Identificación Automática del Idioma en Twitter: Adaptación de Identificadores del Estado del Arte al Contexto Ibérico*

Automatic Language Identification in Twitter: Adapting State-of-the-Art Identifiers to the Iberian Context

Yerai Doval, David Vilares y **Jesús Vilares**
Grupo LYS, Departamento de Computación, Facultade de Informática,
Universidade da Coruña, Campus de A Coruña, 15071 – A Coruña
{yerai.doval, david.vilares, jesus.vilares}@udc.es – `www.grupolys.org`

Resumen: Describimos aquí nuestra participación en el TweetLID. Tras estudiar la problemática de la identificación del idioma, los recursos disponibles y diseñar una solución para la normalización del texto en este tipo de tareas, presentamos dos sistemas a competición: el primero basado en el clasificador `langdetect`, re-entrenado y adaptado para trabajar con texto normalizado; el segundo, una solución basada en la votación de clasificadores individuales igualmente re-entrenados y adaptados. Los resultados son analizados tanto globalmente como por idioma y tipo de tuit.
Palabras clave: Identificación del idioma; tuit; normalización de texto; español; gallego; catalán; euskera; inglés; portugués.

Abstract: We describe here our partipation in TweetLID. After having studied the problem of language identification, the resources available, and designed a text conflation approach for this kind of tasks, we joined the competition with two systems: the first one was based in the guesser `langdetect`, re-trained and adapted in order to work with conflated text; the second one was an approach based on majority vote which used a set of re-trained and adapted classifiers. Results are analized both globally and at language and tweet-type levels.
Keywords: Language identification; tweet; text conflation; Spanish; Galician; Catalan; Basque; English; Portuguese.

1. Introducción y objetivos

La *identificación del idioma* (LID o *language identification*) es un caso particular de *clasificación multiclase* donde, dado un texto de entrada, se pretende averiguar su idioma (Cavnar y Trenkle, 1994). A pesar de los avances en este campo, ciertos dominios de aplicación, como Twitter, siguen constituyendo un desafío (Baldwin y Lui, 2010a; Bergsma et al., 2012) y el TweetLID es un buen ejemplo (San Vicente et al., 2014).

En primer lugar está la naturaleza de las entradas, *tuits* de escasos 140 caracteres cuando la longitud es un factor crítico en LID (Baldwin y Lui, 2010a). Son textos, además, a menudo informales, escritos incorrectamente o empleando *texting* (Oliva et al., 2013), lo que introducirá mucho ruido.

En segundo lugar está lo particular del dominio. En LID se suele asumir que las entradas son *monolingües*, en un solo idioma (Baldwin y Lui, 2010a). Sin embargo, en Twitter no es extraño encontrar tuits *multilingües*, que combinan varios idiomas, p.ej. *"Adiós, brother!"* (EN+ES). Asimismo, la identificación debería ser precisa, pero entradas tan cortas pueden resultar a veces *ambiguas*, pudiendo pertenecer a cualquiera de varios idiomas, p.ej. *"Boa noite"* (GL/PT). Hablaríamos entonces de un problema de *clasificación multi-etiqueta* (Tsoumakas y Katakis, 2007), más complejo y donde una entrada puede estar asociada simultáneamente a varias clases.

El tercer factor son los recursos. Por una parte, algunas de las herramientas de LID disponibles no permiten re-entrenarlas, y no suelen disponer de todos los idiomas requeridos para el TweetLID. Por otra parte, aún pudiendo hacerlo, necesitamos *datasets* para ello, y aunque existen varios públicamente disponibles (Lui y Baldwin, 2011; Lui y Baldwin, 2012; Majliš, 2012) pocos contem-

* Trabajo parcialmente financiado por el Ministerio de Economía y Competitividad y FEDER (TIN2010-18552-C03-02), Ministerio de Educación, Cultura y Deporte (Beca FPU ref. 13/01180) y Xunta de Galicia (CN2012/008).

plan todos los idiomas necesarios, lo que es conveniente para asegurar un comportamiento lo más homogéneo posible. Asimismo, lo ideal habría sido contar con *datasets* de tuits, pero éstos son aún más escasos y no cubren los idiomas requeridos (Bergsma et al., 2012; Lui y Baldwin, 2014).

Para terminar, el propio conjunto de idiomas involucrado en la tarea, aunque pequeño, es en sí problemático. Tomemos como botón de muestra el gallego. En primer lugar, su convivencia con el español supone que no es inusual que un gallego pueda introducir —incluso sin percatarse—, léxico del gallego al emplear el español y viceversa, especialmente en un contexto tan informal como el de un tuit. En segundo lugar, factores como el grado de normativización efectiva de la lengua, sus variantes dialectales, situación geográfica, etc. hacen que podamos encontrarnos, por ejemplo, con dos gallego-parlantes que empleen un gallego más castellanizado uno, y más próximo al portugués, el otro. Finalmente, la proximidad lingüística entre idiomas supone un gran problema, pues compartirán vocabulario, morfemas, etc. dificultando así la clasificación. Estaríamos, pues, ante un problema de clasificación multi-etiqueta en el que no sólo las clases no son disjuntas, sino que además algunas de ellas comparten buen número de elementos.

2. Arquitectura

Aunque inicialmente planteamos dos enfoques, uno multiclase y otro multi-etiqueta, las limitaciones de tiempo sólo nos permitieron poner a punto las soluciones multiclase menos complejas. De este modo, dado un texto de entrada, devolveremos como resultado un único idioma, aquél con el que con mayor seguridad está escrito.

Preparamos dos soluciones. La primera corresponde al caso más sencillo, emplear un único clasificador y, en caso de devolver varios resultados posibles, tomar el mejor. La segunda consiste en una solución basada en *votación* donde, dada una serie de clasificadores, el idioma más veces devuelto entre los resultados parciales de cada uno constituirá el resultado final (Lui y Baldwin, 2014).

3. Recursos empleados

Hemos empleado, en lo posible, recursos de LID ya disponibles, que ahora describimos.

3.1. Clasificadores

Aunque fueron probados otros más, éstos fueron los finalmente escogidos para trabajar.

TextCat. De los más conocidos y sencillos, se trata de una implementación hecha por van Noord (1997) del algoritmo de categorización de Cavnar y Trenkle (1994), en el cual se comparan modelos de lenguaje basados en n-gramas de caracteres de acuerdo a una métrica para la comparación de *rankings* denominada OOP (*Out-Of-Place*).[1]

`langid.py`. Diseñado por Lui y Baldwin (2012), emplea un clasificador bayesiano multinomial combinado con características basadas en n-gramas de *bytes*.

`langdetect`. Implementado por Shuyo (2010), combina un clasificador bayesiano que opera sobre n-gramas de caracteres con mecanismos de normalización.

3.2. Datasets

Hemos preferido no emplear los modelos pre-entrenados en los clasificadores y re-entrenarlos nosotros mismos para así tener más flexibilidad y control sobre nuestros experimentos. Como se dijo en la Sección 1, no fue tarea fácil encontrar corpus apropiados. Finalmente se consideraron dos fuentes.

En primer lugar, el texto del *Tratado de la Constitución Europea* o TCE (Unión Europea, 2004), si bien sólo las Partes I a IV hasta el Artículo IV-448, ya que son las únicas disponibles en sus traducciones a gallego, catalán y euskera.[2] Se trata, pues, de un corpus *paralelo*, con unos 430KB de texto, unas 63.000 palabras, por idioma.

En segundo lugar hemos empleado el *Yali Dataset Long* (Majliš, 2012), comparable aunque no paralelo, con unos 4.5MB de texto, más de 690.000 palabras, por idioma.

El corpus de tuits de desarrollo no se empleó inicialmente ya que, al no disponer por entonces de un segundo corpus de test, habríamos tenido que realizar las pruebas sobre el propio corpus de entrenamiento, con lo que los resultados no habrían sido fiables.

4. Evaluación

4.1. Puesta a punto del sistema

A la hora de decidir la configuración final de nuestro sistema, se realizó una serie de

[1] El código se retocó para poder trabajar con UTF-8 así como para aumentar su eficiencia.

[2] `http://repositori.uji.es/xmlui/`

run	*P*	*R*	*F*
SELECCIÓN CORPUS			
TCE	0.506	0.540	0.477
Yali	0.559	0.610	**0.548**
SELECCIÓN NORMALIZACIÓN			
Yali	0.559	0.610	0.548
Yali$_{no1}$	0.580	0.635	0.576
Yali$_{no2}$	0.589	0.644	0.582
Yali$_{no3}$	0.591	0.639	0.582
Yali$_{no4}$	0.590	0.645	**0.583**
SELECCIÓN IDENTIFICADOR			
TextCat	0.590	0.645	0.583
`langid.py`	0.413	0.252	0.229
`langdetect`	0.620	0.678	**0.622**
votador	0.423	0.266	**0.249**

Tabla 1: Resultados durante el proceso de afinamiento del sistema (en negrita las configuraciones seleccionadas en cada fase).

experimentos de puesta a punto, cuyos resultados de precisión (P), cobertura (R) y medida-F (F) recoge la Tabla 1. En todos ellos se empleó el corpus de desarrollo proporcionado por la organización (San Vicente et al., 2014), previamente limpiado de elementos propios de los tuits tales como enlaces, *hashtags*, *smilies*, etc., que introducirían ruido (Tromp y Pechenizkiy, 2011). Empleamos para ello TWOKENIZE, un *tokenizador*-preprocesador para tuits (Owoputi et al., 2013) e independiente del idioma.

En primer lugar se seleccionó el *dataset* a emplear para entrenar las herramientas. Ambos corpus, TCE y Yali, fueron probados sobre el TextCat, obteniendo los resultados que aparecen en la parte superior de la Tabla 1, mostrándose Yali superior.

Seguidamente, creímos conveniente normalizar el texto para así reducir las fuentes de ruido. De este modo, tomando como base el texto original sin normalizar —previamente limpiado en el caso de los tuits— (*Yali* en la Tabla 1), tanto *datasets* como tuits fueron procesados de forma similar a como ocurre al aplicar *stemming*: en primer lugar probamos a pasar el texto a minúsculas y a eliminar los dígitos (*Yali*$_{no1}$), probando a continuación a eliminar a mayores los signos de puntuación (*Yali*$_{no2}$). Los resultados obtenidos mostraron una clara mejora. Asimismo, los fenómenos de *texting* tales como acortamientos, contracciones, transformaciones, etc. (Oliva et al., 2013) nos preocupaban especialmente, pero dichos fenómenos son mayormente específicos del idioma, y eso es justo lo que queremos obtener. Es por ello que únicamente hemos tratado las repeticiones de caracteres, probando a reducir las secuencias de caracteres iguales a máximo uno (*Yali*$_{no3}$) o dos caracteres (*Yali*$_{no2}$), arrojando esta última mejores resultados. La normalización final pasa, pues, por convertir el texto a minúsculas y eliminar sus dígitos y signos de puntuación. En el caso de los tuits, además, se eliminarán los elementos propios de Twitter (enlaces, menciones, etc.) así como las repeticiones de caracteres de longitud mayor de dos.

run		*P*	*R*	*F*	*pos*
CONSTRAINED	`langdetect`	0.732	0.734	0.639	9/12
	votador	0.610	0.582	0.498	12/12
UNCONSTR.	`langdetect`	0.682	0.688	0.581	5/9
	votador	0.588	0.590	0.571	7/9

Tabla 2: Resultados oficiales

Finalmente, procedimos a seleccionar los sistemas concretos a emplear en la competición: por una parte la herramienta individual que obtuviese mejores resultados y, por otra, nuestra aproximación basada en votación (véase Sección 2). En ambos casos los clasificadores considerados fueron TextCat, `langid.py` y `langdetect` (véase Sección 3.1). La solución basada en votación (*votador*) empleaba las tres, por lo que únicamente restaba saber cuál de las tres era mejor. Empleando la configuración establecida previamente se probaron los tres, obteniendo los resultados recogidos en la parte inferior de la Tabla 1), siendo `langdetect` el ganador.

4.2. Resultados oficiales

Los resultados oficiales enviados a la competición correspondían a *langdetect* y a nuestra solución basada en votación, en ambos casos aplicando los mecanismos de normalización ya descritos, faltando además 26 tuits, que no nos fue posible descargar.

En el caso de la modalidad *constrained*, entrenamos los clasificadores con los tuits del corpus de desarrollo, mientras que en la *unconstrained* empleamos el *dataset* Yali. Los valores obtenidos se muestran en la Tabla 2, donde *langdetect* mejora de nuevo los resultados de la votación, si bien la diferencia es ahora mucho menor. Sin embargo, lo que más nos ha sorprendido es ver que los mejores resultados se obtuvieron para *constrained*, cuando

nos esperábamos que fuesen siempre peores por lo reducido y ruidoso del corpus de entrenamiento. En todo caso, los resultados obtenidos no son tan buenos como los obtenidos en tareas de LID sobre texto "normal".

4.3. Análisis por idioma

A la hora de estudiar en detalle el comportamiento del sistema, decidimos realizar una nueva tanda de experimentos sin tener en cuenta los 26 tuits desaparecidos, sólo los 18.397 *disponibles* —véase Tabla 3. También hemos analizado por separado el caso de los tuits *monolingües*, aquéllos escritos en un único idioma, y los *multilingües*, que contienen más de un idioma, 16.841 y 350 tuits, respectivamente. En estos dos últimos casos no se tuvieron en cuenta ni los tuits ambiguos (*amb*) ni los indefinidos (*und*).

Estos resultados nos confirman la caída de rendimiento en el caso de la solución basada en votación (*votador*), teniendo además un comportamiento más irregular. En el caso de `langdetect`, éste muestra un buen comportamiento en el caso del español (ES) y de los tuits ambiguos (AMB). La precisión es menor para catalán (CA), euskera (EU) e inglés (EN), siendo inesperado lo de estos dos últimos, pues al ser ambos idiomas muy diferentes del resto (todas lenguas romances), se esperaría lo contrario. El peor rendimiento fue para el gallego (GL), debido a la confusión con español y portugués (PT), y para los indefinidos (UND), cuya cobertura fue nula.

En lo que respecta al análisis comparativo de tuits monolingües frente a multilingües, los resultados obtenidos en cuanto a precisión para los multilingües han sido incluso ligeramente mejores que para los monolingües. Al mismo tiempo, la cobertura obtenida ha sido prácticamente la mitad que la de los monolingües, y dado que casi todos los tuits multilingües contenían dos idiomas, eso significa que, si bien para esos tuits sólo hemos podido devolver uno de los dos idiomas que contienen, estamos acertando incluso algo más que para el resto. Esto nos permite ser optimistas de cara a futuras mejoras.

5. Conclusiones y trabajo futuro

Dada nuestra limitación de tiempo, en nuestra participación en el TweetLID hemos optado por aproximaciones sencillas basadas en la adaptación de componentes ya existentes: por un lado, la herramienta `langdetect`, convenientemente re-entrenada y adaptada y, por otro, una solución basada en la votación de varios clasificadores. Lo prometedor de los resultados obtenidos nos permite ser optimistas de cara al futuro.

En el futuro, queremos mejorar nuestros sistemas mediante umbrales de medidas de confianza y la combinación de dichos valores, permitiéndonos abordar el caso de idiomas desconocidos así como la implementación de clasificadores multi-etiqueta (Lui, Lau, y Baldwin, 2014). Asimismo, prestaremos especial atención al caso del gallego.

Bibliografía

Baldwin, T. y M. Lui. 2010a. Language identification: The long and the short of the matter. En *Proc. of HLT'10*, pp. 229–237. ACL.

Bergsma, S., P. McNamee, M. Bagdouri, C. Fink, y T. Wilson. 2012. Language identification for creating language-specific Twitter collections. En *Proc. of LSM'12*, pp. 65–74. ACL.

Cavnar, W. B. y J. M. Trenkle. 1994. N-gram-based text categorization. En *Proc. of SDAIR-94*, pp. 161–175.

Unión Europea. 2004. Tratado por el que se establece una Constitución para Europa. No. C 310 del Diario Oficial de la Unión Europea. 16 Dic.

Lui, M. y T. Baldwin. 2011. Cross-domain feature selection for language identification. En *Proc. of IJCNLP 2011*, pp. 553–561.

Lui, M. y T. Baldwin. 2012. `Langid.py`: An off-the-shelf language identification tool. En *Proc. of ACL 2012 System Demonstrations*, pp. 25–30. ACL. Herramienta: `https://github.com/saffsd/langid.py`

Lui, M. y T. Baldwin. 2014. Accurate language identification of twitter messages. En *Proc. of LASM 2014*, pp. 17–25. ACL.

Lui, M., J. H. Lau, y T. Baldwin. 2014. Automatic detection and language identification of multilingual documents. *Transactions of the ACL*, 2:27–40.

Majliš, M. 2012. Yet another language identifier. En *Proc. of the Student Research Workshop at the EACL'12*, pp. 46–54. ACL. Herramienta: `http://ufal.mff.cuni.cz/tools/yali/`

tuits considerados			*disponibles* (18397)			*monolingües* (16841)			*multilingües* (350)		
run		*leng*	***P***	***R***	***F***	***P***	***R***	***F***	***P***	***R***	***F***
CONSTRAINED	`langdetect`	PT	0.828	0.904	0.864	0.864	0.907	0.885	0.773	0.680	0.723
		ES	0.977	0.815	0.889	0.989	0.823	0.899	0.987	0.505	0.668
		CA	0.564	0.892	0.691	0.654	0.925	0.766	0.674	0.341	0.453
		GL	0.231	0.841	0.362	0.248	0.853	0.384	0.136	0.333	0.194
		EN	0.696	0.822	0.754	0.819	0.923	0.868	0.965	0.311	0.470
		EU	0.561	0.848	0.675	0.694	0.958	0.805	0.979	0.461	0.627
		UND	1.000	0.008	0.017	–	–	–	–	–	–
		AMB	1.000	0.754	0.860	–	–	–	–	–	–
		GLOBAL	0.732	0.736	0.639	0.711	0.898	0.768	0.752	0.438	0.522
	votador	PT	0.507	0.707	0.591	0.528	0.711	0.606	0.239	0.440	0.310
		ES	0.937	0.668	0.780	0.950	0.676	0.790	0.948	0.356	0.518
		CA	0.487	0.732	0.585	0.526	0.757	0.621	0.676	0.294	0.410
		GL	0.091	0.467	0.153	0.106	0.473	0.173	0.056	0.222	0.089
		EN	0.527	0.655	0.584	0.588	0.725	0.650	0.885	0.305	0.454
		EU	0.328	0.713	0.449	0.365	0.791	0.500	0.833	0.441	0.577
		UND	1.000	0.003	0.006	–	–	–	–	–	–
		AMB	1.000	0.719	0.837	–	–	–	–	–	–
		GLOBAL	0.610	0.583	0.498	0.511	0.689	0.556	0.606	0.343	0.393
UNCONSTRAINED	`langdetect`	PT	0.679	0.842	0.752	0.708	0.847	0.771	0.480	0.480	0.480
		ES	0.974	0.746	0.845	0.987	0.757	0.857	0.982	0.362	0.530
		CA	0.567	0.865	0.685	0.641	0.897	0.748	0.778	0.329	0.463
		GL	0.175	0.751	0.284	0.185	0.749	0.297	0.184	0.778	0.298
		EN	0.588	0.832	0.689	0.693	0.910	0.786	0.918	0.441	0.595
		EU	0.472	0.830	0.602	0.570	0.927	0.706	0.962	0.490	0.649
		UND	1.000	0.007	0.015	–	–	–	–	–	–
		AMB	1.000	0.635	0.776	–	–	–	–	–	–
		GLOBAL	0.682	0.689	0.581	0.631	0.848	0.694	0.717	0.480	0.503
	votador	PT	0.680	0.773	0.724	0.700	0.778	0.737	0.474	0.360	0.409
		ES	0.932	0.868	0.899	0.947	0.872	0.908	0.969	0.709	0.819
		CA	0.747	0.713	0.729	0.806	0.746	0.775	0.929	0.153	0.263
		GL	0.321	0.379	0.347	0.327	0.381	0.352	0.300	0.333	0.316
		EN	0.376	0.813	0.515	0.505	0.931	0.655	0.822	0.209	0.333
		EU	0.645	0.620	0.632	0.726	0.704	0.715	0.917	0.324	0.478
		UND	0.000	0.000	0.000	–	–	–	–	–	–
		AMB	1.000	0.565	0.722	–	–	–	–	–	–
		GLOBAL	0.588	0.591	0.571	0.669	0.735	0.690	0.735	0.348	0.436

Tabla 3: Resultados no oficiales para análisis a nivel de idioma(s).

Oliva, J., J. I. Serrano, M. D. del Castillo, y A. Iglesias. 2013. A SMS normalization system integrating multiple grammatical resources. *Natural Language Engineering*, 19(1):121–141.

Owoputi, O., B. O'Connor, C. Dyer, K. Gimpel, N. Schneider, y N. A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. En *Proc. of NAACL-HLT 2013*, pp. 380–390. ACL. Toolkit: `http://www.ark.cs.cmu.edu/TweetNLP/`

San Vicente, I., A. Zubiaga, P. Gamallo, J. R. Pichel, I. Alegría, N. Aranberri, A. Ezeiza, y V. Fresno. 2014. Overview of TweetLID: Tweet Language Identification at SEPLN 2014. En *TweetLID @ SEPLN 2014*.

Shuyo, N. 2010. Language detection library for Java. `http://code.google.com/p/language-detection/`

Sites, D. 2013. CLD2: Compact Language Detector 2. `https://code.google.com/p/cld2/`

Tromp, E. y M. Pechenizkiy. 2011. Graph-based n-gram language identification on short texts. En *Proc. of Benelearn 2011*, pp. 27–34.

Tsoumakas, G. y I. Katakis. 2007. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13.

van Noord, G. 1997. TextCat. `http://odur.let.rug.nl/\~vannoord/TextCat/`