 20, pp. 155-171.

Mikheev, A. (1997). Automatic rule induction for unknown-word guessing. *Computational Linguistics*, vol. 23(3), pp. 405-423.

Miller, D.; Leek, T.; Schwartz, R. (1999). A hidden Markov model information retrieval system. In *Proceedings of the 22nd. Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pp. 214-221.

Mohri, M. (1995). On some applications of finite-state automata theory to natural language processing. *Natural Language Engineering*, Cambridge University Press, vol. 1(1), pp. 1-21.

Mohri, M. (1997). Finite-state transducers in language and speech processing. *Computational Linguistics*, vol. 23(2), pp. 269-311.

Moreno Sandoval, A. (1991). Un modelo computacional basado en la unificación para el análisis y la generación de la morfología del español. *Tesis Doctoral*, Departamento de Lingüística, Lenguas Modernas, Lógica y Filosofía de la Ciencia, Universidad Autónoma de Madrid.

Moreno Sandoval, A.; Goñi Menoyo, J.M. (1995). GRAMPAL: A morphological model and processor for Spanish implemented in Prolog. In M. Sessa and M. Alpuente (eds.), *Proceedings of the Joint Conference on Declarative Programming (GULP-PRODE'95)*, Marina di Vietri (Italy), pp. 321-331.

Nadas, A. (1991). Good, Jelinek, Mercer and Robbins on Turing's estimate of probabilities. *American Journal of Mathematical and Management Sciences*, vol. 11(3-4), pp. 229-308.

Ney, H.; Essen, U.; Kneser, R. (1995). On the estimation of "small" probabilities by leaving-one-out. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17(12) (December), pp. 1202-1212.

Padró, L. (1996). POS tagging using relaxation labelling. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, Copenhagen, Denmark.

Porta, J. (1996). RTAG. *Technical Report*, Grupo de Investigación de Lingüística Computacional, Universidad de Barcelona.

Quenouille, M.H. (1949). Approximate tests of correlation in time-series. *Journal of the Royal Statistical Society, Series B*, vol. 11, pp. 68-84.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, vol. 77, pp. 257-286.

Rabiner, L.; Juang, B.H. (1993). Fundamentals of speech recognition. *Prentice-Hall*, Englewood Cliffs, NJ.

Ramshaw, L.A.; Marcus, M.P. (1994). Exploring the statistical derivation of transformational rules sequences for part-of-speech tagging. In *The Balancing Act. Proceedings of the Workshop*, Association of Computational Linguistics, pp. 86-95, Morristown, NJ.

Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In *Proceedings of the 1st Conference on Empirical Methods in Natural Language Processing*, pp. 133-142.

Ratnaparkhi, A.; Reynar, J.; Roukos, S. (1994). A maximum entropy model for prepositional phrase attachment. In *Proceedings of the Human Language Technology Workshop (ARPA)*, pp. 250-255.

Revuz, D. (1992). Minimization of acyclic deterministic automata in linear time. *Theoretical Computer Science*, vol. 92(1), pp. 181-189.

Ristad, E.S.; Thomas, R.G. (1997). Hierarchical non-emitting Markov models. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics/8th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 381-385.

Roche, E.; Schabes, Y. (1995). Deterministic part-of-speech tagging with finite-state transducers. *Computational Linguistics*, vol. 21(2), pp. 227-253.

Russell, G.; Petitpierre, D. (1995). MMORPH - The Multext Morphology Program, version 2.3. *MULTEXT deliverable report.*

Sacristán Agulló, O. (1999). Diseño e implementación de un modelo estocástico para la etiquetación de textos en lenguaje natural. *Proyecto Fin de Carrera en Ingeniería Informática*, dirigido por J. Graña Gil en el Departamento de Computación de la Universidad de La Coruña.

Salton, G.; Thorpe, R.W. (1962). An approach to the segmentation problem in speech analysis and language translation. In *Proceedings of the 1961 International Conference on Machine Translation of Languages and Applied Language Analysis*, vol. 2, pp. 703-724.

Sampson, G. (1994a). The SUSANNE corpus, release 3, 04/04/1994. School of Cognitive & Computing Sciences, University of Sussex, Falmer, Brighton (England).

Sampson, G. (1994b). English for the computer. *Oxford University Press.*

Samuelsson, C. (1993). Morphological tagging based entirely on Bayesian inference. In *Proceedings of the 9th Nordic Conference on Computational Linguistics*, Stockholm University, Stockholm, Sweden.

Samuelsson, C.; Tapanainen, P.; Voutilainen, A. (1996). Inducing constraint grammars. In *Proceedings of the Third International Colloquium on Grammatical Inference.*

Samuelsson, C.; Voutilainen, A. (1997). Comparing a linguistic and a stochastic tagger. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics/8th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 246-253.

Sánchez León, F. (1994). Spanish tagset for the CRATER project. *Technical Report*, Facultad de Filosofía y Letras, Universidad Autónoma de Madrid.

Sánchez León, F.; Nieto Serrano, A.F. (1995). Development of a Spanish version of the XEROX tagger. *Technical Report*, Facultad de Filosofía y Letras, Universidad Autónoma de Madrid.

Sánchez León, F.; Nieto Serrano, A.F. (1995). Desarrollo de un etiquetador morfosintáctico para el español. *Revista de la Sociedad Española para el Procesamiento del Lenguaje Natural*, vol. 17 (Septiembre), pp. 14-28.

Sánchez, F.; Porta, J.; Sancho, J.L.; Nieto, A.; Ballester, A.; Fernández, A.; Gómez, J.; Gómez, L.; Raigal, E.; Ruiz, R. (1999). La anotación de los corpus CREA y CORDE. *Revista de la Sociedad Española para el Procesamiento del Lenguaje Natural*, vol. 25 (Septiembre), pp. 175-182.

Sawaf, H.; Schütz, K.; Ney, H. (2000). On the use of grammar based language models for statistical machine translation. In *Proceedings of the Sixth International Workshop on Parsing Technologies (IWPT-2000)*, ACL/SIGPARSE, Trento (Italy), pp. 231-241.

Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*, pp. 44-49, Manchester (England).

Schütze, H.; Singer, Y. (1994). Part-of-speech tagging using a variable memory Markov model. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pp. 181-187.

Smeaton, A.F. (1992). Progress in the application of natural language processing to information retrieval tasks. *The Computer Jorunal*, vol. 35, pp. 268-278.

Stolcke, A.; Omohundro, S. (1993). Hidden Markov model induction by Bayesian model merging. In S.J. Hanson, J.D. Cowan and C. Lee Giles (eds.), *Advances in Neural Information Processing System*, vol. 5, pp. 11-18. Morgan Kaufmann, San Mateo, CA.

Stolcke, A.; Omohundro, S. (1994). Best-first model merging for hidden Markov model induction. *Technical Report* TR-94-003, International Computer Science Institute, University of California at Berkeley.

Stolz, W.S.; Tannenbaum, P.H.; Carstensen, F.V. (1965). A stochastic approach to the grammatical coding of English. *Communications of the ACM*, vol. 8, pp. 399-405.

Strzalkowski, T. (1995). Natural language information retrieval. *Information Processing & Management*, vol. 31, pp. 397-417.

Tapanainen, P.; Voutilainen, A. (1994). Tagging accurately - Don't guess if you know. In *Proceedings of the Fourth Conference on Applied Natural Language Processing*, Stuttgart.

Triviño Rodríguez, J.L. (1995). SEAM - Sistema experto para análisis morfológico. *Master's thesis*, Universidad de Málaga.

Triviño Rodríguez, J.L.; Morales Bueno, R. (2000). A Spanish POS tagger with variable memory. In *Proceedings of the Sixth International Workshop on Parsing Technologies (IWPT-2000)*, ACL/SIGPARSE, Trento (Italy), pp. 254-265.

Vilares Ferro, M. (1992). Efficient incremental parsing for context-free languages. *PhD. Thesis*, Atelier National de Reproduction des Thèses de Grenoble, Grenoble (France).

Vilares Ferro, M.; Valderruten Vidal, A.; Graña Gil, J.; Alonso Pardo, M.A. (1995). Une approche formelle pour la gènération d'analyseurs de langages naturels. In P. Blache (ed.), *Proceedings of TALN'95*, Marseille (France), pp. 246-255.

Vilares Ferro, M.; Graña Gil, J.; Cacheda Seijo, F. (1996a). Verification of morphological analyzers. In *Proceedings of DIALOGUE'96*, Moscow (Russia), pp. 69-72.

Vilares Ferro, M.; Graña Gil, J.; Pan Bermúdez, A. (1996b). Building friendly architectures for tagging. *Revista de la Sociedad Española para el Procesamiento del Lenguaje Natural*, vol. 19 (Septiembre), pp. 127-132.

Vilares Ferro, M.; Graña Gil, J.; Alvariño Alvariño, P. (1997). Finite-state morphology and formal verification. In B.K. Boguraev, R. Garigliano, J.I. Tait and A. Kornai (eds.), *Journal of Natural Language Engineering, special issue on Extended Finite State of Language*, vol.2(4) (December), pp.303-304. Cambridge University Press.

Vilares Ferro, M.; Graña Gil, J.; Araujo, T.; Cabrero, D.; Diz, I. (1998). A tagger environment for Galician. In *Proceedings of Workshop on Language Resources for European Minority Languages*, Granada (Spain).

Viterbi, A.J. (1967). Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Trans. Information Theory*, vol. IT-13 (April), pp. 260-269.

Voutilainen, A.; Heikkilä, J. (1994). An English constraint grammar (ENGCG): a surface-syntactic parser of English. In Fries, Tottie and Schneider (eds.), *Creating and using English language corpora*. Rodopi.

Weischedel, R.; Meteer, M.; Schwartz, R.; Ramshaw, L.; Palmucci, J. (1993). Coping with ambiguity and unknown through probabilistic models. *Computational Linguistics*, vol. 19, pp. 359-382.

Younger, D.H. (1967). Recognition of context-free languages in time n^3. *Information and Control*, vol. 10(2), pp. 189-208.

Zavrel, J.; Daelemans, W. (1999). Recent advances in memory-based part-of-speech tagging. En *Actas del VI Simposio Internacional de Comunicación Social*, Centro de Lingüística Aplicada, Ministerio de Ciencia y Medio Ambiente, Santiago de Cuba, pp. 590-597.

Índice de Materias