Tesis de doctorado

Uso de técnicas de recomendación en sistemas dispersos.

Autor: Diego Fernández Iglesias

Directores: Fidel Cacheda Seijo
Víctor Manuel Carneiro Díaz

Departamento: Tecnologías de la Información y las Comunicaciones
Año: 2014

A mis padres y a mi hermano

Agradecimientos

Muchos son los que me han acompañado en este largo camino para llegar hasta donde hoy estoy y que merecen ser nombrados.

Gracias a mis directores de tesis, Fidel y Víctor, por su cercanía y comprensión, sus consejos y opiniones.

Gracias a todas las personas del HPC-Lab del ISTI-CNR, por haberme dado la oportunidad de vivir una experiencia inolvidable.

Gracias especialmente a mis compañeros de laboratorio, por tantas risas, por tantos momentos compartidos, por hacer del día a día algo especial, por su inestimable ayuda, por su amistad.

Gracias a todos y cada uno de mis amigos, a todo esa gente que ha ido apareciendo en mi vida para quedarse y que me ha empujado hasta aquí. Sin ellos esto no habría sido posible.

Gracias, por supuesto, a mi familia por todo lo que ha hecho por mí para que este momento se haga realidad. Gracias por estar ahí siempre.

Gracias también a ti, por escucharme, apoyarme y animarme.

Resumen

En esta tesis nos centramos en el problema que padecen los sistemas de recomendación basados en filtrado colaborativo para recomendar cuando pocos usuarios han valorado los mismos productos: el problema de la dispersión de datos.

El análisis de *datasets* realizado en el dominio de recomendación de películas evidencia las limitaciones de las medidas de similitud tradicionales de los algoritmos basados en vecinos y el elevado número de usuarios con pocas puntuaciones. Para paliar estas limitaciones, proponemos una medida de similitud menos estricta, que mejora los resultados en predicción en situaciones dispersas, y ahondamos en técnicas de expansión de perfil, que aumentan el número de puntuaciones para dichos usuarios, aliviando la dispersión, lo que se refleja en los resultados de las recomendaciones.

Para profundizar en nuestro estudio, recurrimos al dominio de la recomendación de consultas, por ser especialmente disperso, como demostramos en el análisis efectuado de un *query log* público. Las técnicas colaborativas se ajustan bien a este nuevo ámbito, a pesar de la gran dispersión de datos. Ésta disminuye al aplicar técnicas de preprocesamiento y al eliminar las sesiones con pocas consultas, mejorando con ello significativamente los resultados de los algoritmos empleados.

Resumo

Nesta tese centrámonos no problema que padecen os sistemas de recomendación baseados en filtrado colaborativo para recomendar cando poucos usuarios valoraron os mesmos produtos: o problema da dipersión de datos.

A análise de *datasets* realizada no dominio de recomendación de películas evidencia as limitacións das medidas de similitude tradicionais dos algoritmos baseados en veciños e o elevado número de usuarios con poucas puntuacións. Para paliar estas limitacións, propoñemos unha medida de similitude menos estrita, que mellora os resultados en predición en situacións dispersas, e afondamos en técnicas de expansión de perfil, que aumentan os número de puntuacións para ditos usuarios, aliviando a dispersión, o que se reflicte nos resultados das recomendacións.

Para profundar no noso estudo, recorremos ó dominio da recomendación de consultas, por ser especialmente disperso, como demostramos na análise efectuada dun *query log* público. As técnicas colaborativas axústanse ben a este novo ámbito, a pesar da grande dispersión de datos. Esta diminúe ó aplicar técnicas de preprocesamento e ó eliminar as sesións con poucas consultas, mellorando con eso significativamente os resultados dos algoritmos empregados.

Abstract

In this thesis we focus on the difficulties that recommender systems based on collaborative filtering present in order to recommend when few users have rated the same products: the sparsity problem.

We accomplish the analysis of different datasets in the movie recommendation domain, which make clear the limitations of the traditional similarity measures that neighbor-based algorithms usually employ and the big amount of users with few ratings. In order to alleviate these limitations, we propose a less strict similarity measure, which increases the accuracy in the prediction task in sparse situations, and we go through the profile expansion techniques in depth, which increase the number of ratings for those users, alleviating the sparsity problem, what is reflected significantly in the recommendation results.

To deepen our subject, we turn to the query recommendation domain, since it is especially sparse, as we demonstrate with the analysis of a public query log. The collaborative techniques fit well in this new scope, despite the important sparsity of the data. This sparsity diminishes by applying preprocessing techniques and ignoring sessions with few queries and, then, the results improve significantly.

Índice general

Índice general I

Índice de figuras VII

Índice de tablas XI

I Contextualización 1

1. Introducción 3

1.1. Motivación 3

1.2. Objetivos 6

1.3. Estructura de la tesis 7

2. Sistemas recomendadores 9

2.1. Concepto de sistema de recomendación 9

2.2. Tareas de un sistema de recomendación 11

2.3. Proceso de recomendación 13

2.4. Técnicas de recomendación 15

2.5. Principales problemas 18

2.6. Evaluación 22

2.6.1. Métricas de precisión 22

2.6.1.1. Métricas de precisión en la predicción 22

2.6.1.2. Métricas de precisión en la clasificación 24

2.6.1.3. Métricas de precisión en la ordenación 25

2.6.2. Otras métricas 26

II Dispersión en sistemas de recomendación de películas 27

3. Filtrado colaborativo 29
3.1. Formalización del problema . . . 29
3.2. Tipos de técnicas de filtrado colaborativo . . . 30
3.3. Principales algoritmos para predicción . . . 32
3.3.1. Algoritmos basados en memoria . . . 32
3.3.2. Similarity Fusion . . . 36
3.3.3. Algoritmos basados en modelo . . . 36
3.3.3.1. Basados en regresión lineal . . . 36
3.3.3.2. Slope One . . . 37
3.3.3.3. Modelos de factorización de matrices . . . 38
3.3.3.4. Cluster Based Smoothing . . . 39
3.3.3.5. Basado en tendencias . . . 40
3.3.4. Técnicas colaborativas híbridas . . . 42
3.3.4.1. Personality Diagnosis . . . 42
3.3.4.2. Modelo integrado: basado en vecinos – SVD++ . . . 42
3.3.5. Otros algoritmos . . . 42
3.4. Tarea de recomendación . . . 43
3.5. Experimentos . . . 43
3.5.1. *Datasets* . . . 43
3.5.2. Metodología . . . 45
3.6. Resultados . . . 47
3.6.1. Datos dispersos en predicción . . . 47
3.6.2. Algoritmos basados en modelo frente a algoritmos basados en memoria . . . 50
3.7. Conclusiones . . . 56

4. Mejora de la tarea de predicción en sistemas dispersos 59
4.1. Introducción . . . 59
4.2. Cold-start . . . 62
4.3. Análisis de dos *datasets* . . . 64
4.3.1. Limitaciones de las medidas de similitud tradicionales . . . 64
4.3.2. Productos populares . . . 68
4.3.3. La importancia del problema de cold-start . . . 70
4.4. Mejorando los algoritmos kNN . . . 70
4.5. Experimentos . . . 75

4.5.1. Diseño de experimentos 75
4.5.2. La influencia de los parámetros β y N 76
4.5.3. Comparación con las medidas basadas en correlaciones 78
4.5.4. Impacto de *Inverse User Frequency* 79
4.6. Conclusiones y trabajos futuros 80

5. Mejora de la tarea de recomendación en sistemas dispersos 81
5.1. Introducción 81
5.2. Expansión de perfil 83
5.2.1. Expansión global basada en productos 85
5.2.2. Expansión local basada en productos 86
5.2.3. Expansión global basada en usuarios 86
5.2.4. Expansión local basada en usuarios 87
5.3. Expansión de perfil de order superior 88
5.3.1. Serie 88
5.3.2. Paralela 89
5.4. Experimentos 90
5.4.1. Diseño de experimentos 90
5.4.2. Técnicas de expansión de perfil 91
5.4.3. Expansión de perfil de orden superior 93
5.4.3.1. Serie 94
5.4.3.2. Paralela 97
5.5. Conclusiones 99

6. Expansión de perfil basada en contenido 101
6.1. Introducción 101
6.2. Expansión de perfil basada en contenido 103
6.2.1. Selección de características 103
6.2.2. Representación de las características 106
6.2.3. Medidas de similitud 106
6.2.3.1. Distancia coseno 107
6.2.3.2. Similitud ponderada 107
6.2.4. Selección de productos 107
6.2.4.1. Información colaborativa 108
6.2.5. Estimación de la puntuación 109
6.2.5.1. Propuesta colaborativa 109
6.2.5.2. Propuestas basadas en contenido 109

6.3. Metodología y experimentos . . . 111
6.3.1. Evaluación y métricas . . . 112
6.3.1.1. Expansión . . . 112
6.3.1.2. Recomendación . . . 113
6.4. Resultados . . . 113
6.4.1. Expansión . . . 113
6.4.2. Recomendación . . . 116
6.5. Conclusiones . . . 119

III Un dominio diferente: la recomendación de consultas 121

7. Introducción a la recomendación de consultas 123
7.1. *Query logs* . . . 124
7.2. Proceso de búsqueda . . . 125
7.3. Consultas . . . 127
7.4. Segmentación de sesiones . . . 128
7.4.1. Basadas en tiempo . . . 129
7.4.2. Basadas en el léxico . . . 130
7.4.3. Basadas en información semántica . . . 131
7.4.4. Detección de misiones . . . 132
7.5. Sugerencia de consultas . . . 132

8. Análisis de un *query log* 135
8.1. *Query log* . . . 135
8.2. Estudio de sesiones lógicas . . . 136
8.3. Clics . . . 138
8.3.1. Clics según la posición en las sesiones lógicas . . . 138
8.3.2. Clics según el ranking de los documentos . . . 144
8.3.3. Clics múltiples . . . 148
8.4. Aplicación de técnicas de preprocesado . . . 150
8.5. Sesiones lógicas satisfactorias . . . 154
8.6. Conclusiones . . . 157

9. Search Shortcuts: El problema de los atajos de búsqueda 159
9.1. Introducción . . . 159
9.2. Search Shortcuts . . . 161
9.2.1. Descripción del problema y motivación . . . 161

9.2.2. Modelo teórico de Search Shortcuts 163
9.3. Algoritmos de Search Shortcuts 165
9.4. Experimentos 168
9.4.1. Configuración de los experimentos 168
9.4.2. Resultados 171
9.5. Conclusiones 178

10. Conclusiones y trabajos futuros **181**
10.1. Conclusiones 181
10.2. Contribuciones 184
10.2.1. Publicaciones y estancias de investigación 186
10.3. Trabajos futuros 188

Bibliografía **191**

ÍNDICE GENERAL

Índice de figuras

3.1. Puntuaciones por usuario en los *datasets* de MovieLens y Netflix. 44
3.2. Porcentaje de cada puntuación, en ambos *datasets*. 45
3.3. Evolución de la precisión (en MAE) en el *dataset* de MovieLens según la densidad de la matriz. 48
3.4. Evolución de GIM y GPIM según la densidad de la matriz, en el *dataset* de MovieLens. 48
3.5. Evolución de RMSE y de *coverage* según la densidad de la matriz, en el *dataset* de MovieLens. 49
3.6. Evolución del *coverage* y la precisión según la densidad de la matriz, en el *dataset* de Netflix. 50
3.7. Precisión y *coverage* de los distintos algoritmos bajo condiciones de dispersión, con una estrategia Given-N, en el *dataset* de MovieLens. 51
3.8. Precisión y *coverage* de los distintos algoritmos bajo condiciones de dispersión, con una estrategia Given-N, en el *dataset* de Netflix. 52
3.9. Resultados de las métricas de clasificación para los *datasets* de MovieLens y Netflix, con un subconjunto de entrenamiento del 50 %. 54
3.10. Evolución de la precisión y el *coverage* según el número de *clusters*. . . . 55
3.11. Precisión según el número de vecinos, con un subconjunto de entrenamiento del 80 %. 55
3.12. Precisión y *coverage* según el umbral de correlación, con un subconjunto de entrenamiento del 80 %. 56
3.13. Efecto de la normalización (Z-Score) sobre la precisión de los algoritmos basados en usuario. 56

4.1. Histogramas para la correlación de Pearson. 65
4.2. Número de productos valorados por ambos usuarios, según la correlación de Pearson. 66
4.3. Histogramas para las correlaciones de Pearson ponderadas. 66

ÍNDICE DE FIGURAS

4.4. Número de pares usuario-usuario, según el porcentaje de productos que han valorado ambos (con respecto al número total de productos valorados). 67
4.5. Número total de productos valorados por ambos usuarios, según el porcentaje de productos que ambos han valorado. 67
4.6. Usuarios que han valorado ambos productos según la distancia coseno. . 68
4.7. Distribución de puntuaciones para cada cuartil del número de puntuaciones, en el *dataset* de MovieLens 10M. 69
4.8. Distribución de puntuaciones para cada cuartil del número de puntuaciones, en el *dataset* de Netflix. 69
4.9. Distribución de las puntuaciones según el porcentaje de usuarios. 71
4.10. Número de productos valorados por ambos usuarios, usando la similitud de los productos valorados en común. 73
4.11. Número de productos valorados por ambos usuarios, usando la similitud de los productos basada en niveles. 74
4.12. Resultados de MAE para distintos valores de β con el 10 % de las puntuaciones como subconjunto de entrenamiento, con la similitud basada en niveles y la similitud basada en distancia. 77
4.13. Resultados de RMSE para distintos valores de β con el 10 % de las puntuaciones como subconjunto de entrenamiento, con la similitud basada en niveles y la similitud basada en distancia. 77
4.14. Comparación de distintas medidas de similitud, con el 10 % de las puntuaciones como conjunto de entrenamiento. 78
4.15. Comparación de distintas medidas de similitud, con el 90 % de las puntuaciones como subconjunto de entrenamiento. 79
4.16. Impacto en MAE del factor *iuf* con el 90 % de puntuaciones como subconjunto de entrenamiento. 80

5.1. Comportamiento general de los algoritmos kNN, que usan el perfil del usuario para seleccionar los vecinos. 83
5.2. Expansión de perfil. 84
5.3. Expansión de perfil global basada en productos. 85
5.4. Expansión de perfil local basada en productos. 86
5.5. Expansión de perfil local basada en usuarios. 87
5.6. Alternativa en serie. 89
5.7. Alternativa en paralelo. 90
5.8. Evolución de MAP para los *baselines* con $l = 5$. 92

5.9. Evolución de P@5 y MAP según N, usando combinaciones en serie con la técnica global basada en productos y $l = 2$. 95
5.10. Evolución de P@5 y MAP según N, usando combinaciones en serie con las técnicas locales basadas en usuarios y $l = 2$. 96
5.11. Evolución de P@5 y MAP según N, usando el mismo algoritmo dos veces y $l = 2$. 97

6.1. Curvas P-R en expansión para diferentes características. 114
6.2. MAE en expansión para diferentes técnicas de expansión de perfil, según el número de productos añadidos al perfil, l. 116
6.3. Curvas P-R en recomendación para distintos algoritmos con su número óptimo de productos añadidos al perfil, l. 118

8.1. Número de sesiones lógicas por tamaño, para un umbral de 30 minutos. 138
8.2. Porcentaje de número de clics (0, 1, 2, 3, 4, 5 y >5) según posición de la consulta en la sesión lógica, para sesiones de longitud 2, 3, 4 y 5 (umbral de 30 minutos). 142
8.3. Longitud de la sesión lógica según la posición de las consultas que tienen por lo menos un clic, para un umbral de 30 minutos. 143
8.4. Para un umbral de 30 minutos, distribución de la posición de las consultas con clic según el tamaño de la sesión lógica. 143
8.5. Porcentaje de clics en función del ranking del documento teniendo en cuenta sólo las consultas con al menos un clic. 145
8.6. Distribución de los clics por tamaño de la sesión lógica, según la posición del documento. 145
8.7. Distribución de los clics por posición de la consulta en la sesión lógica, según la posición del documento. 146
8.8. Longitud de la sesión lógica según el ranking de los documentos con clic. 147
8.9. Número de clics según el ranking de los documentos, para las consultas iniciales y finales de cada sesión lógica. 148
8.10. Porcentaje de clics (respecto al total de consultas con clic) en los siguientes resultados teniendo en cuenta que el usuario hizo clic en el primer, segundo, tercer o cuarto resultados, según corresponda. 149
8.11. Número de consultas que se repiten el mismo número de veces aplicando *stopwords* y *term stemming*. 151
8.12. Frecuencia de una consulta según su posición en la sesión lógica aplicando *stopwords* y *term stemming*. 152

8.13. Frecuencia de una consulta según la longitud de la sesión lógica aplicando *stopwords* y *term stemming*. 153
8.14. Frecuencia de una consulta según su papel en la sesión lógica, para sesiones lógicas de por lo menos dos consultas aplicando *stopwords* y *term stemming*. 154
8.15. Para cada consulta inicial, porcentaje de éxito frente a número de sesiones lógicas que empiezan por esa consulta. 155
8.16. Éxito según el número de sesiones lógicas iniciadas por una consulta. . . 156
8.17. Número de consultas iniciales según las veces que se repiten. 156

9.1. Porcentaje de sesiones lógicas satisfactorias en el *query log* de AOL, según la popularidad de la primera consulta. 162
9.2. Efecto de las diferentes técnicas de preprocesado sobre los *query logs* de AOL y MSN. 171
9.3. Resultados de MAE y *coverage* en los *datasets* de AOL y MSN. 172
9.4. Resultados, en el *dataset* de AOL, de las métricas de precisión, ROC y *half-life utility*. 174
9.5. Resultados, en el *dataset* de MSN, de las métricas de precisión, ROC y *half-life utility*. 175
9.6. Correlación entre la función de similitud de *Search Shortcuts* y la precisión, dependiendo de la función f elegida, tanto en el *dataset* de AOL como el de MSN. 176
9.7. Porcentaje de mejora, en precisión del algoritmo, cuando las sesiones con menos de 3 consultas son ignoradas, en los *datasets* de AOL y MSN. . . 178
9.8. Efecto de distintas técnicas de preprocesado en la precisión, en los *datasets* de AOL y MSN. 179

Índice de tablas

5.1. P@5 y MAP para los *baselines*. 93
5.2. P@5 y MAP para las combinaciones en serie, con $l = 2$. 94
5.3. P@5 y MAP para las alternativas en paralelo. 98

6.1. Precisión en expansión para diferentes técnicas de expansión de perfil, según el número de productos añadidos al perfil, l. 115
6.2. P@5 en recomendación para diferentes técnicas de expansión de perfil, según el número de productos añadidos al perfil, l, con $\alpha = 1$. 117
6.3. P@5 en recomendación para diferentes técnicas de expansión de perfil, según el número de productos añadidos al perfil, l, con $\alpha = 0.75$. 117
6.4. P@5 en recomendación para diferentes técnicas de expansión de perfil, según el número de productos añadidos al perfil, l, con $\alpha = 0$. 118

8.1. Extracto de la información acerca de las consultas. 136
8.2. Extracto de la información acerca de los clics. 136
8.3. Número de sesiones lógicas según la longitud y umbral empleados. . . . 137
8.4. Porcentaje de sesiones lógicas exitosas según tamaño y umbral. 139
8.5. Media de clics por consulta, según el umbral y el tamaño de la sesión lógica. 140
8.6. Número de consultas según tratamiento aplicado. 151

9.1. Resultados de varios algoritmos según la métrica de similitud de *Search Shortcuts*, en el *dataset* de AOL. 177
9.2. Resultados de varios algoritmos según la métrica de *Search Shortcuts*, en el *dataset* de MSN. 177

ÍNDICE DE TABLAS

Parte I

Contextualización

Capítulo 1

Introducción

Los sistemas recomendadores han experimentado un gran auge en las últimas décadas. Una de las razones principales es que se han convertido en una herramienta de suma importancia para el comercio electrónico. Estos sistemas facilitan las tomas de decisión a los usuarios, al ser capaces de filtrar, en función de los gustos de los propios usuarios, la gran cantidad de productos de los que un sitio web puede disponer. Por tanto, permiten obtener grandes beneficios. Además, debido al interés suscitado tanto en el campo de la investigación como de la industria, son objeto de numerosas líneas de investigación, muchas de ellas encaminadas a solucionar alguno de los problemas a los que estos sistemas se tienen que enfrentar. Esta tesis se centra en uno de ellos: el problema de la dispersión de los datos.

En la siguiente sección indicaremos cuál es la motivación que ha llevado al desarrollo de esta tesis. Después, en la Sección 1.2, enumeraremos los objetivos que nos planteamos. Finalmente, en la última sección del capítulo mostraremos cuál es la estructura de esta tesis.

1.1. Motivación

El nacimiento de Internet representa un gran hito en la historia de la humanidad. Desde mediados de los años 90 surge como un medio de comunicación global, de desarrollo de aplicaciones comerciales y de difusión de información. Las causas son variadas: la concepción del ordenador personal como electrodoméstico, el aumento de las capacidades de los sistemas de telecomunicación, el desarrollo de herramientas más fáciles de usar (como los navegadores web), la universalización (debido a la conexión de todo tipo de dispositivos, a las nuevas tecnologías y a los impulsos de los gobiernos), etc. Mucha gente no concibe el día a día sin tener acceso a la Red de Redes.

1. INTRODUCCIÓN

Muchas de las actividades llevadas a cabo diariamente a través de Internet están relacionadas con el comercio electrónico o con la búsqueda de información. El comercio electrónico consiste en la compra y venta de productos y servicios a través de la red. Existe una cantidad ingente de información de acceso público en la web y que evoluciona día a día, y esto ha provocado que este tipo de comercio haya crecido también increíblemente. Aparte de las causas comentadas anteriormente acerca de la gran explosión de Internet, la personalización supone una de las razones más importantes del gran auge del comercio electrónico. El alcance global de Internet y la gran riqueza de opciones y productos disponibles permiten que esto sea así. De hecho, la personalización alcanza gran importancia a la hora de fidelizar a los usuarios de un sitio web. Los usuarios, a cambio de proporcionar cierta información acerca de sus gustos, reciben productos y sugerencias individualizadas y ajustadas a sus preferencias. Los sistemas recomendadores son las herramientas capaces de proporcionar estas sugerencias personalizadas. Son útiles tanto para el proveedor del servicio, pues le permite incrementar sus beneficios al aumentar la fidelización de sus usuarios, como para el usuario, que reduce el tiempo invertido en la selección de los productos y ve simplificado sus tomas de decisión, al disponer de una herramienta capaz de filtrar gran cantidad de información adecuadamente y de proporcionarle sugerencias. Aunque se hayan empleado con éxito en comercio electrónico, su uso no está limitado a este dominio.

Por su parte, la búsqueda de información, es una actividad habitual dada la gran cantidad de información disponible. Sin embargo, ésta no está estructurada. Por eso son necesarias herramientas para manejar, recuperar y filtrar toda esta información. Los motores de búsqueda representan una de estas herramientas. Su principal finalidad es ayudar al usuario a satisfacer sus necesidades de información, expresadas por medio de consultas, de un modo eficiente. Las consultas enviadas por los usuarios forman parte de la información que las grandes compañías de búsqueda web almacenan acerca de la interacción de sus usuarios. Estas consultas aportan datos sobre ellos y sus quehaceres diarios, preferencias, etc. Sin embargo, no siempre están bien formuladas, por lo que los resultados obtenidos no tienen por qué ser los adecuados. La propia ambigüedad inherente al lenguaje humano y la falta de rigor que con frecuencia se tiene a la hora de escribir una consulta hacen compleja esta tarea. A esto hay que añadir que, aún formulando bien las consultas, interpretar correctamente los resultados devueltos por el motor de búsqueda también se antoja complicado. Por estas razones los motores de búsqueda disponen de herramientas para facilitar todo este proceso a los usuarios. Una de estas herramientas es la sugerencia de consultas. De esta manera se le puede ahorrar tiempo al usuario, evitando que formule un excesivo número de consultas para alcanzar

su objetivo. Existen técnicas de sugerencia de consultas. Los sistemas recomendadores son una alternativa que también puede proporcionar sugerencias personalizadas en este dominio.

Aunque durante las dos últimas décadas se han desarrollado numerosas nuevas técnicas, muchos son todavía los retos abiertos y las mejoras necesarias [Adomavicius y Tuzhilin, 2005], ya que el estudio de los sistemas recomendadores representa una disciplina aún joven. A pesar de ello, dada su gran aplicabilidad, suscita mucha atención tanto de la industria como de la comunidad investigadora. En concreto, las técnicas colaborativas [Goldberg et al., 1992] son las más utilizadas. Estas técnicas emplean únicamente como fuente de información las puntuaciones que los usuarios hacen sobre los productos. Surgieron partiendo de que con frecuencia nos basamos en las opiniones de la gente que nos rodea para tomar decisiones cotidianas como qué canción escuchar, qué libro leer, etc. Cuando nuestras preferencias sobre un determinado tipo de productos son cercanas a las de otra persona, solemos hacer caso a las sugerencias de dicha persona. Es por ello que este tipo de técnicas principalmente buscan similitudes entre usuarios y/o entre productos en base sólo a las preferencias (en forma de puntuaciones) y realizan sugerencias a partir de dichas similitudes.

Sin embargo, toda esta gran variedad de técnicas de recomendación sufre en mayor o menor medida distintos problemas. En particular, las técnicas colaborativas tienen dificultades para realizar recomendaciones cuando la densidad de datos (el número de puntuaciones) es escasa, o, lo que es lo mismo, cuando los datos son muy dispersos. El problema se debe a que las puntuaciones son la única fuente de información que emplean estas técnicas. Estas situaciones de dispersión de datos producen un empeoramiento en el comportamiento de los algoritmos. Este empeoramiento se agrava cuando el desarrollo del algoritmo no se ajusta a los datos subyacentes.

El dominio de los motores de búsqueda y, más concretamente, de la recomendación de consultas es particularmente disperso. Millones de usuarios realizan billones de consultas diariamente, muchas de las cuales se repiten muy pocas veces. Elaborar sugerencias adecuadas para gran parte de estas consultas resulta una tarea sumamente complicada.

El problema de la dispersión de datos supone el punto de unión de todo el trabajo realizado en esta tesis.

1.2. Objetivos

El estudio de la dispersión de los datos en los sistemas de recomendación se puede abordar desde diversos puntos de vista. Nuestro principal objetivo es comprender el comportamiento tanto de las técnicas de recomendación como de los usuarios en diferentes dominios para así poder mejorar las sugerencias que dichas técnicas pueden ofrecer en situaciones de gran dispersión de datos. Este objetivo se puede desglosar, a su vez, en varios subobjetivos, que describiremos a continuación:

- Estudiar el estado del arte de los algoritmos de filtrado colaborativo, poniendo especial atención en cómo influye la dispersión de los datos en ellos. La finalidad de los algoritmos de filtrado colaborativo, como la de cualquier otra técnica de recomendación, es sugerir productos útiles a los usuarios. Su peculiaridad es que se basan únicamente en las puntuaciones para realizar las recomendaciones. Debido a su gran popularidad, es interesante conocer las diferentes técnicas existentes, compararlas y analizar su comportamiento ante diferentes condiciones de dispersión de los datos.

- Comprender cómo se comportan los usuarios en distintos dominios. El análisis tanto de los datos pertenecientes a algunos de los *datasets* más populares en el dominio de la recomendación de películas, como de los registros de la interacción de los usuarios con un motor de búsqueda permitirán comprender mejor el comportamiento de los usuarios de dichos dominios, obtener conclusiones más fiables de la evaluación de los algoritmos de recomendación empleados, entender sus limitaciones y aportar soluciones a sus problemas en base a las lecciones aprendidas.

- Abordar la dispersión de los datos para los algoritmos de filtrado colaborativo centrándonos en añadir información a aquellos usuarios que tengan pocas puntuaciones. Los sistemas de recomendación no siempre se encuentran en las condiciones más favorables. El comportamiento de determinados algoritmos, cuando tienen poca información acerca del usuario, dista mucho de ser el deseado. En caso de las técnicas colaborativas, la dispersión de los datos se acrecentará, lógicamente, cuantos más usuarios haya con pocas puntuaciones. El principal problema es que, empleando este tipo de técnicas, en un momento u otro un usuario tendrá que pasar por esta situación. Por tanto, en estos casos, bien el usuario se involucra y proporciona más información (ya sea o no colaborativa) para mejorar las sugerencias que recibe, o bien se desarrollan técnicas para paliar el perjuicio de estas

situaciones. Nuestro objetivo es optar por esta segunda opción

- Aplicar técnicas de filtrado colaborativo a la recomendación de consultas. El dominio de la recomendación de consultas es especialmente interesante porque permitirá profundizar en el estudio de las técnicas colaborativas en un entorno particularmente disperso. Sin embargo, al tratarse de un domino muy diferente a los dominios tradicionales en los que se suelen usar estas técnicas, su aplicación en este contexto trae consigo una serie de retos que deberemos afrontar, como, por ejemplo, encontrar una correspondencia entre los conceptos de filtrado colaborativo y los del nuevo dominio, inferir las puntuaciones a partir de los datos disponibles, etc. La idea subyacente es que nuevos usuarios, habiendo formulado el mínimo de consultas posible, puedan beneficiarse de los procesos de búsqueda de otros usuarios que hayan tenido la misma necesidad de información, reduciendo así el tiempo y el número de consultas necesarios para alcanzar los resultados deseados.

1.3. Estructura de la tesis

Esta tesis está dividida en tres partes. En la primera parte realizamos una contextualización, aportando una primera aproximación a los distintos problemas que abordaremos. En el Capítulo 2 hacemos una introducción a los sistemas recomendadores. Explicamos la noción de sistema recomendador y los conceptos básicos que en él intervienen. Además, incidimos en las distintas tareas que puede abordar, tanto desde el punto de vista del usuario como del proveedor del servicio. También explicamos en qué consiste el proceso de recomendación propiamente dicho, presentamos una clasificación general de las distintas técnicas existentes, así como los principales problemas que tienen que abordar, entre los que se encuentra la dispersión de los datos, el tema principal de esta tesis. Asimismo, comentamos las distintas alternativas para llevar a cabo la evaluación de estos sistemas.

En la segunda parte de esta tesis analizamos cómo afecta la dispersión de los datos a los algoritmos de recomendación en un dominio tan estudiado como es el de la recomendación de películas, para posteriormente proponer diversas alternativas para hacer frente a este problema. En el Capítulo 3 presentamos las principales técnicas colaborativas existentes en el estado del arte y las comparamos. En concreto, ponemos especial atención en cómo son capaces de extraer información conforme la densidad de los datos de los que disponen es menor, es decir, en su evolución conforme la dispersión es más marcada.

En la literatura, como comentaremos más adelante, hay dos tareas que acometen los sistemas recomendadores desde el punto de vista del usuario que copan la gran mayoría de los trabajos: la tarea de predicción y la tarea de recomendación. Así, en el Capítulo 4 se abordará precisamente la primera de ellas, que consiste en calcular la puntuación que un usuario daría a un producto. Nos centramos en los algoritmos de filtrado colaborativo y, particularmente, en los basados en vecinos. En especial, a partir de una análisis previo de los datos, estudiaremos cómo se comportan las medidas de similitud tradicionales usadas en este tipo de algoritmos en situaciones de dispersión y, para mitigar sus carencias, propondremos una medida alternativa.

La otra tarea comentada se plantea en los Capítulos 5 y 6. En el primero de ellos se presentan las técnicas de expansión de perfil y cómo éstas pueden ser empleadas para paliar las limitaciones que tienen los algoritmos de filtrado colaborativo cuando se enfrentan al problema del nuevo usuario. En concreto, se propone la combinación de varias técnicas de expansión simples para seleccionar los productos con los que expandir el perfil de un usuario y obtener así mejores resultados en la recomendación. En el Capítulo 6, por su parte, se aborda una aproximación basada en contenido para seleccionar los productos para la expansión.

La tercera parte de la tesis se centra en el dominio de la recomendación de consultas y en cómo aplicar los algoritmos de filtrado colaborativo a dicho dominio. El Capítulo 7 introduce los conceptos básicos de la búsqueda web (como consulta, sesión, clic, segmentación...) y comenta los trabajos del estado del arte más relevantes.

Tal y como sucede para la recomendación de películas, en este dominio también es importante hacer un análisis previo de los datos antes de desarrollar cualquier técnica. Por ello, en el Capítulo 8 realizamos un estudio de un *query log* público, prestando especial atención a aspectos relacionados con la dispersión de los datos como puede ser el número de consultas únicas en función de cada segmentación empleada, el número de sesiones lógicas de cada longitud o las consecuencias de aplicar las diversas técnicas de preprocesamiento propuestas. También hacemos un estudio de la distribución de los clics, al ser una medida de relevancia parcial importante en este dominio.

Tras haber analizado el dominio, en el Capítulo 9 especificamos cómo se pueden aplicar algoritmos de filtrado colaborativo para la recomendación de consultas, explicando cómo abordamos diferentes retos que surgen en este proceso, y evaluamos los resultados obtenidos.

Para finalizar, en el Capítulo 10 exponemos las conclusiones alcanzadas y los futuros trabajos que pueden surgir a partir de esta tesis.

Capítulo 2

Sistemas recomendadores

Los sistemas recomendadores son sistemas capaces de predecir si un determinado producto será o no del agrado de un usuario. Hoy en día resultan de gran utilidad a numerosos sitios web para sugerir a sus usuarios productos de acuerdo a sus preferencias. De este modo, se filtra el catálogo de productos disponible, a menudo muy extenso, ajustándose a los gustos del usuario. Para ello el sistema se puede basar en las valoraciones del usuario (tanto explícitas como implícitas), en atributos de usuarios y productos y en valoraciones de otros usuarios. Existen gran variedad de técnicas, que presentan diversas ventajas e inconvenientes en función de los problemas a los que se enfrentan.

A lo largo de este capítulo haremos una introducción a los sistemas recomendadores. En primer lugar abordaremos su origen y características generales. A continuación, en la Sección 2.2 explicaremos cuáles son las tareas que puede acometer uno de estos sistemas, tanto desde el punto de vista del usuario, como del proveedor del servicio. Después, abordaremos el proceso de recomendación y los diferentes conceptos que en él intervienen. En la Sección 2.4 mostraremos una visión general de las diferentes técnicas de recomendación existentes, para acto seguido pasar a enumerar los principales problemas a los que tienen que hacer frente. La última sección del capítulo la dedicaremos a explicar cómo se evalúan estos sistemas.

2.1. Concepto de sistema de recomendación

A menudo, nos basamos en consejos dados por nuestro grupo de amigos, compañeros de trabajo, etc., para tomar decisiones del día a día. Qué película ver, qué libro comprar, etc., son preguntas que nos hacemos con frecuencia. Si una persona coincide en sus gustos sobre determinados productos con un grupo de gente, es probable que dicha per-

sona considere relevantes otros productos que desconoce, pero que fueron recomendados por dicho grupo de gente. Los primeros sistemas recomendadores surgieron imitando esta idea, usando las preferencias de una comunidad de usuarios para hacer una recomendación a un usuario concreto. De hecho, estos sistemas son básicamente técnicas y herramientas software que sugieren productos a usuarios [Resnick y Varian, 1997], siendo un producto o elemento un término genérico empleado para designar aquello que el sistema recomienda al usuario. Las sugerencias están relacionadas precisamente con diferentes tomas de decisión, tales como qué película ver, qué libro comprar, etc.

A medida que los usuarios disponen de más y más productos entre los que escoger, se hace más palpable la necesidad de una herramienta capaz de proporcionar sugerencias acorde a sus intereses. Esto provoca que los sitios web, hoy en día en continuo crecimiento, hagan uso de sistemas recomendadores para conducir al usuario hacia aquello que más se acerca a sus necesidades. De hecho, un sitio web puede disponer de un catálogo de productos enorme, el cual difícilmente será conocido en su totalidad por un usuario. Mediante este tipo de sistemas el usuario puede recibir recomendaciones que se ajusten a sus gustos y necesidades. De otro modo la gran variedad de productos entre los que escoger puede convertirse en excesiva información, produciendo incluso que el usuario pueda dejar de usar un determinado servicio ante las dificultades a la hora de la toma de decisión. Así, este tipo de sistemas, al restringir el número de productos, facilitan las tomas de decisión a los usuarios. Un claro ejemplo de los beneficios que pueden aportar estos sistemas es *Amazon.com* [Linden et al., 2003], que incluye la recomendación de libros, dispositivos electrónicos y una amplia variedad de productos. Otro ejemplo es Netflix [Bennett y Lanning, 2007] en el que el 60 % de las películas se seleccionan de acuerdo a recomendaciones personalizadas y que se ha convertido en el servicio de alquiler de películas *online* más grande del mundo.

El estudio de los sistemas recomendadores surgió como una nueva área de investigación a mediados de los años 90 [Goldberg et al., 1992; Hill et al., 1995; Resnick et al., 1994; Shardanand, 1994] con la aparición de los primeros trabajos sobre filtrado colaborativo (ver Sección 3.1), lo cual hace de ella una disciplina joven en comparación con otras como las bases de datos o los motores de búsqueda, por poner dos ejemplos. Durante las dos últimas décadas se ha trabajado mucho en el desarrollo de nuevas técnicas. A pesar de todo el trabajo realizado hasta el momento en este campo, muchas son todavía las mejoras necesarias y los campos de acción abiertos [Adomavicius y Tuzhilin, 2005]. La investigación está evolucionando en muy diversas direcciones, nuevos temas aparecen, etc. Por ejemplo, ya que hoy en día es habitual disponer de dispositivos electrónicos con acceso a Internet en cualquier lugar y momento, uno de los retos es que

los sistemas sean más proactivos [Bulander et al., 2005; Puerta Melguizo et al., 2007], es decir, que no recomienden únicamente cuando se lo solicita el usuario, sino que sean capaces de predecir cuándo recomendar, detectando, para ello, las peticiones implícitas de los usuarios.

Debido a la gran aplicabilidad de las soluciones y a la necesidad de ellas para ayudar a los usuarios a tratar con la inmensa cantidad de información disponible hoy en día, el interés en esta área es importante, tanto desde el punto de vista de la investigación como de la industria.

2.2. Tareas de un sistema de recomendación

Como ya se ha dicho, la cantidad de sistemas de recomendación utilizados en sitios web crece a pasos agigantados. Youtube, IMDb o Amazom.com tienen a dichos sistemas como parte importante del servicio ofrecido. Las tareas que realizan estos sistemas se pueden ver desde dos puntos de vista bien diferenciados: el del proveedor del servicio y el del usuario.

Desde el punto de vista del proveedor del servicio, existen diversas tareas encomendadas a un sistema de recomendación que pueden ser las razones por las que una determinada compañía opte por usar este tipo de sistemas. La primera, y quizá más importante de las razones, es el aumento de los beneficios. Mediante este tipo de sistemas los productos ofrecidos se pueden ajustar más a los gustos y preferencias de los usuarios, por lo que tras varias recomendaciones, es previsible que el usuario confíe más en el sistema, y, de este modo, acepte las recomendaciones y consuma los productos sugeridos. Además, los sitios web disponen de una amplia variedad de productos y también ofrecen su servicio a unos usuarios con una amplia variedad de gustos diferentes, debido al alcance global de Internet. Mientras que sin sistemas recomendadores el sitio web rara vez se arriesgaría a promover un producto poco popular para ajustarse a unos gustos muy concretos de un usuario, usando sistemas recomendadores esto es posible. De este modo se ajustan las sugerencias (ya sean productos populares o no) a los gustos de cada uno de los usuarios. Si las recomendaciones son precisas y la interfaz es *amigable*, la satisfacción del usuario se puede ver incrementada, con lo que probablemente intentará alimentar al sistema con más información para así obtener mejores recomendaciones. El usuario mejora su experiencia, y el sistema adquiere más información para poder realizar mejores recomendaciones. Cuanto más conozca el sistema del usuario, más podrá adaptar su interfaz a sus gustos, incrementando generalmente así el grado de fidelización con el sitio web. Pero no sólo eso: esta información permite

conocer mejor las necesidades de los usuarios, sus preferencias y ajustar los productos a ellas. Hay que tener en cuenta que desde el punto de vista del proveedor del servicio un recomendador en general deberá seleccionar aquel producto que, si es seleccionado por el usuario, maximice su valor tanto para el usuario como para el proveedor en un momento dado. Además, puede que la mejor recomendación para el usuario no siempre sea la mejor para el proveedor.

Desde el punto de vista del usuario, existen otras finalidades u objetivos de los sistemas de recomendación. En [Herlocker et al., 2004] el autor identifica un conjunto de hasta diez tareas que estos sistemas pueden acometer independientemente del dominio en el que se desplieguen:

- *Annotation in context.* El sistema predice la preferencia de un usuario sobre un producto determinado en función del contexto. Según dicha anotación, el usuario será capaz de decidir si el producto en cuestión es acorde o no a sus gustos.
- *Find good items.* Consiste en sugerir al usuario un conjunto de productos de acuerdo a sus gustos. Dichas sugerencias pueden ir acompañadas de una puntuación para determinar hasta qué punto el producto puede satisfacer al usuario.
- *Find all good items.* En determinadas ocasiones no es suficiente con sugerir algunos productos del gusto del usuario, sino que es necesario recuperar todos los productos que pueden ser de su interés, aún a riesgo de que el usuario tenga que invertir gran cantidad de tiempo en analizarlos. Es el caso de la búsqueda de jurisprudencia para un determinado caso judicial, por ejemplo.
- *Recommend sequence.* Los productos, en lugar de ser recomendados individualmente, son sugeridos siguiendo una secuencia, de tal manera que sea la propia secuencia como un todo del agrado del usuario.
- *Just browsing.* El usuario, en determinadas ocasiones, puede no desear adquirir ningún producto de un determinado sitio web, sino que únicamente recorrerlo con el solo objetivo de buscar artículos de su interés.
- *Find credible recommender.* Con frecuencia los usuarios no se fían de las recomendaciones proporcionadas por los sistemas de recomendación. Esta tarea consiste en ofrecer al usuario la posibilidad de probar el comportamiento del propio sistema para aumentar así su fiabilidad.
- *Improve profile.* Los sistemas recomendadores, en su gran mayoría, proporcionan recomendaciones a los usuarios, pero dichas recomendaciones a menudo no serían

posibles si el usuario no indicase de alguna manera sus gustos interaccionando con el sistema. Esta tarea consiste, precisamente, en obtener dicha información por parte de los usuarios. Proporcionando puntuaciones, interaccionando con el sistema, etc. el usuario mejorará su perfil y, de esta manera, a cambio esperará obtener mejores recomendaciones.

- *Express self.* Determinados usuarios pueden llegar a disfrutar del mero hecho de proporcionar su opinión acerca de los productos. Los sistemas recomendadores pueden facilitar a este tipo de usuarios la posibilidad de expresarse y aportar su parecer.

- *Help others.* Otra tarea que puede acometer un sistema es el de permitir a un usuario mostrar información a una comunidad de usuarios, no para mejorar sus propias recomendaciones, sino para facilitar a otros usuarios su toma de decisión.

- *Influence others.* A menudo aparecen usuarios que pretenden influenciar con su opinión a otros usuarios, o incluso sesgar las recomendaciones para que el sistema bien favorezcan o perjudiquen a unos productos en particular. Los sistemas de recomendación han de prevenir este tipo de situaciones.

Como ya se indicaba en [Herlocker et al., 2004], las tareas de *annotation in context* y de *find good items* son, con diferencia, las que más se han estudiado en la literatura. A partir de ahora nos referiremos a ellas, para simplificar, como tarea de predicción y tarea de recomendación, respectivamente.

2.3. Proceso de recomendación

En el proceso de recomendación el primer paso es encontrar los productos que recomendar. Éstos pueden ser obtenidos por el propio sistema (tarea de recomendación) o proporcionados por el usuario (tarea de predicción). Evidentemente, para mostrar los productos recomendados, la información acerca del producto ha de ser adaptada a una representación *amigable* y útil para el usuario. De acuerdo al tipo de sistema existente, la predicción se podrá calcular en función de los gustos del usuario y de información acerca del producto. Según la interfaz del sistema y la interacción del usuario con el mismo, se mostrarán más o menos productos, con mayor o menor cantidad de información acerca de ellos.

Los sistemas recomendadores, como ya se ha mencionado, usan información de muy diferente índole a la hora de realizar una recomendación. En cuanto a la información

acerca de los productos, ésta variará en función de la complejidad de los mismos y de la técnica de recomendación empleada. Algo semejante sucede con los usuarios, de los que el sistema podrá mantener diferente información estructurada de acuerdo a la técnica utilizada. Además, el sistema ha de capturar y almacenar también datos sobre la interacción de los usuarios y sus preferencias para poder realizar las recomendaciones. Dichas preferencias se pueden obtener tanto implícita como explícitamente. Las preferencias implícitas son fáciles de obtener (clics, tiempo transcurrido en una determinada página, etc.), aunque, por su naturaleza, únicamente representan relevancia parcial positiva sobre un objeto, a la vez que presentan mucho ruido. Sin embargo, suelen molestar menos al usuario, al no implicar una acción específica por su parte. Por otro lado, cuando el usuario proporciona datos al sistema siendo consciente de que está dando dicha información (la valoración de un producto según sus gustos e intereses, por ejemplo) dichas preferencias se dice que son explícitas. En este caso el usuario ha de estar dispuesto a puntuar los productos, por lo que, aunque aportan más información que las preferencias implícitas a cambio de mejores recomendaciones, el hecho de puntuar puede suponer una tarea demasiado tediosa para el usuario. En general, las preferencias explícitas se clasifican en:

- Escalas de puntuaciones. Pueden ser tanto numéricas como cualitativas, indicando cada puntuación hasta qué punto un determinado producto gusta o no gusta a un usuario. Las escalas numéricas, a su vez, pueden ser tanto continuas como discretas, siendo las discretas las más habituales. Qué escala emplear no es una decisión trivial, ya que esta decisión puede acarrear un gran impacto en el sistema [Sparling y Sen, 2011].
- Puntuaciones binarias. El usuario dispone de una puntuación positiva y de una puntuación negativa, que puede otorgar a cada producto.
- Puntuaciones unarias. El usuario únicamente indica que considera relevante un producto, pero no el grado de relevancia.

No existe una norma global que indique que un tipo de preferencias es mejor o peor, dependerá del dominio en concreto en el que se emplee [Cosley et al., 2003]. Cuanto mayor sea la escala de las puntuaciones explícitas, más información proveerá al sistema, pero más compleja será la interfaz y más costosa será la tarea de la puntuación para el usuario. A su vez existen varios focos de variabilidad intrínsecos al propio hecho de puntuar [Amatriain et al., 2009; Hill et al., 1995]. En primer lugar existen diferentes modos de puntuar [Herlocker et al., 1999; Resnick et al., 1994]. Desde un usuario que

puntúa siempre de un modo muy positivo, por encima de las medias de los productos, hasta otro que puntúa muy negativamente. Por otro lado, hay que tener en cuenta también que los propios gustos o nuestra forma de puntuar pueden variar con el tiempo [Xiang y Yang, 2009], y también con el contexto [Ellenberg, 2008] (el momento del día, el estado de ánimo, la compañía, etc.).

Aparte de lo comentado, el propio usuario puede proporcionar al sistema *feedback* sobre la recomendación proporcionada, que podrá ser tenido en cuenta en futuras recomendaciones.

2.4. Técnicas de recomendación

Son múltiples los sistemas de recomendación existentes. Todos ellos guían a los usuarios hacia productos que pueden ser de su interés. Desde el dominio al que están dirigidos, hasta la fuente de conocimiento usada o el algoritmo de recomendación utilizado, existen diversos aspectos en los que pueden diferir dos sistemas de recomendación. A continuación enumeramos los tipos de sistemas de recomendación más importantes [Burke, 2002; Ricci et al., 2011]:

- Filtrado colaborativo. El sistema genera las recomendaciones empleando sólo como fuente de información las puntuaciones que los usuarios han hecho sobre los productos. Es decir, en un sistema de filtrado colaborativo puro no se tiene en cuenta el contenido de los productos; lo único importante es poder identificarlos. Principalmente estos sistemas buscan similitudes entre los usuarios en función de sus puntuaciones y las recomendaciones se crean a partir de dichas similitudes. Se trata de la técnica de recomendación más empleada y, probablemente, la más madura [Resnick et al., 1994]. Al ser el tipo de técnicas en el que nos centraremos a lo largo de esta tesis, nos detendremos más adelante en ellas (ver Capítulo 3).

- Basado en contenido. Estos recomendadores trabajan con perfiles de usuarios obtenidos a partir de las características de contenido presentes en los productos que los usuarios puntúan. Los perfiles de usuario podrán ser de diferentes clases en función del método de aprendizaje empleado (árboles de decisión, redes neuronales, representaciones vectoriales...). El sistema busca similitudes entre el perfil del usuario y el de los productos, para así obtener recomendaciones que se ajusten a las características que son del agrado del usuario. Este tipo de técnicas están limitadas por las características asociadas a los productos considerados (*análisis de contenido limitado*). Una consecuencia de ello es que dos productos que tengan los

mismos valores para todas sus características serán considerados de igual relevancia para un usuario, aunque realmente no lo sean. Por ejemplo, dos textos escritos con exactamente las mismas palabras pueden ser de calidad muy diferente.

No todas las características del producto formarán parte de su representación de contenido, por lo que es necesario, para cada dominio, escoger el conjunto de características adecuadas y cuáles de entre ellas son realmente importantes. Aunque las técnicas basadas en contenido se comportan mejor cuando se emplean únicamente características útiles, es necesario tener en cuenta que tampoco se pueden reducir excesivamente las características a utilizar, ya que cada usuario tiene su propia percepción de la importancia de cada característica.

Se han empleado diferentes aproximaciones a este tipo de técnicas como un modelo de espacio vectorial [Billsus y Pazzani, 1998], clasificadores bayesianos [Kohavi et al., 1997] o clasificadores SVM [Sebastiani, 2002]. Gran parte de los modelos presentan representaciones textuales de los productos. A veces se trata de datos estructurados. Sin embargo, en muchas ocasiones no sabemos los valores que pueden tomar las características, con los problemas que eso conlleva (polisemia y sinonimia, por ejemplo). Esto es debido a que los perfiles basados en texto tradicionales no son capaces de capturar la semántica de los intereses del usuario porque se basan en operaciones de *string matching*. Muchos de estos modelos presentan el problema de su *falta de inteligencia*. Es decir, conforme los perfiles de usuarios y productos crecen, el funcionamiento de este tipo de modelos es peor, puesto que no pueden inferir nuevos términos de un mismo campo semántico que puedan ser útiles para el usuario en cuestión; es decir, a partir de las palabras *Michael Jordan* , el modelo no es capaz de obtener *baloncesto*, por ejemplo. Así, a menudo necesitan adquirir conocimiento por medio de ontologías o enciclopedias.

- Basado en información demográfica. En este tipo de sistemas se emplea la información demográfica del usuario para realizar recomendaciones [Rich, 1979; Wang et al., 2012]. Por tanto, el sistema considerará distintas preferencias en función de edad, sexo, etc.. Cada usuario tendrá asignado una clase demográfica en función de su perfil. El sistema usará después las características comunes de los usuarios de dichas clases para justificar las recomendaciones.

- Basado en conocimiento. Estos sistemas basan su recomendación en un conocimiento exhaustivo del propio dominio y en el manejo de los requisitos explícitos de los usuarios [Burke, 2000]. Se estudia cómo determinadas características se ajustan a los gustos y preferencias del usuario. No sufren el problema de *cold-start* (ver

Sección 2.5), ya que los requisitos del usuario se obtienen directamente durante la sesión de recomendación. Sin embargo, presentan el problema de cómo representar formalmente el conocimiento del dominio y de cómo adquirir dicho conocimiento, ya sea el conocimiento del propio catálogo de productos, el conocimiento funcional acerca de cómo hacer corresponder las necesidades de los usuarios a los productos que satisfacen dichas necesidades, o el conocimiento acerca de las preferencias de los usuarios. Existen estos tipos principalmente:

  - Basados en casos [Bridge et al., 2006; Lorenzi y Ricci, 2005]. Para realizar la recomendación primero se calcula la similitud entre las características que el usuario necesita y aquéllas que poseen los productos disponibles.
  - Basados en restricciones [Felfernig y Burke, 2008]. Explota una base de conocimiento para asociar las necesidades de los usuarios con un conjunto de reglas que permiten filtrar aquéllos productos que pueden ser del interés del usuario.

- Basado en utilidad [Burke, 2000; Felfernig et al., 2006]. Se centran en la idea de que cada usuario decide la preferencia por un determinado producto ponderando las ventajas y desventajas de cada característica acorde a la frecuencia con la que ésta resulta beneficiosa o perjudicial para sus intereses. El usuario podrá incorporar todas las características de los productos que crea oportuno, y ha de ponderar cada característica, de tal modo que se pueda construir una función de utilidad. De este modo se pueden incorporar factores que contribuyen al valor del producto (como los términos de la garantía o las opciones de envío, por ejemplo), que no son específicamente características. En ciertas ocasiones esto supone un problema con productos que posean un gran número de ellas. Después se calculará el valor de dicha función para cada producto.

  En general, se presupone que el usuario tendrá voluntad de interaccionar con el sistema, de participar en el proceso de selección del producto y de realizar cierto esfuerzo a la hora de procesar la información [Spiekermann y Paraschiv, 2002].

  A veces se clasifican como un tipo de sistema basado en conocimiento.

- Basado en comunidades. Este tipo de sistemas hace las recomendaciones basándose en las preferencias de otros usuarios del círculo de amigos, es decir, en relaciones de confianza. Esto se debe a que la gente tiende a confiar más en las recomendaciones de los amigos que en recomendaciones de otras personas desconocidas, aunque de gustos similares [Sinha et al., 2001]. Debido a la gran proliferación de

las redes sociales, estos sistemas están adquiriendo un gran apogeo. Es habitual referirse a este tipo de sistemas como sistemas recomendadores sociales [Golbeck, 2006], puesto que se basan en la relaciones sociales de un determinado usuario para obtener las preferencias de sus amigos.

- Híbridos. Estos sistemas se basan en la combinación de al menos dos de los otros tipos de técnicas, con el objetivo de mejorar los resultados del empleo de cada una de ellas individualmente. Aunque nada prohíbe combinar dos técnicas del mismo tipo, las combinaciones más prometedoras son aquéllas que emplean soluciones de tipos diferentes [Burke, 2007]. Existen diferentes alternativas a la hora de realizar dichas combinaciones. En [Burke, 2002] se definen siete tipos:

  - *Weighted*. Las puntuaciones de las distintas técnicas se combinan para producir una única recomendación.
  - *Switching*. El sistema emplea una técnica u otra en función de un determinado criterio.
  - *Mixed*. Se presentan al mismo tiempo recomendaciones proporcionadas por diferentes técnicas.
  - *Feature combination*. Las características de diferentes fuentes de información procedentes de las distintas técnicas, se consideran de forma conjunta a modo de un único algoritmo.
  - *Cascade*. Una técnica refina las combinaciones dada por otra, de tal manera que se sigue un proceso secuencial en el que la recomendación de una técnica es usada por la siguiente como datos de entrada.
  - *Feature augmentation*. En este caso la salida de una determinada técnica se emplea como entrada para la siguiente aunque siendo considerada únicamente como una característica más, de tal modo que productos no presentes en la recomendación de la primera técnica pueden ser recomendados por la segunda (lo cual no sucede en el caso del método *cascade*).
  - *Meta-level*. Este tipo de combinación implica que un modelo aprendido por un recomendador se usa como entrada para otro.

2.5. Principales problemas

Los sistemas recomendadores, como ya se ha comentado, están experimentando un continuo avance. Durante los últimos años se han abierto múltiples líneas de investi-

gación. Muchas de ellas están destinadas a solucionar algunos de los problemas que afectan a este tipo de sistemas. A continuación, destacaremos algunos de ellos :

- Escasez de puntuaciones o *sparsity* [Huang et al., 2004b]. Es un problema que aparece principalmente en las técnicas colaborativas y que consiste en la dificultad para realizar recomendaciones cuando pocos usuarios han puntuado los mismos productos. Esto es debido a que este tipo de técnicas se basan principalmente en el solape en las puntuaciones de diferentes usuarios. Por tanto, aparecerá este problema cuando el número de productos sea muy elevado respecto al número de usuarios. Los sistemas basados en conocimiento y en utilidad no presentan este problema.

- *Cold-start* [Schein et al., 2002] o *ramp-up* [Konstan et al., 1998]. Es un problema que podemos dividir, a su vez, en diversos tipos. El primero de ellos, conocido como de *nuevo usuario*, sucede cuando el sistema es capaz de generar con dificultad una recomendación para un usuario con pocas puntuaciones. Esto tiene lugar principalmente cuando el sistema se basa únicamente en puntuaciones.

  De modo similar, resulta complicado recomendar productos que son nuevos en el sistema (problema del *nuevo producto*). Este problema es claro cuando el sistema sólo emplea puntuaciones.

  Existirá también otro caso, el problema del *nuevo sistema*, que se refiere a los casos en donde un determinado sistema se ha desplegado recientemente, por lo que presentará pocas puntuaciones de los usuarios.

  Este problema también se denomina problema del *early rater*, debido a que el primer usuario que puntúa un determinado producto no obtiene ningún beneficio de ello, puesto que esta puntuación no le permitirá al sistema obtener usuarios más similares.

  Los sistema demográficos, basados en conocimiento y en utilidad no presentan este problema, ya que no se necesitan puntuaciones para poder calcular una recomendación. Los sistemas basados en contenido, sin embargo, también presentan un problema de *start-up*, puesto que necesitarán puntuaciones suficientes para poder construir un clasificador adecuado. Dependerán, por supuesto, de la información de contenido disponible acerca de sus productos, ya que los productos en sí no aportan ningún tipo de información para este tipo de sistemas.

  Tanto el problema de *sparsity* como *cold-start* serán los problemas en los que nos centraremos en esta tesis, y que serán abordados con detenimiento en los

próximos capítulos

- Sinonimia. Con frecuencia productos muy similares, o incluso productos iguales, presentan nombres muy diferentes, por lo que habitualmente son tratados por los sistemas recomendadores como productos distintos. En filtrado colaborativo, por ejemplo, se ha intentado solucionar el problema por medio de expansión de términos o de diccionarios o tesauros, lo cual decrementa el rendimiento del sistema.

- *Oveja gris* [Claypool et al., 1999]. La oveja gris es un problema que presentan tanto las técnicas colaborativas como los sistemas demográficos. Un usuario nuevo que no tiene usuarios parecidos (en un sistema de filtrado colaborativo), o no tiene un estereotipo de usuarios al que parecerse (caso del sistema demográfico), no podrá recibir recomendaciones adecuadas, ya que el sistema carecerá de información para hacerlo.

- *Outside the box*. Es un problema que típicamente tienen los sistemas basados en contenido, por ejemplo. Consiste en que el sistema no será capaz de recomendar productos que no tengan ninguna característica en común con aquéllos que el usuario ha puntuado, aunque, en realidad, sí que estén relacionados. Por ejemplo, en un sistema de recomendación de películas que sólo se base en el género, si todos los usuarios a los que le gustan las comedias, les gustan también los romances, a un nuevo usuario que haya puntuado comedias el sistema basado en contenido nunca le recomendará romances. Esto no sucedería con un sistema basado en filtrado colaborativo o un sistema basado en información demográfica. Los sistemas basados en conocimiento podrían recomendar este tipo de productos sólo si la base de conocimiento presenta relaciones entre ellos.

- Ataques y manipulaciones de preferencias o *shilling attacks* [Lam y Riedl, 2004; Mobasher et al., 2007]. Determinados usuarios pueden querer favorecer o perjudicar unos productos frente a otros, introduciendo información inadecuada en el sistema, como, por ejemplo, puntuaciones malintencionadas en caso de los sistemas basados en filtrado colaborativo. Para este caso se han presentado en la literatura diversas soluciones a cómo afrontar este problema [Chirita et al., 2005; Sandvig et al., 2007].

- Privacidad. Hoy en día existe una sensibilidad especial con respecto a la privacidad de los datos. Los sistemas recomendadores explotan los datos de los usuarios para realizar las recomendaciones. Sin embargo, muchos usuarios son reacios a

proporcionar información acerca de sus gustos o preferencias para que el sistema no adquiera demasiada información acerca de ellos. Este hecho afecta particularmente a los sistemas que emplean información demográfica, por el tipo de información a tratar. Para solucionar este problema en los últimos años se han propuesto técnicas para garantizar la privacidad de los usuarios [Aïmeur et al., 2008; Canny, 2002a; Lam et al., 2006; Ramakrishnan et al., 2001; Shokri et al., 2009].

- Escalabilidad. Son múltiples los trabajos realizados que presentan nuevas técnicas de recomendación. Muchos de ellos no abordan cómo integrar dichos sistemas en el mundo real o cómo tratar con una gran cantidad de datos sin que ello repercuta en la eficiencia de la solución. De hecho, a menudo complejos algoritmos que son adecuados en su presentación teórica no se pueden usar en la práctica debido a su ineficiencia. Es por ello que en la actualidad abordar este problema ha llamado el interés de muchos investigadores debido a su importancia de cara al desarrollo de soluciones realmente aplicables [Papagelis et al., 2005b; Sarwar et al., 2002, 2005].

- Diversidad. A menudo los usuarios usan sus recomendaciones como una herramienta para descubrir distintas alternativas, diferentes productos. Sin embargo, los sistemas recomendadores con frecuencia se restringen a recomendar productos sumamente parecidos a los ya valorados por el usuario, que, incluso, puede que no le aporten nada. Este problema de ceñirse demasiado a lo ya valorado se conoce con el nombre de sobreespecialización. Es deseable que el sistema proporcione recomendaciones de diversa índole, aún a riesgo de decrementar la precisión de las mismas. De esta manera es más probable que el usuario encuentre un producto adecuado entre los propuestos por el sistema [Smyth y McClave, 2001; Zhang, 2009]. Los sistemas basados en contenido, por ejemplo, sufren claramente el problema de la sobreespecialización, puesto que el sistema no es capaz de recomendar un producto que se aleje del perfil del usuario. Además, en ciertos casos este tipo de sistemas también filtran productos muy similares a los ya valorados por el usuario. Uno de los modos de abordar este problema es mediante el uso de recomendaciones inesperadas y fortuitas, a la par que útiles, que el usuario quizá no podría descubrir de otro modo (*serendipity*). También, a través de recomendaciones novedosas que ayudan al usuario a encontrar productos interesantes que quizá podría haber descubierto por sí mismo (*novelty*) [Herlocker et al., 2004].

- Efecto *portfolio*. Es un problema muy relacionado con el de falta de diversidad.

Consiste en no recomendar productos muy similares a aquéllos que ha valorado el usuario, aun cuando sí que son relevantes para él. Un caso muy habitual es un sistema recomendador de noticias, en donde el sistema puede no sugerir un artículo por ser demasiado semejante a alguno ya leído por el usuario, aunque éste aporte información nueva.

- Transparencia. La explicación de una recomendación juega un papel muy importante para que el usuario la considere aceptable. Que el sistema sea transparente puede ayudar al usuario a tomar una mejor decisión. De hecho, explicar el porqué aumenta el compromiso del usuario con el sistema, puesto que comprende la razón de las recomendaciones [Herlocker et al., 2000].

2.6. Evaluación

Aún a costa de tener que abordar ciertos problemas, un sistema recomendador ha de proporcionar *buenas recomendaciones*, lo que se traduce, en la práctica, en que ha de satisfacer a sus usuarios. En la literatura podemos encontrar diversos métodos para medir cuán bueno es uno de estos sistemas, y así poder compararlo con otras alternativas. De hecho, los sistemas recomendadores se pueden evaluar desde muy diferentes puntos de vista. La tendencia general es la de medir la precisión de los algoritmos. Las métricas de precisión, sin embargo, no son capaces de evaluar nuevos aspectos considerados en los sistemas recomendadores, como pueden ser la novedad o la diversidad. Por tanto, no son suficientes para medir su rendimiento, ya que la precisión es sólo uno de los puntos a tener en cuenta en la evaluación de un sistema [McNee et al., 2006]. Una recomendación precisa no implica necesariamente que sea útil. En [Herlocker et al., 2004] se presenta un estudio más detallado de las decisiones más importantes a la hora de realizar la evaluación.

2.6.1. Métricas de precisión

A continuación, presentamos algunas de las métricas más populares que podemos encontrar en la literatura.

2.6.1.1. Métricas de precisión en la predicción

Calculan cuán cerca es la puntuación predicha por el sistema de la puntuación proporcionada por el usuario.

- *Mean Absolute Error* (MAE). Esta métrica mide la media de la diferencia en valor absoluto entre el valor predicho por el algoritmo y la puntuación real dada por el usuario. Se calcula según la Ecuación 2.1:

$$MAE = \frac{\sum^{N} |r_{ui} - \hat{r}_{ui}|}{N} \tag{2.1}$$

en donde N es el número total de puntuaciones predichas.

A pesar de sus limitaciones a la hora de evaluar la tarea de recomendación, su facilidad de cálculo y sus propiedades estadísticas [Herlocker et al., 2004] la han convertido en una de las métricas más populares para evaluar sistemas recomendadores.

- *Root Mean Squared Error* (RMSE). Esta métrica de precisión en la predicción, a diferencia de MAE, da más valor a errores grandes, puesto que pueden ser los que más influyan en la decisión del usuario. El modo en que se calcula se muestra en la Ecuación 2.2,

$$RMSE = \sqrt{\frac{\sum^{N} (r_{ui} - \hat{r}_{ui})^2}{N}} \tag{2.2}$$

en donde, de nuevo, N es el número total de puntuaciones predichas.

Se trata también de una métrica popular, en parte gracias a que fue la métrica propuesta para el concurso organizado por Netflix [Bennett y Lanning, 2007].

- *Good Items MAE* (GIM) y *Good Predicted Items MAE* (GPIM). En [Cacheda et al., 2011a] propusimos dos métricas que, en lugar de tener en consideración todos los productos, se centran únicamente en aquéllos que son relevantes. Para ello se define un umbral, θ, que indica la puntuación a partir de la que un producto se considera relevante. De esta manera, N_θ será el número de puntuaciones relevantes y $\hat{N}_\theta$ será el número de puntuaciones predichas como relevantes. Ambas métricas, definidas respectivamente en las Ecuaciones 2.3 y 2.4, complementan las métricas tradicionales de predicción, en el sentido de que se centran en aquellos productos que realmente tienen importancia para los usuarios y aportan una información importante acerca de si un algoritmo tiene tendencia a predecir por encima o por debajo de la puntuación real de los productos.

$$GIM = \frac{\sum^{N_\theta} |r_{ui} - \hat{r}_{ui}|}{N_\theta} \tag{2.3}$$

$$GPIM = \frac{\sum^{\hat{N}_\theta} |r_{ui} - \hat{r}_{ui}|}{\hat{N}_\theta} \tag{2.4}$$

2.6.1.2. Métricas de precisión en la clasificación

Se centran en el concepto de relevancia y en la capacidad que tiene el sistema de detectar qué producto es relevante y cuál no para un determinado usuario. Para definir estas métricas se tienen en cuenta las siguientes medidas: verdaderos positivos (VP), productos relevantes que han sido recomendados por el algoritmo; verdaderos negativos (VN), productos no relevantes que no han sido recomendados por el algoritmo; falsos positivos (FP), productos no relevantes que han sido recomendados por el algoritmo; y falsos negativos (FN), productos relevantes que no han sido recomendados por el algoritmo. A partir de ellas, se definen las siguientes métricas:

- *Precision* (P). Es el ratio entre los productos que el algoritmo considera relevantes y que lo son, entre el número total de productos que para el algoritmo son relevantes. Se calcula según la siguiente ecuación:

  $$P = \frac{VP}{VP + FP} \tag{2.5}$$

  Para tareas de recomendación a menudo suele ser interesante no tanto medir la precisión para todos los productos recomendados, sino únicamente para los n productos que son mostrados al usuario (P@n).

- *Recall* (R). Representa el número de productos relevantes recomendados frente al número total de productos relevantes. La fórmula de esta métrica es la siguiente:

  $$R = \frac{VP}{VP + FN} \tag{2.6}$$

  Igual que sucede con la Precisión, en la tarea de recomendación se usa el Recall en los n primeros productos de la lista (R@n).

- Curvas de Precision-Recall o curvas P-R [van Rijsbergen, 1979]. Son curvas que permiten estudiar la evolución de Precision y Recall conjuntamente, en lugar de estudiar directamente los valores de ambas métricas.

- *Mean Average Precision* (MAP) [Manning et al., 2008]. Es una métrica asociada a una curva P-R que aproxima el área bajo dicha curva.

- F_1 o *F-measure*.

  Aunque sería deseable que tanto Precision como Recall alcanzasen valores altos, lo cierto es que son inversamente proporcionales. Esta métrica combina las dos anteriores haciendo una media ponderada de ambas. La fórmula se muestra en la Ecuación 2.7.

$$F_1 = \frac{2PR}{P+R} \tag{2.7}$$

- Curvas ROC. Las curvas *Receiver Operating Characteristic* [Herlocker et al., 2004] representan gráficamente la capacidad que tiene un determinado algoritmo de distinguir productos relevantes de los que no lo son. Así, el eje de abscisas de la gráfica representa la tasa de FP, especificidad ($\frac{FP}{FP+VN}$) y el de ordenadas la tasa de VP o sensibilidad ($\frac{VP}{VP+FN}$). Es necesario establecer un umbral de relevancia de tal modo que los productos cuya puntuación sea mayor que el umbral se consideren relevantes, y no relevantes en otro caso.

  Por otro lado existe también una métrica que mide la sensibilidad de las curvas ROC, que en la práctica es el área bajo la curva (conocida por sus siglas en inglés *AUC* o *Area Under the Curve*). Cuanto mayor es esta área, mejor será el rendimiento del sistema.

2.6.1.3. Métricas de precisión en la ordenación

Estas métricas se encargan de evaluar cuánto se acerca el orden de un determinado conjunto de productos sugeridos por el sistema al orden que un usuario le daría a ese mismo conjunto.

- *Half-life utility*. Se basa en el hecho de que los productos al principio de la lista de recomendación tienen mayor probabilidad de ser consultados por el usuario, probabilidad que decrece exponencialmente según se baja en la lista. Ha sido propuesta en [Breese et al., 1998]. Se calcula de acuerdo a la siguiente ecuación:

$$R_a = \frac{\sum_j \max(r_{aj} - d, 0)}{2^{(j-1)/(\alpha-1)}} \tag{2.8}$$

  donde d es un voto neutral, ligeramente negativo, y α es la posición del producto en donde hay un 50 % de probabilidades de ser consultado por el usuario (*the viewing half-life*).

2.6.2. Otras métricas

Como ya se ha comentado, las métricas de precisión no son suficientes para medir aspectos como la calidad o la utilidad de las recomendaciones. Es por ello que se emplean una serie de métricas para cubrir sus limitaciones. Entre ellas, podemos destacar las siguientes:

- Cobertura o *coverage*. Se trata del porcentaje de productos para los cuales el sistema es capaz de hacer predicciones [Good et al., 1999; Herlocker et al., 1999; Sarwar et al., 1998]. Generalmente, el *coverage* decrece con la precisión, es decir, cuánta mayor es el valor de esta métrica, menos precisas serán las recomendaciones. De hecho se trata de una métrica que puede ayudar a medir hasta qué punto nos podemos fiar de buenos resultados en métricas de precisión. Sistemas que son capaces de hacer predicciones sumamente precisas pero sólo para un número muy reducido de productos, suelen ser poco útiles, ya que estos productos serán productos populares que el usuario normalmente conoce.

- Novedad y *serendipity* [Sarwar et al., 2001]. Son métricas que miden la habilidad de los sistemas recomendadores de recomendar productos no esperados por el usuario.

- Diversidad. Esta métrica mide la variedad o las diferencias existentes entre los productos recomendados.

- Tasa de aprendizaje o *learning rate* [Herlocker et al., 2004; Schein et al., 2001]. Métrica que mide la capacidad que tiene el algoritmo de generar mejores recomendaciones conforme aumenta la cantidad de datos aprendidos.

- Confianza. Responde a la pregunta de cuánta certeza tiene el algoritmo de que sus recomendaciones son precisas [Herlocker et al., 2000].

- Eficiencia. Los algoritmos han de ser computacionalmente eficientes para poder ser empleados en un entorno real [Bell y Koren, 2007; Formoso et al., 2012; Koren, 2010a].

- Escalabilidad. Los sistemas de filtrado colaborativo tienen que manejar frecuentemente millones de usuarios o productos, por lo que la escalabilidad de los algoritmos se hace indispensable [Amatriain et al., 2009; Bell y Koren, 2007; Formoso et al., 2013b; Koren, 2010a].

Parte II

Dispersión en sistemas de recomendación de películas

Capítulo 3

Filtrado colaborativo

La finalidad de los sistemas recomendadores es proporcionar a los usuarios sugerencias que les puedan ser útiles. En el Capítulo 2 presentamos diversas técnicas de recomendación. De aquí en adelante nos centraremos en los algoritmos de filtrado colaborativo, que se basan únicamente en las puntuaciones que los usuarios otorgan a los diferentes productos. Son muy populares, tanto en el ámbito de la investigación como de la industria. Sin embargo, como muchas otras técnicas sufren el problema de la dispersión de los datos.

En este capítulo en primer lugar formalizaremos el problema de filtrado colaborativo. A continuación, en la Sección 3.2 mostraremos una visión general de las técnicas existentes. Después describiremos los *datasets* utilizados y la metodología de evaluación. Finalmente, mostraremos los experimentos realizados e incidiremos en cómo afecta la dispersión de los datos a las distintas técnicas en un dominio muy estudiado, como es el de la recomendación de películas. En la última sección expondremos las conclusiones obtenidas. El contenido de este capítulo se basa en los resultados que obtuvimos en [Cacheda et al., 2010, 2011a].

3.1. Formalización del problema

Entre todas las técnicas de recomendación nombradas hasta ahora a lo largo de esta tesis nos centraremos en las basadas en filtrado colaborativo [Goldberg et al., 1992]. Este tipo de técnicas basan su comportamiento en que, dado un usuario, las opiniones de otros usuarios con gustos similares serán valiosas para predecir qué productos le gustarían al primero. Esto es algo bastante habitual en la vida diaria. Por ejemplo, si tenemos gustos parecidos con un amigo, y éste nos recomienda una película que le ha gustado, nos fiaremos de su opinión y, probablemente, la película también nos gustará.

Las técnicas de filtrado colaborativo, como cualquier otro tipo de sistema de recomendación, tienen como objetivo sugerir productos útiles a los usuarios, ofreciendo recomendaciones personalizadas. Para ello, emplean únicamente las puntuaciones que los usuarios otorgan a los productos.

Más formalmente, en un sistema de recomendación basado en filtrado colaborativo existirá un conjunto de usuarios $U = \{u_1, u_2, \ldots, u_n\}$ y un conjunto de productos $I = \{i_1, i_2, \ldots, i_n\}$. Para cada usuario $u_i \in U$ el sistema almacenará un perfil asociado formado por el subconjunto de productos que el usuario ha valorado, $I_u \subseteq I$, y sus correspondientes puntuaciones, tal y como se muestra en la Ecuación 3.1.

$$P_u = \langle (i_1, r_{ui_1}), (i_2, r_{ui_2}), \ldots, (i_s, r_{ui_s}) \rangle \tag{3.1}$$

De modo análogo, se define también el subconjunto de usuarios que han valorado un cierto producto, $U_i \subseteq U$. El usuario para el cual el sistema está calculando la predicción se denomina usuario activo o usuario actual, u_a.

Las puntuaciones serán números enteros pertenecientes a un determinado intervalo, siendo R el conjunto de todas las posibles puntuaciones, r_{min} la puntuación mínima, r_{max} la puntuación máxima, $\hat{r}$ la puntuación media.

Se define también la matriz de puntuaciones, M, que representa el conjunto de todos los perfiles de usuario, o lo que es lo mismo, las puntuaciones que los usuarios han hecho de los productos. Cada posición de M, $m_{ui} \in R \cup \varnothing$, representa la puntuación que el usuario u realizó del producto i, lo cual equivale a r_{ui}. El valor $\varnothing$ indica que el usuario no ha valorado ese producto todavía. La tarea de predicción consistirá precisamente en intentar obtener la valoración para estos casos. Denotaremos con $\hat{r}_{ui} \in R \cup \varnothing$ la predicción que el algoritmo hace del producto $i \in I$ para el usuario $u \in U$.

Definimos también el subconjunto de todas las puntuaciones realizadas por un mismo usuario, $r_{u.} = \{r_{ui} \in M \mid i \in I_u\}$, y el subconjunto de todas las puntuaciones que ha recibido un determinado producto, $r_{.i} = \{r_{ui} \in M \mid u \in U_i\}$. Además, $\bar{r}_{u.}$ será la puntuación media realizada por un determinado usuario, y $\bar{r}_{.i}$ la puntuación media de un producto.

Finalmente, el objetivo de la tarea de recomendación será obtener una lista de productos $rec(u) = \{i_1, i_2, \ldots, i_n \mid i \in I - I_u\}$.

3.2. Tipos de técnicas de filtrado colaborativo

En función de cómo procesan la matriz de puntuaciones los algoritmos de filtrado colaborativo se suelen clasificar en algoritmos basados en memoria y algoritmos basados

en modelo.

Los algoritmos basados en memoria, los primeros en surgir [Resnick et al., 1994], procesan toda la matriz de puntuaciones para calcular la similitud entre los usuarios o productos y hacer predicciones o recomendaciones de acuerdo a estas similitudes calculadas. Existen abundantes ejemplos en donde este tipo de algoritmos han sido desplegados. La razón de su uso es que son fácilmente implementables y altamente efectivos [Linden et al., 2003]. Generalmente, los algoritmos basados en memoria, a pesar de ser más simples que los basados en modelo, obtienen resultados bastante precisos. Además, es fácil añadir incrementalmente nuevos datos.

Sin embargo, estos algoritmos presentan también varias limitaciones. Por ejemplo, son más sensibles que los algoritmos basados en modelo a algunos problemas comunes de los sistemas recomendadores como la dispersión de los datos, el *cold-start* o los *shilling attacks*. El principal problema es que cuando los datos son dispersos el número de productos en común entre los usuarios tiende a ser bajo, por lo que las similitudes se basan en pocos productos también, lo cual lleva a que el cálculo de estas similitudes sea poco fiable. Además, al tener que procesar toda la matriz de puntuaciones cada vez que realizan una predicción, presentan importantes problemas de escalabilidad, lo cual los hace desaconsejables para un sistema *online* con elevado número de usuarios o productos, debido a la necesidad de que las recomendaciones sean en tiempo real.

Para intentar solucionar los problemas de los algoritmos basados en memoria, surgieron los algoritmos basados en modelo. Así, por ejemplo, el problema de *sparsity* disminuye al aplicar técnicas basadas en modelo, puesto que reducen la dimensionalidad de la matriz de puntuaciones, y así ésta pasa a tener una mayor densidad de datos. En gran medida, la reducción (aunque sólo parcial) de los problemas de los algoritmos basados en memoria se debe a su habilidad para obtener características ocultas en los datos y extraer así más información. Para ello, construyen un modelo a partir de las puntuaciones de los usuarios, que representa su comportamiento. Será empleado para realizar las predicciones y recomendaciones [Breese et al., 1998]. Los parámetros del modelo se estiman *offline* a partir de los datos de la matriz de puntuaciones.

Los algoritmos basados en modelo también tienen que tratar con una serie de problemas. En primer lugar, no arreglan completamente las limitaciones de los algoritmos basados en memoria. Además, suelen ser sumamente complejos, ya que poseen una gran cantidad de parámetros que estimar (a veces sobreajustados), y que acostumbran a ser demasiado sensibles a cambios en los datos. Otras veces los datos no cumplen las suposiciones del modelo, lo cual lleva a malas recomendaciones. Por ejemplo, los algoritmos de reducción de dimensionalidad, que veremos en la Sección 3.3.3.3, pierden

información útil durante la construcción del modelo, debido a que se descarta la información menos relevante o a que determinados matices no pueden ser representados por el propio modelo. En la práctica, muchos modelos teóricos no se pueden aplicar con datos reales. Finalmente, a pesar de ser más rápidos en predicción que los basados en memoria, la construcción del modelo (y su actualización) normalmente consume mucho tiempo.

Para evitar estos problemas, algunos autores han desarrollado algoritmos que usan modelos que son rápidos y fáciles de calcular [Lemire y Maclachlan, 2005], o han hecho variantes eficientes y precisas de algoritmos populares [Funk, 2006; Paterek, 2007]. Otros autores han combinado distintas técnicas colaborativas para aprovechar así lo mejor de cada una [Koren, 2008; Pennock et al., 2000].

3.3. Principales algoritmos para predicción

La tarea de predicción es una tarea fundamental de cualquier sistema de recomendación. En esta sección ofreceremos una visión general de las alternativas existentes [Cacheda et al., 2011a; Su y Khoshgoftaar, 2009], poniendo especial atención en los algoritmos de filtrado colaborativo que emplearemos a lo largo de esta tesis.

3.3.1. Algoritmos basados en memoria

Los algoritmos basados en memoria, como ya se ha dicho, procesan toda la matriz de puntuaciones para generar una predicción o recomendación. Entre ellos, los algoritmos basados en vecinos sobresalen por su popularidad [Koren, 2008]. De hecho, se trata de una de las estrategias más populares en cuanto a filtrado colaborativo se refiere. Estos algoritmos se basan en la idea de que similitudes en puntuaciones pasadas son un indicador de similitudes en futuras puntuaciones.

Los algoritmos basados en vecinos, se pueden dividir, a su vez, en algoritmos basados en usuarios [Resnick et al., 1994; Shardanand, 1994] y algoritmos basados en productos [Sarwar et al., 2001], que se centran en buscar similitudes entre los usuarios o entre los productos, respectivamente. Dado que son muy similares, nos centraremos en los algoritmos basados en usuarios, mencionando explícitamente los basados en productos cuando sea necesario.

Este tipo de algoritmos siguen un proceso en tres pasos [Herlocker et al., 2002]:

1. Cálculo de la similitud entre el usuario activo (o el producto) y el resto de usuarios (o productos) [Herlocker et al., 2004].

- Coeficiente de correlación de Pearson.

  Esta es una de las primeras técnicas propuestas [Resnick et al., 1994] y también una de las más usadas. La similitud entre los usuarios se mide en función de su correlación, usando el coeficiente de Pearson:

$$s(a,u) = \frac{\sum_{i \in I_a \cap I_u} (r_{ai} - \overline{r_{a.}})(r_{ui} - \overline{r_{u.}})}{\sqrt{\sum_{i \in I_a \cap I_u} (r_{ai} - \overline{r_{a.}})^2 \sum_{i \in I_a \cap I_u} (r_{ui} - \overline{r_{u.}})^2}} \tag{3.2}$$

- Coeficiente de Pearson forzado. Es una variante de la medida de similitud anterior propuesta en [Shardanand y Maes, 1995], que consiste en sustituir en la Ecuación 3.2 la media del usuario por la puntuación media entre todas las disponibles ($\bar{r}$). Con esta idea se tienen en cuenta la diferencia entre puntuaciones positivas (las que están por encima de la puntuación media) y puntuaciones negativas (las que están por debajo).

$$s(a,u) = \frac{\sum_{i \in I_a \cap I_u} (r_{ai} - \bar{r})(r_{ui} - \bar{r})}{\sqrt{\sum_{i \in I_a \cap I_u} (r_{ai} - \bar{r})^2 \sum_{i \in I_a \cap I_u} (r_{ui} - \bar{r})^2}} \tag{3.3}$$

- Coeficiente de Pearson ponderado. Esta medida se basa en la idea de capturar la confianza que se puede poner en un determinado vecino. El problema es que si dos usuarios tienen pocos productos en común, se pueden considerar similares simplemente si, por casualidad, las puntuaciones de estos pocos productos coinciden. Conforme el número de productos en común aumenta, esta coincidencia es más probable que sea debida a que los usuarios son realmente similares, y no a pura casualidad. Así, la confianza en la medida de similitud aumenta conforme existen más productos en común. Lo que se trata es de reducir la importancia de una similitud cuando ésta se basa en muy pocas puntuaciones. La siguiente ecuación fue propuesta en [Herlocker et al., 1999] para ponderar el coeficiente de correlación de Pearson por el número de productos en común:

$$s(a,u) = \begin{cases} s_{pearson}(a,u)/\gamma & |I_a \cap I_u| < \gamma \\ s_{pearson}(a,u) & otherwise \end{cases} \tag{3.4}$$

  donde γ es un umbral determinado experimentalmente cuyo valor típico es 50 y que determina el número de productos a partir del cual se puede confiar en la medida de similitud.

- Distancia coseno. Los usuarios son tratados como un vector de puntuaciones [Breese et al., 1998]. El valor de la similitud es entonces el valor del coseno del ángulo que forman los vectores. Por tanto, un valor próximo a uno indicará similitud, mientras que un valor próximo a cero indicará ausencia de similitud.

$$s(a,u) = \sum_{j \in I} \frac{r_{aj}}{\sqrt{\sum_{k \in I_a} r_{ak}^2}} \frac{r_{uj}}{\sqrt{\sum_{k \in I_u} r_{uk}^2}} \tag{3.5}$$

- Coseno ajustado.

  Se trata de una distancia coseno en la que se tienen en cuenta las medias de las puntuaciones de los usuarios. En [Sarwar et al., 2001] se concluye que esta medida es la que obtiene mejores resultados para el cálculo de similitudes entre productos. En la Ecuación 3.6 se puede comprobar el parecido de esta medida con el coeficiente de correlación de Pearson.

$$s(i,j) = \frac{\sum_{u \in U} (r_{ui} - \overline{r_{u.}}) (r_{uj} - \overline{r_{u.}})}{\sqrt{\sum_{u \in U} (r_{ui} - \overline{r_{u.}})^2 \sum_{u \in U} (r_{uj} - \overline{r_{u.}})^2}} \tag{3.6}$$

- Mean Squared Difference (MSD) [Shardanand, 1994]. Esta medida calcula la similitud entre dos usuarios como la diferencia cuadrática media entre las puntuaciones de ambos usuarios. Para ello, la fórmula aplicar es la siguiente:

$$msd(a,u) = \frac{\sum_{i \in I_a \cap I_u} (r_{ai} - r_{ui})^2}{|I_a \cap I_u|} \tag{3.7}$$

  Después, se filtran aquellos usuarios cuya diferencia supere un cierto umbral, L, y el resto de similitudes se ponderan según la ecuación:

$$s(a,u) = \frac{L - msd(a,u)}{L} \tag{3.8}$$

Aparte de las ya mencionadas existen otras medidas de similitud, como Spearman Rank Correlation (SRC), que, a diferencia del coeficiente de Pearson, usa el ranking de la puntuación en vez de la puntuación en sí misma; la correlación τ de Kendall, que usa un ranking relativo en lugar de un ranking absoluto [Herlocker et al., 2004]; o la similitud basada en probabilidad condicional [Deshpande y Karypis, 2004; Karypis, 2001].

2. Selección de los vecinos, un conjunto de usuarios (o productos) en función de su similitud con el usuario activo (o el producto).

Para este paso, consideraremos las siguientes alternativas:

- Umbral de correlación [Shardanand y Maes, 1995]. Consiste en seleccionar sólo aquellos usuarios (o productos) cuya similitud con el usuario activo (o producto) sobrepase un determinado umbral.
- Máximo número de vecinos [Resnick et al., 1994]. Consiste en seleccionar los k usuarios que sean más similares al usuario activo (o al producto), donde k será un parámetro del algoritmo. Es por ellos que los algoritmos basados en vecinos que usan el máximo número de vecinos se conocen también con el nombre de algoritmos kNN (*k-Nearest Neighbors*).

3. Cálculo de la predicción usando las puntuaciones de los vecinos.

   La forma más sencilla de calcular la predicción de la puntuación que un usuario u realizaría de una producto i, es la siguiente:

$$\hat{r}_{ai} = \frac{\sum_{u \in Neigh_a} r_{ui} s(a,u)}{\sum_{u \in Neigh_a} |s(a,u)|} \tag{3.9}$$

   en donde $N_i(a)$ es el vecindario de u que ha valorado el producto i. El problema de esta aproximación es que no considera que dos usuarios diferentes pueden emplear distintas puntuaciones para indiciar un mismo grado de relevancia, es decir, pueden usar distintas escalas. Es por ello que existen alternativas para la normalización de las puntuaciones:

   - Normalización centrada en la media [Resnick et al., 1994]. Se comprueba si un producto tiene una puntuación positiva o negativa, comparando ésta con la media del usuario (o del producto), según la Ecuación 3.10.

$$\hat{r}_{ai} = \overline{r_{a.}} + \frac{\sum_{u \in Neigh_a} [r_{ui} - \overline{r_{u.}}] \, s(a,u)}{\sum_{u \in Neigh_a} |s(a,u)|} \tag{3.10}$$

   - Normalización z-score [Herlocker et al., 2002]. Esta normalización no sólo tiene en cuenta la media del usuario (o del producto), sino también su desviación típica. De esta forma se considera que los usuarios pueden puntuar siguiendo distintas distribuciones.

$$\hat{r}_{ai} = \overline{r_{a.}} + \sigma_a \frac{\sum_{u \in Neigh_a} \left[\left(\frac{r_{ui} - \overline{r_{u.}}}{\sigma_u} \right) s(a,u) \right]}{\sum_{u \in Neigh_a} s(a,u)} \tag{3.11}$$

Existe otra alternativa para asignar una puntuación a un producto, distinta a la aproximación de predicción comentada, pero que no emplearemos en esta tesis. Se denomina *clasificación* y consiste en asignar como puntuación aquélla que sume el mayor valor de similitud entre los vecinos del usuario u que han dado esa puntuación al producto i.

3.3.2. Similarity Fusion

En la mayoría de los sistemas reales los usuarios sólo suelen puntuar un porcentaje pequeño de los productos disponibles, lo cual provoca que las matrices sean dispersas. El problema que tienen las aproximaciones basadas en usuario y en modelo es que sólo usan una parte de la información disponible en la matriz de puntuaciones, por lo que este problema de la dispersión de datos se hace todavía más palpable. Es por ello que en [Wang et al., 2006] se presenta un método alternativo, que considera tanto las relaciones entre los usuarios como las relaciones entre los productos.

El algoritmo propuesto combina las puntuaciones de los usuarios similares al usuario activo (SUR), las puntuaciones del usuario activo sobre productos parecidos (SIR), y las puntuaciones de usuarios similares al activo sobre productos similares (SUIR). La fórmula se muestra en la Ecuación 3.12. En ella δ y λ son parámetros determinados experimentalmente que determinan el peso de cada una de las contribuciones.

$$\begin{aligned} \hat{r}_{ui} &= \sum_{r \in R} P(r_{ui} = r | SUR, SIR, SUIR) \\ &= \left(\sum_{r \in R} P(r_{ui} = r | SUIR)\delta \right) + \left(\sum_{r \in R} P(r_{ui} = r | SUR)\delta(1-\lambda) \right) \\ &\quad + \left(\sum_{r \in R} P(r_{ui} = r | SIR)(1-\delta)(1-\lambda) \right) \end{aligned} \tag{3.12}$$

3.3.3. Algoritmos basados en modelo

3.3.3.1. Basados en regresión lineal

Se trata de un algoritmo basado en producto; es decir, obtiene la predicción para un producto basándose en productos similares. La relación entre dos productos puede verse como un experto, modelado como una función lineal:

$$f_{ij}(x) = x\alpha_{ij} + \beta_{ij}$$

donde los parámetros α_{ij} y β_{ij} se estiman por medio de una regresión lineal simple usando las puntuaciones dadas por los usuarios a los correspondientes pares de productos. Cuando α_{ij} es cercano a $+1$ o -1, se dice que el experto es un buen predictor para la puntuación del producto.

La relación entre los productos se calcula en primer lugar, obteniéndose $n(n-1)$ expertos. Para obtener la predicción de la puntuación, se combinan las predicciones de los expertos del producto en cuestión.

En este trabajo, para llevar a cabo esta combinación, usaremos la media de todos los expertos, después de descartar aquéllos expertos cuyo coeficiente de regresión R^2 esté por debajo de un cierto umbral. En [Vucetic y Obradovic, 2000] se presentan otros dos métodos alternativos.

3.3.3.2. Slope One

Los algoritmos Slope One [Lemire y Maclachlan, 2005] son algoritmos basados en predictores de la forma $f(x) = x + b$. Por tanto, se trata de algoritmos más simples que los basados en regresión, ya que la pendiente de la recta es uno. Por su parte, la constante, b, se define como la diferencia media entre el productos correspondiente y el producto cuya puntuación se predice, calculado entre los usuarios que han valorado ambos productos. La predicción final se muestra en la ecuación siguiente:

$$\hat{r}_{uj} = \overline{r_{u.}} + \frac{1}{|R_j|} \sum_{j \in R_j} \sum_{x \in S_{ji}} \frac{r_{xi} - r_{xj}}{|S_{ji}|} \tag{3.13}$$

en donde S_{ji} es el conjunto de usuarios que han valorado los productos j e i , $S_{ji} = U_j \cap U_i$, y R_j el conjunto de productos valorados por el usuario para el cual $|S_{ji}| > 0$.

Este esquema se puede mejorar si se tiene en cuenta el número de productos disponibles para cada usuario, conduciendo al algoritmo Weighted Slope One:

$$\hat{r}_{uj} = \frac{\sum_{i \in I_u - i_j} \left(\sum_{x \in S_{ji}} \frac{r_{xi} - r_{xj}}{|S_{ji}|} + r_{ui} \right) |S_{ji}|}{\sum_{i \in I_u - i_j} |S_{ji}|} \tag{3.14}$$

Finalmente, también hemos estudiado la variante Bi-Polar Slope One. Esta alternativa considera la media del usuario como umbral, por lo que divide los productos entre aquéllos que el usuario ha valorado positivamente y aquéllos que ha valorado negativamente.

3.3.3.3. Modelos de factorización de matrices

Estos modelo se basan en el cálculo de matrices de dimensionalidad reducida a partir de la matriz de puntuaciones original. Estas nuevas matrices representan atributos latentes en las puntuaciones, permitiendo así encontrar relaciones entre los productos, y eliminando los problemas causados por la dispersión de la matriz y puntuaciones anómalas.

- LSI/SVD.

  Latent Semantic Indexing (LSI) se basa en *Singular Value Decomposition* (SVD), una técnica de factorización de matrices que convierte una matriz, R en tres matrices matrices, $R = U \cdot S \cdot V^T$, donde U y V son matrices ortogonales y S una matriz diagonal de tamaño $r \times r$ (donde r es el rango de la matriz R), formada por los valores singulares de la matriz de puntuaciones. Esta matriz se reduce descartando los valores más pequeños, para finalmente tener una matriz S_k, con $k < r$. La matriz reconstruida $R_k = U_k \cdot S_k \cdot V_k^T$ es la mejor aproximación de rango k de la matriz de puntuaciones. La predicción [Sarwar et al., 2000b] se calcula a partir de las matrices de dimensionalidad reducida, de acuerdo a la fórmula:

$$\hat{r}_{ui} = \overline{r_{u.}} + U_k \cdot \sqrt{S_k^T}(u) \cdot \sqrt{S_k} \cdot V_k^T(i) \tag{3.15}$$

- SVD regularizado. Este modelo, muy popular entre las técnicas de filtrado colaborativo, fue propuesto originalmente para este contexto en [Funk, 2006]. Cada producto se representa por un conjunto de características (aspectos), y cada usuario por un conjunto de valores que indican su preferencia por los distintos aspectos de los productos. La predicción de la puntuación se compone de la suma de ambos:

$$\hat{r}_{ij} = x_i^T y_j \tag{3.16}$$

  donde x_i e y_j son vectores k-dimensionales que representan, respectivamente, la afinidad de cada usuario y cada producto por cada una de las k características. Los valores de tales vectores se estiman en la fase de construcción del modelo, usando una variación de SVD donde las puntuaciones desconocidas se ignoran. Esto permite un modo sencillo de calcular SVD, mucho más rápido que el usado en el algoritmo anterior. Consiste en minimizar la suma de los residuos al cuadrado, usando una técnica de gradiente descendiente con regularización:

$$e_{ij} = r_{ij} - \hat{r}_{ij}$$
$$x_{ik} += lrate * (e_{ij} y_{jk} - \lambda x_{ik})$$
$$y_{jk} += lrate * (e_{ij} x_{ik} - \lambda y_{jk})$$

Cada vez se estima una sola característica.

- SVD regularizado mejorado (RSVD2). Una de las mejoras introducidas al modelo anterior por [Paterek, 2007]. Añade un parámetro adicional por usuario, c_i, y uno por producto, d_j, derivando en el siguiente modelo:

$$\hat{r}_{ij} = c_i + d_j + x_i^T y_j \tag{3.17}$$

- NSVD2. También introducida por [Paterek, 2007]. Este es un método que reduce el número de parámetros, modelando x_i en función de los productos valorados por el usuario. En este caso, el modelo es:

$$\hat{r}_{ij} = c_i + d_j + \sum_{k=1}^{K} y_{jk} \sum_{j_2 \in I_i} y_{j_2 k} \tag{3.18}$$

- SVD++ [Koren, 2008]. Variación del SVD regularizado donde se tiene en cuenta el *feedback implícito* (modelado como los productos que un usuario ha valorado):

$$\hat{r}_{ij} = b_{ij} + y_j^T \left(x_i + |I_u|^{-\frac{1}{2}} \sum_{j_2 \in I_u} w_{j_2} \right) \tag{3.19}$$

donde w es un vector de parámetros del modelo adicional que pondera la contribución del *feedback* implícito.

3.3.3.4. Cluster Based Smoothing

Un cluster contiene a un conjunto de elementos similares entre sí pero diferentes a los elementos de otros clusters. La comparación entre los distintos elementos se puede hacer según diferentes medidas de similitud. Existen diferentes tipos de algoritmos de *clustering*: algoritmos de particionado (como k-means [MacQueen, 1967]), fáciles de implementar y relativamente eficientes; algoritmos basados en densidad (como DBSCAN [Ester et al., 1996]), que buscan *clusters densos* separados por regiones dispersas que representan el ruido; y algoritmos jerárquicos (como BIRCH [Zhang et al., 1996]), que crean una descomposición jerárquica de los datos empleando distintos criterios.

El algoritmo que nos atañe funciona como los algoritmos basados en usuarios, con la excepción de que primero agrupa a los usuarios en *clusters*, para alcanzar dos objetivos: incrementar la densidad de la matriz de puntuaciones aproximando las puntuaciones desconocidas con la media del *cluster*, e incrementar la escalabilidad, al buscar vecinos sólo en los *clusters* más cercanos al usuario activo. El algoritmo, detallado en [Xue et al., 2005], consta de cinco fases:

1. Los usuarios, de manera *offline*, se agrupan en k *clusters*, usando el algoritmo k-means.
2. La media del *cluster* se usa para completar el perfil del usuario, es decir, a los productos que no ha valorado un usuario se les asigna un puntuación igual a la media de las puntuaciones dadas por el resto de usuarios del *cluster*.
3. Se seleccionan los *clusters* más cercanos al usuario activo.
4. La búsqueda de los vecinos se lleva a cabo sólo entre los usuarios pertenecientes a los *clusters* escogidos.
5. Finalmente, la predicción se calcula usando la media ponderada de las puntuaciones de los vecinos.

Existen muchos otros algoritmos basados en *clustering* como, por ejemplo, técnicas basadas en solapamiento de *clusters* [Shafiei y Milios, 2006] o *co-clustering* [George y Merugu, 2005].

3.3.3.5. Basado en tendencias

Se trata de un algoritmo que propusimos en [Cacheda et al., 2011a] y que hemos patentado (tal y como expusimos en la Sección 10.2.1. Se basa en buscar variaciones entre los usuarios a la hora de puntuar, en lugar de basarse en las relaciones entre usuarios o productos. De esta forma hacemos frente a algunas de las variabilidades explicadas en la Sección 2.3, como, por ejemplo, el uso de distintas escalas por parte de dos usuarios con los mismos gustos, o la influencia del contexto a la hora de realizar una valoración.

Tiene en cuenta tanto las tendencias (variaciones) de los usuarios como de los productos, las cuales son fáciles de calcular y necesitan pocos datos, lo cual hace que el algoritmo sea eficiente. Entenderemos por tendencia de un usuario lo proclive que es a puntuar positivamente (por encima de la media del producto) o negativamente (por debajo de la media) un determinado producto. Por tanto, definimos la tendencia de

un usuario, ub_u, como la media de sus puntuaciones con respecto a las medias de los productos que ha valorado.

$$ub_u = \frac{\sum_{i \in I_u} (r_{ui} - \overline{r_{.i}})}{|I_u|} \tag{3.20}$$

La tendencia de un producto, ib_i, medirá lo habitual que es que la puntuación de un determinado producto supere la media de los usuarios. Se calcula del siguiente modo:

$$ib_i = \frac{\sum_{u \in U_i} (r_{ui} - \overline{r_{u.}})}{|U_i|} \tag{3.21}$$

Lógicamente, el valor de ambas tendencias variará entre $-d$ y d, donde d es la diferencia entre las puntuaciones máxima y mínima permitidas.

A la hora de realizar las predicciones, el algoritmo tiene en cuenta tanto las tendencias como las medias de los usuarios y productos, lo cual hace que se puedan dar cuatro casos:

1. Las tendencias de usuario y producto son positivas.

$$\hat{r}_{ui} = \text{máx}(\overline{r_{u.}} + ib_i, \overline{r_{.i}} + ub_u) \tag{3.22}$$

2. Las tendencias de usuario y producto son negativas.

$$\hat{r}_{ui} = \text{mín}(\overline{r_{u.}} + ib_i, \overline{r_{.i}} + ub_u) \tag{3.23}$$

3. Las tendencias tienen distinto signo, pero las medias corroboran las tendencias. Es decir, si el usuario tiene la tendencia de puntuar por encima de las medias de los productos (tendencia del usuario positiva), el producto suele ser valorado por debajo de las medias de los usuarios (tendencia del producto negativa), la media de la puntuaciones del usuario ha de ser mayor que la del producto para poder predecir como puntuación un valor entre ellos.

$$\hat{r}_{ui} = \text{mín}(\text{máx}(\overline{r_{u.}}, (\overline{r_{.i}} + ub_u)\beta + (\overline{r_{u.}} + ib_i)(1 - \beta)), \overline{r_{.i}}) \tag{3.24}$$

   en donde β es un parámetro que controla la contribución de cada uno de las medias.

4. Las tendencias tienen distinto signo, pero las medias no corroboran dichas tendencias.

$$\hat{r}_{ui} = \overline{r_{.i}}\beta + \overline{r_{u.}}(1 - \beta) \tag{3.25}$$

   en donde, de nuevo, β controla la contribución de cada uno de las medias.

3.3.4. Técnicas colaborativas híbridas

Los algoritmos basados en filtrado colaborativo pueden ser combinados entre sí formando técnicas híbridas, cuyo rendimiento suele mejorar el de las técnicas individualmente [Pennock et al., 2000; Yu et al., 2004]. A cambio de una mejora en la precisión, a menudo resultan mucho más complejos y suelen estar pensados para un dominio muy concreto, lo cual les confiere una menor capacidad de adaptación a otros conjuntos de datos y dominios.

3.3.4.1. Personality Diagnosis

Este algoritmo [Pennock et al., 2000] se basa en un modelo probabilístico simple que aproxima el modo en que los usuarios valoran los productos por medio de una distribución normal. La idea es que un usuario, cuando evalúa un producto específico en momentos distintos, puede dar cada vez una puntuación diferente. La diferencia viene motivada por varias causas: humor, otros productos valorados en el mismo contexto... Por tanto, el perfil o *personalidad* de un usuario, correspondiente a sus puntuaciones, debería acompañarse de esta variación, que el algoritmo modela como simple ruido gaussiano. Así, $P(r_{ij} = x | r_{ij}^{true} = y) \simeq e^{-(x-y)^2/2\sigma^2}$, donde r_{ij}^{true} es la puntuación real del usuario.

Usando este modelo, el algoritmo calcula la probabilidad de que el usuario activo tenga la misma personalidad que otros usuarios. Entonces, estas probabilidades se usan para calcular la distribución de probabilidad de la puntuación de un determinado producto. Finalmente, la predicción es la puntuación más probable.

3.3.4.2. Modelo integrado: basado en vecinos – SVD++

Este algoritmo, propuesto en [Koren, 2008], combina el algoritmo SVD++ descrito en la Sección 3.3.3.3 con una aproximación basada en vecinos que emplea una optimización de una función de coste global.

3.3.5. Otros algoritmos

Existen otros muchos algoritmos que no trataremos en esta tesis. Entre ellos, cabe destacar diferentes alternativas que emplean clasificadores bayesianos [Friedman et al., 1997; Miyahara y Pazzani, 2000], modelos de semántica latente [Hofmann, 2001, 2004], de aspectos [Hofmann y Puzicha, 1999], multinomiales [Breese et al., 1998; Marlin, 2004a,b], basados en reglas de asociación [Sarwar et al., 2000a], etc.

3.4. Tarea de recomendación

Hasta el momento, nos hemos centrado en la tarea de predicción. En la literatura existen pocos trabajos sobre la tarea de recomendación. Sólo para sistemas con preferencias implícitas o unarias esta tarea ha resultado relevante, a pesar de ser empleada con éxito en múltiples sistemas comerciales con preferencias tanto implícitas como explícitas.

Diversos autores [Cremonesi et al., 2010; McLaughlin y Herlocker, 2004] han llegado a la conclusión de que los algoritmos ideados para la tarea de predicción no obtienen en recomendación tan buenos resultados como sería deseable. De hecho, muchos de estos algoritmos para recomendar una lista de productos únicamente calculan la preferencia de todos los productos del sistema, para después escoger aquéllos con la puntuación más alta, con lo que se eleva el coste computacional.

Algunos de los algoritmos que más fácilmente se pueden adaptar a esta tarea son los basados en vecinos. Para ello la única variación radica en la última fase mencionada en la Sección 3.3.1, el cálculo de la predicción. En su lugar, se obtendrá la lista de los N productos recomendados. Para ello, en vez de calcular la puntuación que el usuario otorgaría a cada uno de los productos del sistema, simplemente se le asigna un peso a cada producto y se recomiendan los N con mayor peso. De esta manera los valores de los pesos no están restringidos a los de las puntuaciones, los elementos son más fácilmente comparables y sólo es necesario calcular estos valores para los productos que haya valorado algún vecino, y no para todos, como sucede por norma general en la tarea de recomendación. Además, es necesario únicamente un vecindario para cada usuario, a diferencia de en la tarea de predicción, en donde es necesario calcular un vecindario para cada producto cuya puntuación se quiera predecir.

3.5. Experimentos

Una vez presentadas las técnicas de filtrado colaborativo que vamos a emplear, se mostrarán una serie de experimentos realizados que ayudarán a comprender la influencia de la dispersión de los datos en el comportamiento de los algoritmos de recomendación.

3.5.1. *Datasets*

Para la evaluación de los experimentos emplearemos dos *datasets*: MovieLens [Herlocker et al., 1999] y Netflix [Bennett y Lanning, 2007]. Se trata de unos de los conjuntos de datos más usados por investigadores y desarrolladores en el campo del filtrado colaborativo, por ser accesibles públicamente. Por ello son especialmente interesantes,

puesto que permiten comparar los resultados de diferentes trabajos entre sí. Ambos se obtuvieron a partir de sus respectivos recomendadores de películas *online*.

El *dataset* de MovieLens contiene las puntuaciones obtenidas durante un periodo de siete meses (del 19-09-1997 al 22-04-1998). Para los experimentos realizados, hemos eliminado los usuarios con menos de 20 puntuaciones, consiguiendo así un total de 100,000 puntuaciones, de 943 usuarios sobre 1682 películas. Por tanto, existen 1.78 películas por cada usuario, y la densidad es del 6 %.

Por otro lado, el *dataset* de Netflix contiene sobre 100 milliones de puntuaciones, realizadas por 480,189 usuarios sobre 17,770 películas, recogidas entre octubre de 1998 y diciembre de 2005. Este *dataset* se ha hecho sumamente popular en los últimos años, después de haber sido usado por investigadores y competidores del concurso de Netflix [Bennett y Lanning, 2007]. Sin embargo, en estos primeros experimentos no hemos usado el *dataset* completo. La razón es que muchos de los algoritmos que hemos evaluado no escalan suficientemente bien con tal cantidad de datos. Así, para evaluar dichos algoritmos, hemos seleccionado aleatoriamente un 30 % de usuarios y productos, obteniendo un *dataset* con 9,132,089 puntuaciones hechas por 144,190 usuarios sobre 5,354 productos. También hemos comparado resultados de 100 tests con distintos algoritmos y/o configuraciones en el *dataset* completo y en el reducido, para comprobar la correlación entre ambos. Hemos obtenido un fuerte correlación lineal, con un R^2 del 99.91 % y un p-valor de la ANOVA de 0.000, lo cual muestra que podemos confiar en los resultados del *dataset* reducido para estimar los resultados del completo. Finalmente, es de destacar que, a diferencia de MovieLens, el *dataset* reducido de Netflix contiene menos productos que usuarios (en concreto, 0.04 productos por usuario), y una densidad del 1.2 %.

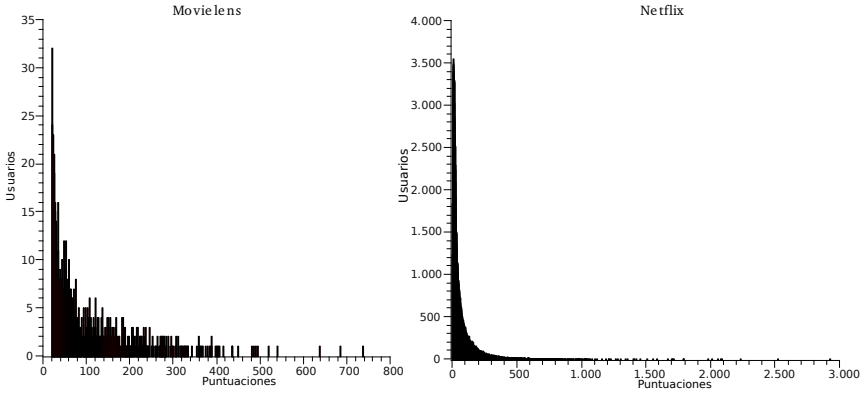


Figura 3.1: Puntuaciones por usuario en los *datasets* de MovieLens y Netflix.

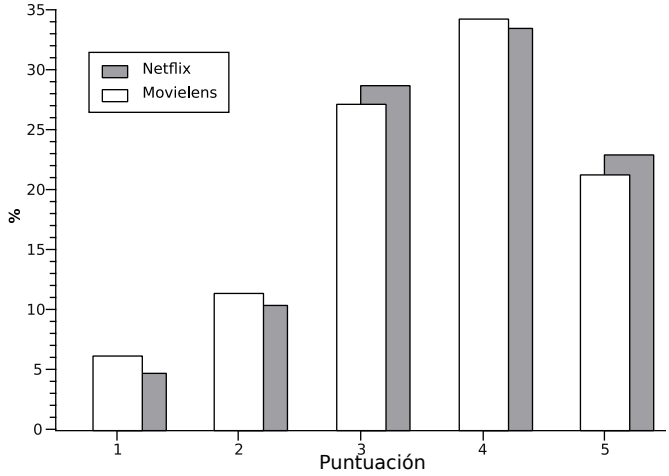


Figura 3.2: Porcentaje de cada puntuación, en ambos *datasets*.

Se puede ver también que la distribución de las puntuaciones en ambos *datasets* es bastante similar. Primero, como se aprecia en la Figura 3.1, la mayoría de los usuarios realizan un bajo número de puntuaciones. Además, en ambos casos las puntuaciones son discretas y varían entre 1 (más baja) y 5 (más alta). Como se observa en el histograma de puntuaciones de la Figura 3.2, los usuarios tienden a valorar películas que les han causado buena impresión, siendo ésta la razón por la que se producen más puntuaciones de 3, 4 y 5 puntos, que de 1 y 2. Estas similitudes entre ambos *datasets* no son sorprendentes, dado que ambos pertenecen al mismo dominio: la recomendación de películas. No deberíamos olvidar que los algoritmos dependen fuertemente de las características del *dataset* sobre el que se usan, así que los resultados obtenidos pueden no coincidir con los obtenidos en otro dominio diferente. La dispersión de la matriz, la relación entre el número de usuarios y el número de productos, la variabilidad de las puntuaciones o la escala pueden influir notablemente en la precisión de un algoritmo.

3.5.2. Metodología

Para realizar los experimentos se ha dividido cada *dataset* en dos subconjuntos: el de entrenamiento y el de evaluación. El primero se corresponde con los datos que el algoritmo conoce. Con esa información, el algoritmo realizará sus recomendaciones, que serán comparadas con los datos originales presentes en el subconjunto de evaluación.

Para construir el subconjunto de entrenamiento hemos seguido dos aproximaciones diferentes. En la primera, se ha seleccionado un porcentaje de las puntuaciones disponibles. Para nuestras pruebas hemos usado los siguientes porcentajes: 10 %, 20 %, 30 %, 40 %, 50 %, 60 %, 70 %, 80 % y 90 %. El subconjunto de evaluación, por su parte, se ha

creado seleccionando aleatoriamente un 10 % del *dataset*. Previamente se había comprobado que el porcentaje de este subconjunto no tiene gran impacto en los resultados. Lógicamente, las puntuaciones que aparecen en el subconjunto de evaluación no son incluidos en el de entrenamiento. Con altos porcentajes podemos evaluar el comportamiento del algoritmo bajo condiciones de densidad relativamente altas. En contraste, un pequeño porcentaje permite evaluar el algoritmo bajo condiciones de baja densidad de datos (condiciones de *sparsity*), muy común en las primeras fases, o en dominios con un gran número de usuarios y/o productos.

Precisamente, dado que, como ya hemos comentado en el Capítulo 2, la mayoría de los sistemas tienen que trabajar en condiciones de baja densidad de datos, con datos dispersos, hemos usado un segundo método para construir los subconjuntos de entrenamiento y evaluación, y así probar los algoritmos bajo condiciones de *sparsity*. Consiste en seleccionar como subconjunto de entrenamiento un número fijo (generalmente bajo) de puntuaciones de cada usuario. Hemos probado seleccionando 1, 2, 3, 4, 5, 7, 10, 12, 15 y 20 puntuaciones. Este método se denomina *Given-N* y fue usado por primera vez en [Breese et al., 1998]. Hemos evaluado las dos tareas más comunes de los sistemas recomendadores: la de predicción y la de recomendación. Para esta última, hemos probado a recomendar 5, 10, 15, 25, 50, 100 y 150 productos.

En la tarea de recomendación hemos tenido que abordar un problema inherente al uso de *datasets offline*: las preferencias de los usuarios para la mayoría de los productos son desconocidas. En estos casos, pequeños errores relacionados con los productos valorados, como incluir un producto con una puntuación baja o dejar sin recomendar un producto con una puntuación alta, afectan considerablemente al resultado final [Herlocker et al., 2004]. Para minimizar este problema, en la evaluación hemos forzado a los algoritmos a recomendar productos presentes en el subconjunto de evaluación. Por tanto, la lista final consistirá en N productos que ya fueron valorados por el usuario en cuestión. Aunque esto permite afrontar el problema, sólo una evaluación con usuarios reales puede solucionarlo satisfactoriamente.

En la tarea de predicción este problema es más pequeño: conforme se hace más significativa la cantidad de datos de evaluación, los resultados se pueden extrapolar mejor a todos los productos. Aún así, los sesgos presentes en el *dataset* pueden influir en los resultados. Por ejemplo, en un *dataset* como el que hemos usado, donde la mayoría de las evaluaciones se corresponden con productos con alta puntuación, un algoritmo capaz de predecir correctamente este tipo de productos obtendrá mejores resultados que uno que prediga mejor los productos con baja puntuación, simplemente porque los posibles errores cometidos en productos malos tendrán menos peso en la

media final. Sin embargo, en este caso esto no supone un problema, ya que estamos interesados precisamente en un algoritmo que prediga correctamente productos con altas puntuaciones, aunque se pueden dar otros sesgos en los *datasets*.

Hemos usado para la evaluación algunas de las métricas más populares presentadas en el Capítulo 2: *coverage*, MAE, RMSE, GIM, GPIM, *Precision*, *Recall*, ROC y *Half-Life Utility*. Cada prueba se ha repetido cinco veces, siendo los resultados que presentaremos en este capítulo las medias obtenidas.

En cuanto a los algoritmos evaluados (ver Capítulo 2), hemos considerado un algoritmo basado en usuario (UB), un algoritmo basado en producto (IB), y *Similarity Fusion* (SF) como algoritmos basados en memoria; un algoritmo basado en regresión (RB), *Slope One* (SO), LSI/SVD (SVD), RSVD, RSVD2, NSVD2, SVD++, el algoritmo de Cluster Based Smoothing (CBS), el algoritmo integrado de Koren98 (IM) y el algoritmo basado en tendencias (TB) como algoritmos basados en modelo; y *Personality Diagnosis* (PD) como algoritmo híbrido.

3.6. Resultados

3.6.1. Datos dispersos en predicción

En primer lugar estudiaremos cómo influye la cantidad de datos considerados para el subconjunto de entrenamiento en la precisión y en el *coverage* de los algoritmos. Lógicamente los algoritmos mejorarán sus resultados conforme el porcentaje de datos para el entrenamiento aumenta, ya que se incrementa la densidad de datos en la matriz de puntuaciones y, por tanto, el algoritmo tiene más información para calcular la predicción. Los resultados para el *dataset* de MovieLens se muestran en las Figuras 3.3, 3.4 y 3.5, mientras que para el de Netflix, en la Figura 3.6.

Del mismo modo, conforme la densidad de información aumenta, se aprecia un ligero decrecimiento en las diferencias entre los algoritmos. La mayoría presentarán resultados parecidos bajo condiciones de densidad relativamente alta, mientras que dichas diferencias se acentuarán cuando la densidad disminuya. De hecho, en el *dataset* de MovieLens y con un subconjunto de entrenamiento del 80 %, no existe significancia estadística (con un 95 % de nivel de confianza usando el contraste múltiple de Bonferroni) sobre las diferencias en MAE entre los seis mejores algoritmos (UB, RSVD2, SVD++, RSVD, SO y TB), mientras que con un 10 % sólo los tres mejores algoritmos (RSVD2, SVD++ y TB) presentan resultados equivalentes. Se pueden obtener las mismas conclusiones a partir de los resultados conseguidos con el *dataset* de Netflix. Con un subconjunto de entrenamiento del 80 %, no existe ninguna diferencia estadísticamente significativa

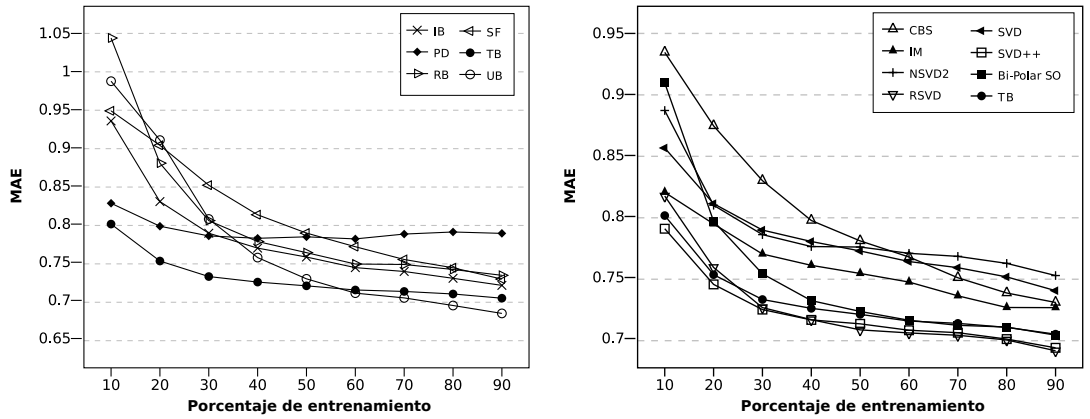


Figura 3.3: Evolución de la precisión (en MAE) en el *dataset* de MovieLens según la densidad de la matriz.

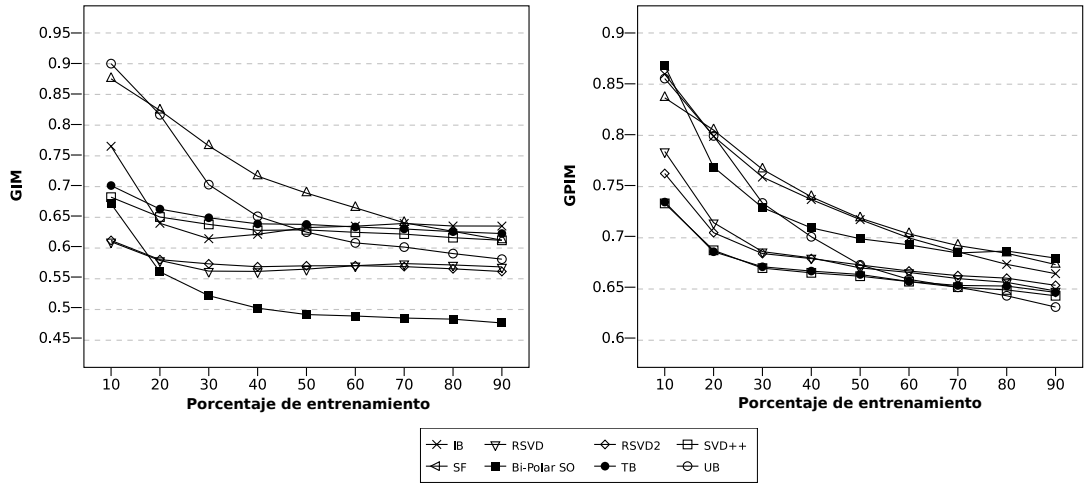


Figura 3.4: Evolución de GIM y GPIM según la densidad de la matriz, en el *dataset* de MovieLens.

entre los mejores algoritmos (RSVD, RSVD2, SVD++, SO y TB), mientras que con el 10 % SVD++ presenta los mejores resultados, seguido de RSVD2, NSDV2, RSVD y TB.

En el Capítulo 2 hemos comentado la existencia de diferentes fuentes de variabilidad a la hora de realizar puntuaciones. Debido a ellas, diversos autores proponen la existencia de un límite en la precisión en la predicción. Aunque no negamos la existencia de este límite, nuestros resultados muestran, en contraste, que la mayor limitación que sufre un algoritmo es precisamente la falta de información. Por esta razón en esta tesis nos centraremos en ella y en diferentes modos de abordarla. Cuanto mayor sea

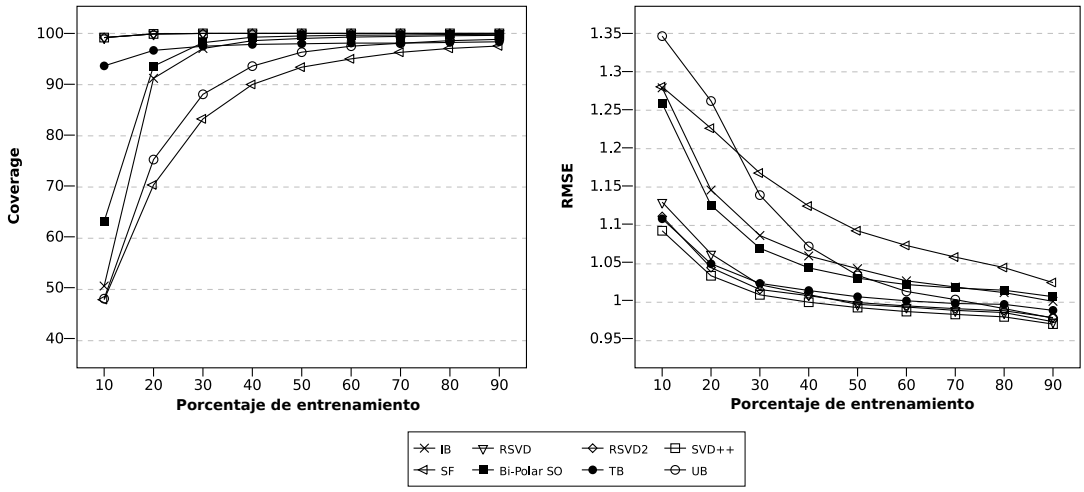


Figura 3.5: Evolución de RMSE y de *coverage* según la densidad de la matriz, en el *dataset* de MovieLens.

la información disponible, y la densidad de datos en la matriz de puntuaciones, mayor será también la información que será capaz de obtener un algoritmo, y, en consecuencia, mejores será los resultados que se puedan obtener.

Ya que no disponemos de una evaluación *online* del sistema para poder aclarar como mayor exactitud si los errores en la precisión son realmente debidos a la falta de información o a la variación natural en las puntuaciones de los usuarios, nos centraremos en la situación de mayor dispersión de datos, en donde los algoritmos se comportan peor. En cualquier caso, una evaluación *online* también carecería de todos los datos necesarios para poder realizar una evaluación completamente fiable que permitiese dilucidar cuál es la causa real de tales errores.

Lo que se puede concluir de lo dicho hasta ahora es que las técnicas no difieren tanto en su precisión cuando existe información suficiente, como en la evolución conforme la densidad de la matriz de puntuaciones cambia. Esto es debido a que los algoritmos extraen el conocimiento de distintas maneras a partir de la información disponible, lo cual provoca que la precisión obtenida en un situación con datos muy dispersos varíe mucho de una solución a otra.

Las pruebas en este tipo de entornos de baja densidad son buenos indicadores de la habilidad de los algoritmos para extraer información de la matriz de puntuaciones. Como ya se ha dicho, hemos usado una estrategia Given-N para realizar dicha evaluación. Los resultados para los *datasets* de MovieLens y Netflix se presentan en las Figuras 3.7 y 3.8, respectivamente. Para mayor claridad, sólo se muestran los resultados para los mejores algoritmos.

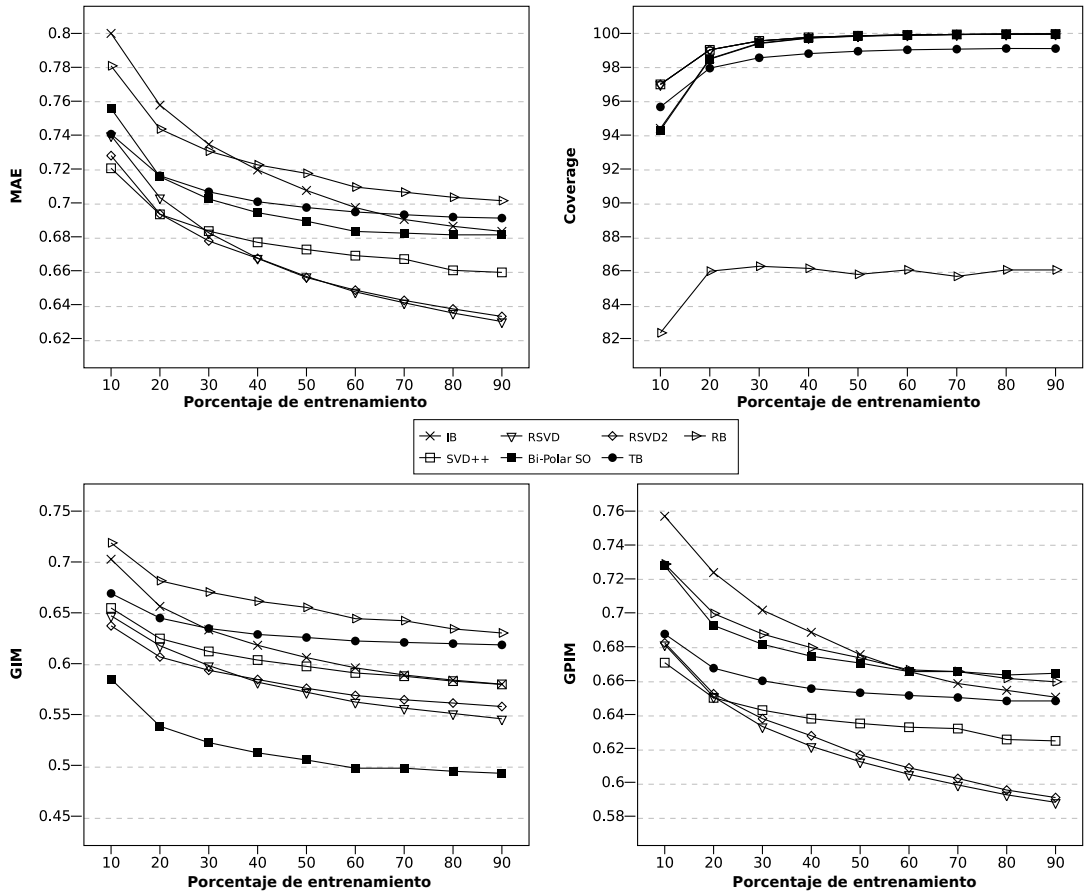


Figura 3.6: Evolución del *coverage* y la precisión según la densidad de la matriz, en el *dataset* de Netflix.

Si comparamos las Figuras 3.3, 3.6, 3.7 y 3.8, podemos ver cómo la evolución de la precisión, conforme la densidad de la matriz de puntuaciones aumenta, es distinta para cada algoritmo. Dicha evolución caracteriza mejor a una técnica que cualquier medida de precisión bajo condiciones específicas. Así, cuanto mayor sea su precisión, lo que diferencia a los algoritmos entre sí es su capacidad para extraer información.

3.6.2. Algoritmos basados en modelo frente a algoritmos basados en memoria

Los algoritmos basados en memoria presentan buenos resultados cuando la matriz de puntuaciones es relativamente densa, pero su precisión decrece considerablemente cuando la matriz es dispersa, en situaciones de *sparsity*. Esto es debido a que este tipo de algoritmos obtienen información bien mediante las relaciones entre usuarios, bien

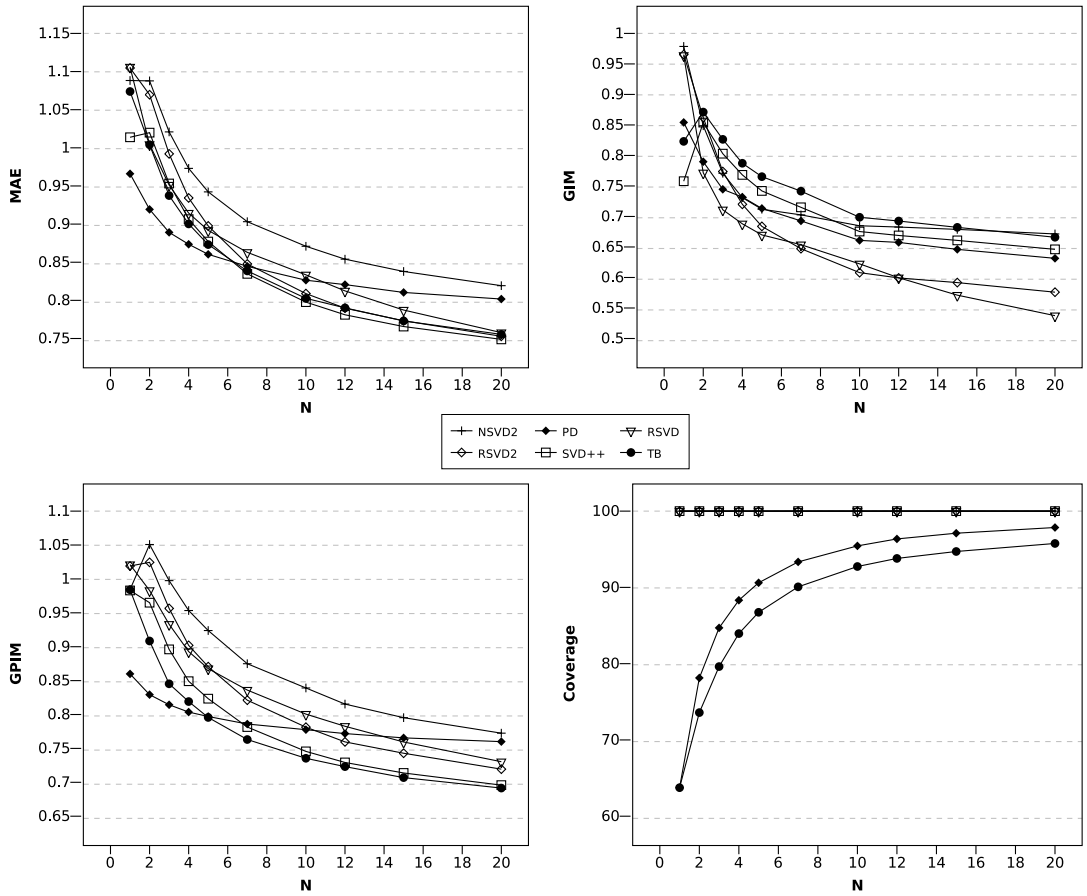


Figura 3.7: Precisión y *coverage* de los distintos algoritmos bajo condiciones de dispersión, con una estrategia Given-N, en el *dataset* de MovieLens.

mediante las relaciones entre productos, por lo que usan sólo una pequeña parte de la información disponible. Cuando la densidad de datos en la matriz de puntuaciones es baja, extraer información usando sólo un tipo de relaciones se convierte en algo complicado, lo cual provoca un empeoramiento notable de la precisión. Además, en estas situaciones estos algoritmos son capaces de realizar predicciones para una menor cantidad de productos (el *coverage* es más bajo), por lo que es fácil relacionar los malos resultados en precisión con la falta de información. Sin embargo, el algoritmo *Similarity Fusion*, a pesar de ser basado en memoria, es capaz de extraer información tanto de las relaciones entre usuarios como de la relaciones entre productos. Por tanto, esto repercute en sus resultados (tanto precisión como *coverage*), que mejoran en situaciones de *sparsity*, ya que es capaz de aprovechar mejor la información disponible.

Los algoritmos basados en modelo, como ya habíamos comentado, surgieron para

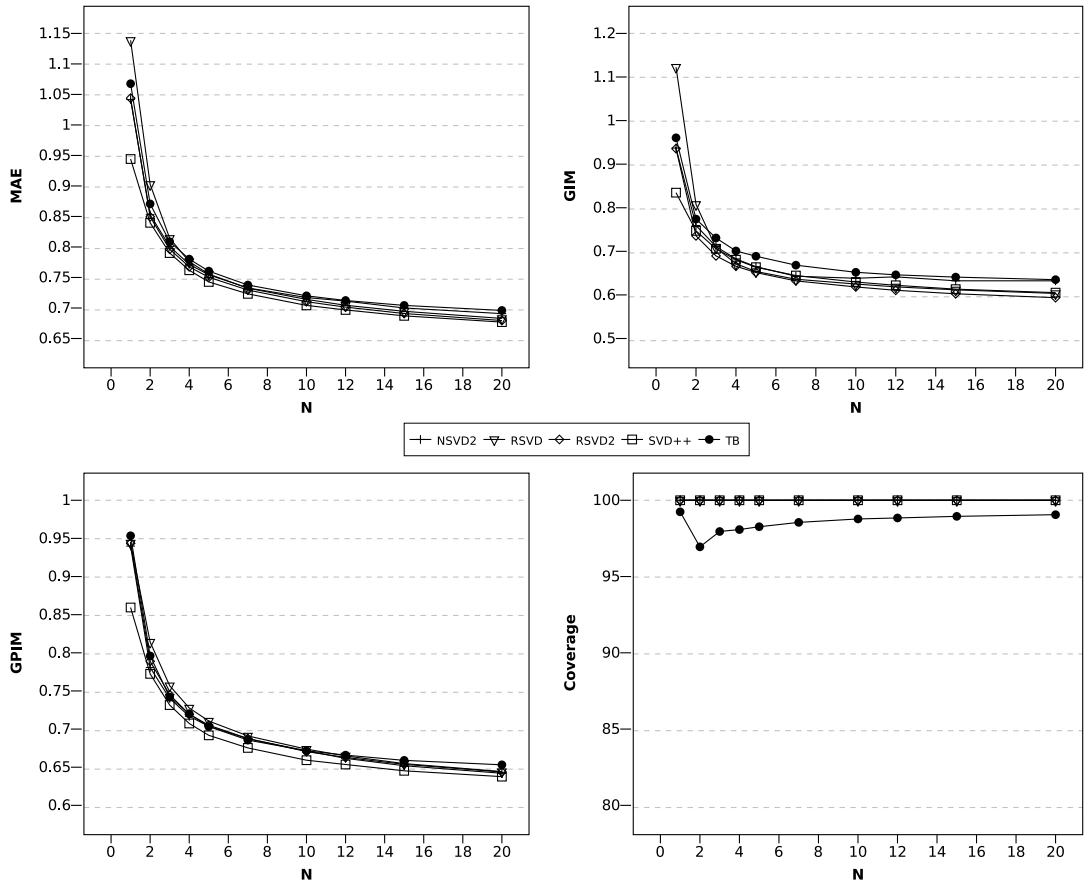


Figura 3.8: Precisión y *coverage* de los distintos algoritmos bajo condiciones de dispersión, con una estrategia Given-N, en el *dataset* de Netflix.

intentar solucionar los problemas de los basados en memoria. Uno de estos problemas es la situación de *sparsity*, donde son una buena alternativa. Son menos sensibles a los cambios de densidad en los datos, lo cual implica una evolución mucho más suave. Esto es debido a que obtienen gran cantidad de información en la fase de construcción del modelo. Parte de dicha información es difícil de obtener usando métodos tradicionales basados en memoria. Sin embargo, la propia construcción del modelo hace que se descarte información, provocando que estos algoritmos sean menos precisos que los basados en memoria cuando la información disponible es suficientemente densa. Este descarte puede realizarse conscientemente, con el objetivo de que el algoritmo sea escalable y más eficiente, o bien puede ser debido a que el modelo no es capaz de explicar esos aspectos de la información. Lógicamente, la precisión del algoritmo dependerá de lo bien que se ajuste a los datos reales. Técnicas que funcionan bien en un determinado

dataset pueden ser desaconsejables para otro.

Los algoritmos basados en SVD han mostrado los mejores resultados entre los basados en modelo. De hecho, las técnicas regularizadas sobresalen por encima de los demás en muchos casos y en ambos *datasets*. En realidad, sólo el algoritmo basado en tendencias puede competir con ellos en todos los casos, aunque atendiendo a la precisión algunos basados en memoria alcancen mejores resultados con alta densidad de datos.

Además, los resultados con las métricas de clasificación muestran que sus recomendaciones son también precisas, como se observa en la Figura 3.9. Todos los algoritmos alcanzan resultados más precisos con el *dataset* de Netflix como era de esperar, dado que dicho *dataset* contiene más datos de puntuaciones. Los algoritmos que presentan los mejores resultados con estas métricas son las técnicas SVD, el algoritmo basado en tendencias y *Slope One* (aunque su precisión no sea demasiado destacable).

Por su parte, el algoritmo *Cluster-based Smoothing* obtuvo los peores resultados. La clasificación de los usuarios en *clusters* no aporta nada a la media del producto. De hecho, los resultados muestran que la media global es una mejor predicción que la media del *cluster* con los usuarios más cercanos. Como se puede ver en la Figura 3.10, conforme aumentamos el número de *clusters* tanto la precisión como el *coverage* disminuyen. La razón es que este algoritmo se basa en la idea de agrupar a los usuarios en *clusters* según sus gustos o intereses. Sin embargo, en un dominio como el de la recomendación de películas, los gustos de los usuarios son muy diferentes entre sí por lo que es muy difícil encontrar un grupo con los mismos intereses. Esto provoca que los resultados usando este tipo de algoritmos no sean suficientemente precisos.

La misma conclusión se puede obtener del comportamiento de los métodos de selección de vecinos en los algoritmos basados en usuario. Como se comentó previamente, estos algoritmos tienen una fase en la que se seleccionan un conjunto de usuarios a partir de los cuales se calculará la predicción. La idea es tener en cuenta usuarios con gustos e intereses lo más similares posible a los del usuario activo: sus vecinos. Tradicionalmente, se han usado dos estrategias: el máximo número de vecinos (la mejor estrategia tanto en [Herlocker et al., 2002] como en nuestros experimentos) y el umbral de correlación. Esto implica que es necesario calcular las similitudes del usuario activo con el resto de usuarios. El máximo número de vecinos, N, determina el tamaño del vecindario. Parece lógico pensar que cuanto más bajo sea N, mejores serán los resultados obtenidos, ya que la predicción estará basada en los usuarios más parecidos al usuario activo. Sin embargo, como se puede observar en la Figura 3.11, cuando el valor de N es inferior a 20, la precisión empeora considerablemente. Los resultados que se obtienen

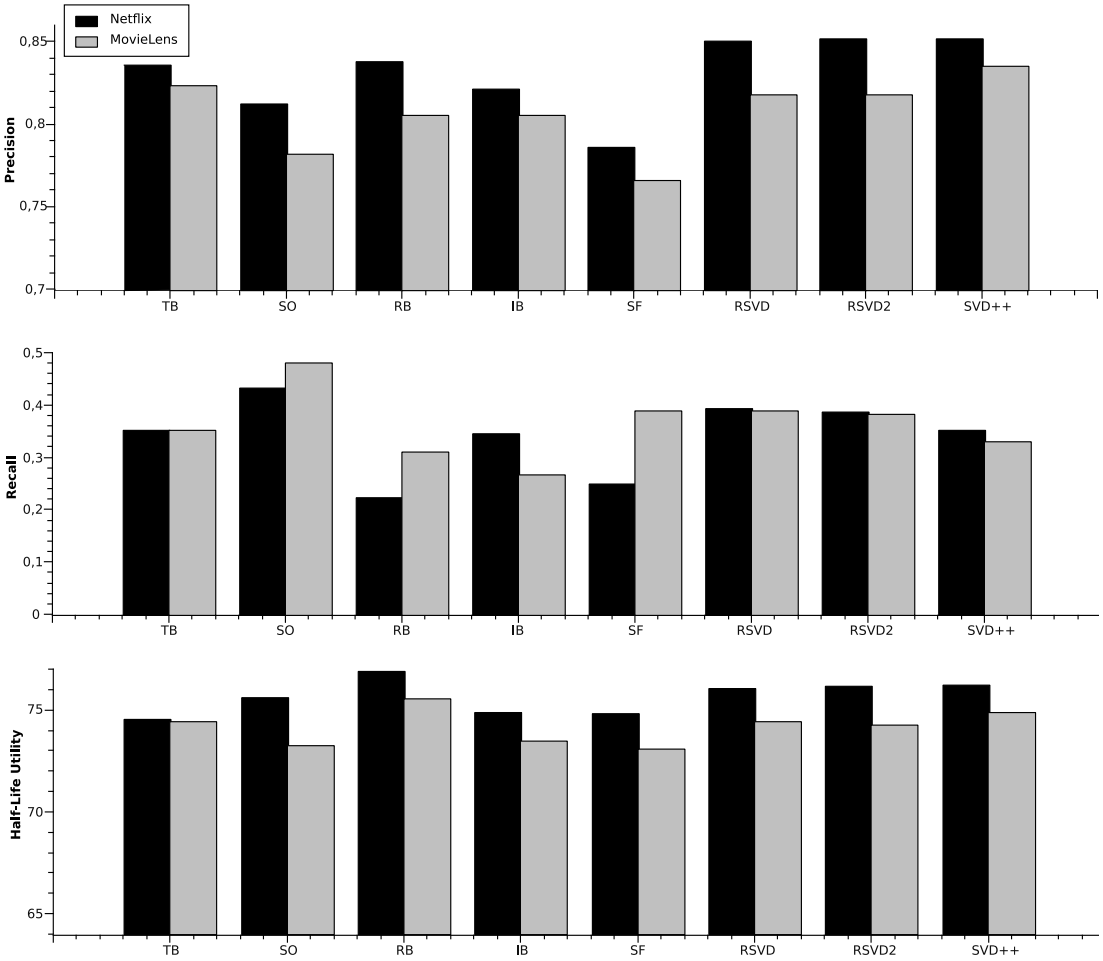


Figura 3.9: Resultados de las métricas de clasificación para los *datasets* de MovieLens y Netflix, con un subconjunto de entrenamiento del 50 %.

con 50 o 100 vecinos son mucho mejores que los obtenidos con 5 o 10, independientemente de la medida de similitud seleccionada. Este comportamiento está relacionado claramente con la dificultad de encontrar dos usuarios con los mismos gustos. Usuarios cuyos gustos coincidan en un buen número de productos, pueden tener opiniones muy divergentes sobre otros.

Este problema también se observa en la segunda estrategia, en donde se seleccionan usuarios con similitudes que sobrepasen un determinado umbral. Como se muestra en la Figura 3.12, conforme aumentamos este umbral (la predicción se basa en los usuarios más similares) la precisión empeora. De hecho, los casos de aparente mejora con altos umbrales (MSD y Coseno), son debidos a un bajo *coverage*, tal y como también se apunta en [Herlocker et al., 2002]. En general, la dificultad de encontrar usuarios

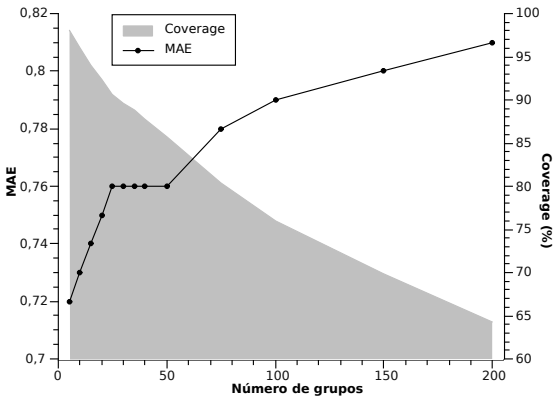


Figura 3.10: Evolución de la precisión y el *coverage* según el número de *clusters*.

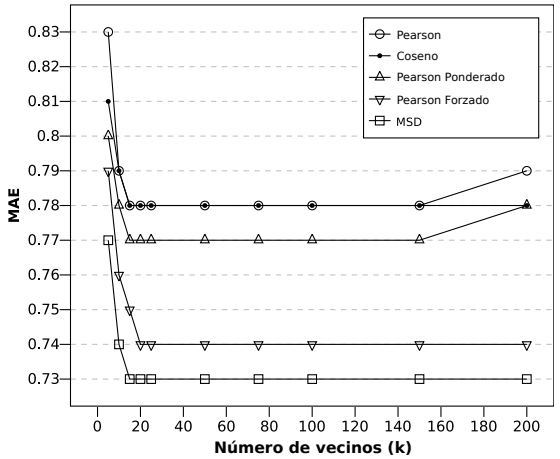


Figura 3.11: Precisión según el número de vecinos, con un subconjunto de entrenamiento del 80 %.

muy correlados provoca el decrecimiento del *coverage* conforme aumenta el umbral de correlación. Este resultado refuerza la idea de la dificultad de encontrar usuarios con gustos similares.

También es interesante observar el efecto de la normalización (z-score) de las puntuaciones de cada vecino antes de calcular la predicción. En todos los casos, la normalización mejora sustancialmente los resultados, como se observa en la Figura 3.13. De hecho, especialmente bajo condiciones de alta densidad, los algoritmos que usan la normalización son claramente mejores que los que no la usan, independientemente de la medida de similitud empleada: el peor algoritmo con normalización presenta mejores resultados que el mejor sin normalización [Herlocker et al., 2002].

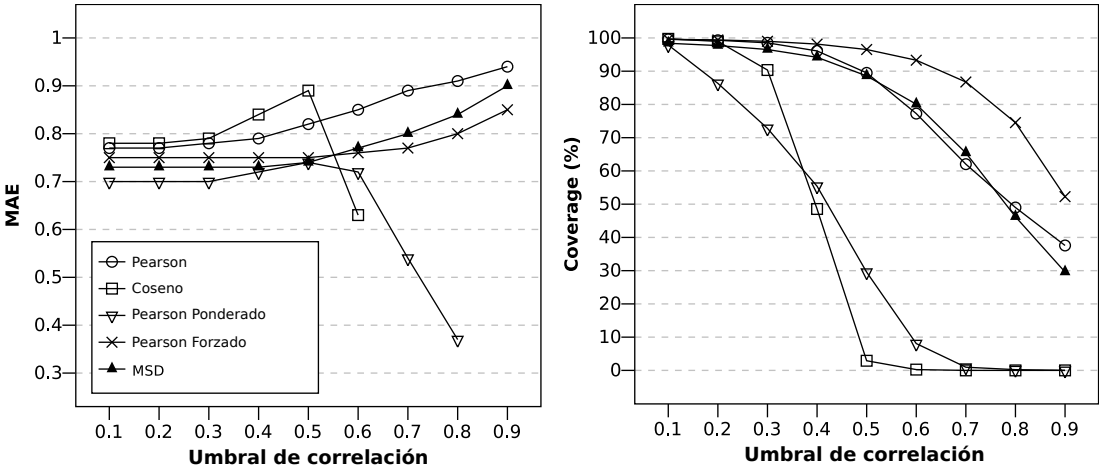


Figura 3.12: Precisión y *coverage* según el umbral de correlación, con un subconjunto de entrenamiento del 80 %.

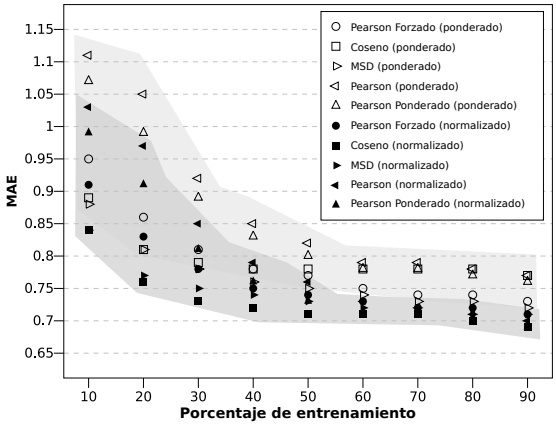


Figura 3.13: Efecto de la normalización (Z-Score) sobre la precisión de los algoritmos basados en usuario.

3.7. Conclusiones

En este capítulo hemos realizado una comparación entre diferentes algoritmos de filtrado colaborativo, para observar el comportamiento de los algoritmos bajo diversas circunstancias, no sólo bajo las condiciones más favorables.

Los resultados muestran la gran influencia de la densidad de la matriz de puntuaciones en la precisión de los algoritmos, que puede deferir dependiendo del tipo de algoritmo. Hemos visto cómo los algoritmos basados en memoria ofrecen un buen comportamiento con matrices relativamente densas, empeorando significativamente con la ausencia de información; y hemos observado cómo los algoritmos basados en modelo,

sin llegar a la precisión de los basados en memoria en condiciones óptimas, se comportan mejor en condiciones adversas. Las pruebas realizadas nos permiten relacionar este comportamiento con la habilidad de los distintos algoritmos de interpretar los datos disponibles y extraer información útil. La mayoría de los algoritmos se comportan de forma similar con los dos *datasets* estudiados, lo cual es normal dado que se trata de datos del mismo dominio.

Hemos mostrado que la mayor parte de las dificultades que tienen los algoritmos en la extracción de información está relacionada con las limitaciones de las técnicas a la hora de encontrar las similitudes entre usuarios y/o productos. Dos posibles explicaciones para estas limitaciones son la gran diversidad en gustos y opiniones, y la dificultad de encontrar tales similitudes en condiciones de baja densidad de datos *sparsity*.

En el futuro, tanto la inclusión de nuevos algoritmos como la realización del estudio en nuevos dominios pueden ser un trabajo interesante.

Capítulo 4

Mejora de la tarea de predicción en sistemas dispersos

En los últimos años los sistemas recomendadores, en general, y los algoritmos de filtrado colaborativo, en particular, han alcanzado una gran popularidad. Se han desarrollado y aplicado gran cantidad de nuevas técnicas. Sin embargo, en muchos casos los algoritmos no obtienen los resultados esperados. En concreto, cuando el modelo aplicado no se ajusta a los datos reales los resultados son especialmente malos. Esto sucede porque con frecuencia los modelos se aplican directamente a un dominio sin un análisis previo de los datos. Además, estos malos resultados se agravan cuando los datos son especialmente dispersos, como hemos visto en el Capítulo 3.

En este capítulo estudiaremos detenidamente los conjuntos de datos más populares en el dominio de la recomendación de películas, con el objetivo de comprender cómo se comportan los usuarios en este contexto particular. Una vez analizados los datos, buscaremos aplicar las lecciones aprendidas a la mejora de los algoritmos de filtrado colaborativo. Particularmente, nos centraremos en los algoritmos kNN por ser algoritmos intuitivos, simples y eficientes. Veremos que las medidas de similitud usadas tradicionalmente para este tipo de algoritmos pueden ser cuestionados, y propondremos nuevas medidas de similitud especialmente apropiadas para contextos dispersos, poniendo especial atención en los usuarios que han valorado muy pocos productos (situación de *cold-start*).

4.1. Introducción

Los algoritmos de filtrado colaborativo han sido objeto de numerosos estudios en los últimos años. En concreto, la comunidad científica ha centrado su esfuerzo en la mejora

de estos algoritmos en la tarea de precisión o *annotation in context*, en donde el sistema predice la utilidad que un producto dado tiene para un usuario. Los algoritmos de filtrado colaborativo procesan la información acerca de los gustos de los usuarios, expresada habitualmente en forma de puntuaciones, para ofrecer recomendaciones personalizadas.

En la literatura podemos encontrar muchos ejemplos de este tipo de algoritmos. Aunque muchos obtienen muy buenos resultados, es también verdad que a menudo se desarrollan basándose en ideas o modelos aplicados en otros dominios, sin prestar atención a los datos reales. Muchas veces, es simplemente la evaluación de los resultados lo que permite a los desarrolladores saber si el algoritmo se ajusta o no a los datos. En ese caso, problemas en la metodología de evaluación o en las métricas pueden conducir a conclusiones erróneas.

Como ya hemos visto en el Capítulo 3 estos algoritmos se clasifican generalmente en función de cómo procesan la matriz de puntuaciones en algoritmos basados en memoria y algoritmos basados en modelo. Los algoritmos kNN basados en usuarios [Resnick et al., 1994; Shardanand, 1994], como cualquier otra técnica basada en memoria, necesitan procesar la matriz de puntuaciones completa para predecir una puntuación. Por tanto, presentan problemas de escalabilidad, aunque generalmente alcanzan resultados suficientemente precisos. Además, también se ven afectados por la dispersión de los datos [Huang et al., 2004b; Sarwar et al., 2001], cold-start [Schein et al., 2002] y los problemas de shilling [Chirita et al., 2005; Lam y Riedl, 2004], como muchas otras aproximaciones de sistemas de recomendación. Sin embargo, se trata de algoritmos fáciles de implementar; sus recomendaciones son fácilmente justificables debido a similitudes con los usuarios vecinos; son eficientes, ya que los vecinos se pueden computar *offline*; y son estables ante la inserción de nuevos usuarios y/o productos en el sistema [Desrosiers y Karypis, 2011].

Algunos autores han intentando solucionar estos problemas con técnicas como los algoritmos basados en modelo. Estas técnicas extraen más información de los datos disponibles, pero no solucionan completamente los problemas mencionados. De hecho, introducen algunos nuevos [Cacheda et al., 2011a]: el sobreajuste, no siempre se adaptan a los datos reales, a veces requieren configurar demasiados parámetros... Estos problemas conducen a malas predicciones. Una de las razones por la que esto sucede es que a menudo un modelo se aplica directamente a un dominio dado, en lugar de analizar los datos e intentar desarrollar un modelo que represente adecuadamente los datos.

Por ejemplo, en contextos como la recomendación de películas, sin un estudio previo de los datos, parece razonable pensar usar un algoritmo basado en *cluster*. Sin embargo, esta técnica en realidad obtiene malos resultados [Cacheda et al., 2011a]. Un estudio

previo de los datos podría haber mostrado que en este contexto no es una buena idea agrupar a los usuarios en *clusters* disjuntos.

No obstante, hasta donde nosotros sabemos, existen muy pocos trabajos que realmente desarrollen un algoritmo basándose en un análisis previo de los datos. En la mayoría de los trabajos, los autores analizan los datos simplemente para dar un contexto del dominio: algunos de ellos se centran en una pequeña parte del conjunto de datos completo [Shardanand, 1994], otros estudian simplemente una pequeña parte de sus características [Canny, 2002b; Wang, 2008], etc.

Por otro lado, algunos casos concretos han mostrado la utilidad del análisis de los conjuntos de datos. Por ejemplo, en algunos contextos las puntuaciones no están disponibles y deben ser inferidas a partir del comportamiento de los usuarios [Hahsler, 2010], así que se necesita un análisis previo para poder comprender cómo interactúan los usuarios con el sistema. Además, dado que la comunidad está cada vez más interesada en introducir los aspectos de tiempo para dar recomendaciones más precisas, trabajos recientes han descrito la evolución de las puntuaciones, los usuarios y los productos con el tiempo [Koren, 2010b; Lathia et al., 2010; Liu et al., 2010]. A veces estos aspectos son tenidos en cuenta para desarrollar nuevos algoritmos.

Este capítulo está basado en el trabajo que presentamos en [Cacheda et al., 2011b]. En él consideramos una aproximación a partir del estudio de los datos para diseñar nuestro algoritmo. Comenzaremos indicando las principales contribuciones del estado del arte al problema de *cold-start* que nos atañe. Después, analizaremos los *datasets* más populares en el dominio de la recomendación de películas, para comprender el comportamiento del usuario. Analizamos varios de los puntos débiles de los algoritmos basados en usuario tradicionales, e intentamos descubrir, basándonos en la información que está disponible en esos conjuntos de datos, cómo superar esas limitaciones. Presentamos nuestro estudio en la Sección 4.3. Con las conclusiones obtenidas a partir de dicho estudio, en la Sección 4.4 proponemos algunas mejoras a los algoritmos kNN tradicionales atendiendo a condiciones de baja densidad de datos, es decir, donde hay pocos datos disponibles. En particular, presentamos una nueva medida para calcular la similitud entre usuarios, que proporciona valores fiables incluso con usuarios con pocas puntuaciones. También abordamos el uso de *inverse user frequency* como un modo de dar diferente importancia a cada producto en función de su popularidad. En la Sección 4.5 evaluamos nuestra aproximación. La importancia de realizar un análisis previo del conjunto de datos antes del diseño de cualquier algoritmo se discute en la Sección 4.6, junto con varias mejoras que no han sido evaluadas todavía y que serán abordadas en futuros trabajos.

4.2. Cold-start

Los algoritmos de filtrado colaborativo funcionan bien cuando disponen de muchas puntuaciones. Sin embargo, a veces la información sobre un usuario particular (problema del nuevo usuario), un producto (problema del nuevo producto) o, incluso, la información de todo el sistema (problema del nuevo sistema) es realmente escasa. Estos tres casos suponen los tres problemas de *cold-start*, ya introducidos en la Sección 2.5.

En general, estas situaciones no pueden ser manejadas adecuadamente por las técnicas de filtrado colaborativo, por lo que en la literatura se han propuesto diferentes técnicas para abordar cada una de dichas situaciones, muchas de ellas híbridas, en las que se considera información adicional, aparte de las puntuaciones [Balabanović y Shoham, 1997].

El problema del nuevo producto se puede tratar combinando las técnicas colaborativas con información de contenido. Un paradigma ampliamente extendido es el llamado *collaboration via content*, que consiste en explotar perfiles basados en contenido para detectar similitudes entre usuarios [Pazzani, 1999]. La información de contenido puede ser usada para estimar las puntuaciones y mejorar el perfil del usuario. Por ejemplo, en [Melville et al., 2002] se usa un clasificador de texto naïve bayessiano extendido para estimar las puntuaciones usando características de las películas. Una vez estimadas, el algoritmo de filtrado colaborativo emplea el perfil completo. Otros autores han estudiado la utilidad de los bots, es decir, usuarios ficticios que puntúan automáticamente productos de acuerdo a información de contenido. Por ejemplo, en [Good et al., 1999] los autores experimentaron con diversas técnicas de bots, desde aproximaciones simples como bots que puntuaban películas de acuerdo a su género, hasta aproximaciones más complejas basadas en TF-IDF o lógica inductiva. Además, combinaron varias de dichas técnicas, obteniendo buenos resultados. Más tarde, en [Park et al., 2006] se mejoró la escalabilidad y el rendimiento de esta idea, proponiendo un número reducido de bots de filtrado globales.

También se han diseñado sistemas en donde la información de contenido es parte del modelo. Schein *et al.* modelaron los datos de contenido por medio de un modelo de aspectos de factor latente que relaciona usuarios con el reparto de las películas, además de un modelo de aspectos puramente colaborativo que relaciona usuarios con películas [Schein et al., 2002]. Se propuso también otra solución basada en máquinas Boltzman para modelar las relaciones entre usuarios y productos, usando información de contenido para estimar los parámetros del modelo [Gunawardana y Meek, 2008]. Modelaron el contenido como vectores de características, donde una característica podía representar

la presencia de un actor, el peso de un término, etc. Otras técnicas también han sido estudiadas [Claypool et al., 1999; Leung et al., 2007; Park y Chu, 2009].

En esta tesis nos centraremos en el problema del nuevo usuario. En la literatura existen diferentes aproximaciones para abordarlo. Se pueden aplicar sistemas híbridos, como los anteriormente mencionados. En este caso, se han estudiado también soluciones con información demográfica [Lam et al., 2008; Nguyen et al., 2007], o relaciones de confianza en una red social [Papagelis et al., 2005a]. En general, se pueden dividir básicamente en aquéllas que involucran al usuario para que proporcione información adicional al sistema (siendo éstas bastante habituales), y aquéllas que no conllevan la intervención del usuario.

El modo más sencillo para obtener información adicional acerca del usuario es preguntándole directamente. Este proceso de recolección de las puntuaciones iniciales, conocido con el nombre de proceso de elicitación [Golbandi et al., 2011; Rashid et al., 2008; Rokach y Kisilevich, 2012], puede significar el éxito o el fracaso a la hora de que el usuario permanezca en el sistema [Elahi et al., 2011]. Los productos iniciales deben ser cuidadosamente escogidos y el número de ellos no debería ser excesivo para evitar molestar a los usuarios [Cremonesi et al., 2012]. A cambio, los usuarios esperar recibir buenas recomendaciones. De este modo, el usuario tendrá al menos el número de puntuaciones que el sistema solicite cuando el usuario se registre.

Sin embargo, a veces no es adecuado optar por preguntarle directamente al usuario. Por ejemplo, estas peticiones pueden incordiar a algunos usuarios que no consideren necesario valorar productos cuando acaban de llegar al sistema. Cuando un usuario tiene pocas puntuaciones, la recomendación es poco precisa por normal general e, incluso, el sistema puede ser incapaz de calcularla. A pesar de ello, dependiendo del contexto, la mejor opción puede ser no mostrar recomendaciones que puedan molestar a los usuarios. De cualquier modo, si la información es escasa pero el sistema puede proporcionarle al usuario una recomendación suficientemente precisa para que éste permanezca en el sistema, el proceso de elicitación puede ser innecesario.

Son preferibles otras alternativas sin la intervención del usuario. Algunos autores estudiaron cómo modificar los algoritmos kNN para mejorar las recomendaciones. Esto es debido a que, como veremos, las medidas de similitud tradicionales presentan problemas cuando la información es escasa. Es por ello que se han presentado medidas alternativas: una basada en proximidad, impacto y popularidad [Ahn, 2008]; otra basada en la combinación de hasta seis medidas diferentes [Bobadilla et al., 2012]; o una medida de similitud basada en probabilidad [Piccart et al., 2010].

4.3. Análisis de dos *datasets*

En el capítulo anterior realizamos una breve descripción de dos de los *datasets* más empleados en filtrado colaborativo: Netflix y Movielens. En esta ocasión, profundizamos en el análisis de Netflix y empleamos Movielens 10M, que incorpora mayor número de puntuaciones que el *dataset* empleado en el capítulo anterior. El primero, como ya comentamos, contiene contiene más de 100 millones de puntuaciones de 480,189 usuarios a 17,770 películas, reunidas entre octubre de 1998 y diciembre de 2005. El segundo contiene más de 10 millones de puntuaciones sobre 10,681 películas, realizados por 71,567 usuarios del servicio de recomendación de películas *online* de Movielens[1] entre enero de 1995 y enero de 2009.

4.3.1. Limitaciones de las medidas de similitud tradicionales

Un paso importante en los algoritmos kNN es encontrar el vecindario, es decir, el conjunto de usuarios más parecidos al usuario actual. Una medida de similitud como la correlación de Pearson se usa para calcular el grado de similitud entre cada par de usuarios. Teóricamente, emplear los usuarios más parecidos es la mejor forma de predecir la puntuación del usuario. Sin embargo, hemos demostrado en el capítulo anterior que usar pocos vecinos y muy correlados en realidad empeora la precisión del algoritmo, lo cual es un resultado que *a priori* sería inesperado. Este comportamiento lo asociamos con la dificultad de encontrar usuarios altamente correlados debido a la falta de información. En realidad, pensamos que en el dominio de la recomendación de películas no suelen aparecer frecuentemente las correlaciones fuertes. Esto es debido a que los gustos de los usuarios son generalmente diferentes: el hecho de que a dos usuarios les gusten las mismas películas no implica que estén de acuerdo en todas ellas. Por ejemplo, a dos usuarios les pueden gustar las películas de acción, pero mientras que a uno le gustan también las de terror, el otro prefiere las películas del oeste. Esto es muy común en el dominio que nos atañe.

Para estudiar este hecho, hemos analizado alrededor de 25,000 relaciones usuario-usuario aleatoriamente escogidas del conjunto de datos Movielens 10M. En la Figura 4.1, se muestra el histograma para la correlación de Pearson. Se puede ver que la mayoría de los usuarios presentan una correlación muy baja, alrededor de 0.0. Esto es lo esperado, y prueba que la mayor parte de los usuarios tienen diferentes gustos, así que no están en absoluto correlados. Sin embargo, muchos usuarios presentan también una fuerte correlación (alrededor de -1.0 y 1.0), lo cual puede parecer sorprendente

[1] www.movielens.org

y aparentemente contradice la afirmación previa que aseguraba que las correlaciones fuertes no existen en los dominios de películas. No obstante, si echamos un vistazo a la Figura 4.2 podemos comprobar que las correlaciones fuertes sólo se alcanzan entre usuarios que han valorado muy pocas películas en común. En este caso, por supuesto, la similitud calculada no refleja realmente la similitud entre los usuarios. Por ejemplo, si el usuario anterior al que le gustaban las películas de acción y de terror hubiese valorado solamente las primeras, probablemente parecería muy similar al otro usuario, cuando en realidad no lo es. Como caso paradigmático, con la correlación de Pearson siempre se obtiene una correlación perfecta cuando dos usuarios han valorado solamente un único producto en común.

Para evitar este problema, algunos autores han propuesto ponderar la correlación de acuerdo al número de productos en los que se basa. Una aproximación conocida es la correlación de Pearson ponderada (ver Sección 3.3.1). Esta técnica mitiga el problema mencionado eliminando las correlaciones fuertes incorrectas. Sin embargo, el efecto real es que los usuarios fuertemente correlados desaparecen, como se puede ver en la Figura 4.3. Así, el problema no ha sido resuelto todavía.

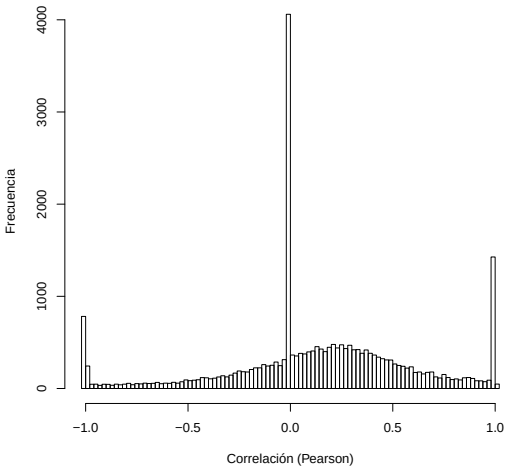


Figura 4.1: Histogramas para la correlación de Pearson.

Existen dos posibles explicaciones al citado problema. Primero, la falta de información. Dado que la matriz de puntuaciones es demasiado dispersa, puede estar sucediendo que no se encuentren los usuarios correlados simplemente porque han valorado muy pocos productos en común. Esto se puede ver en la Figura 4.4: la mayoría de los usuarios sólo han valorado un porcentaje bajo de productos en común. Por otro lado, también

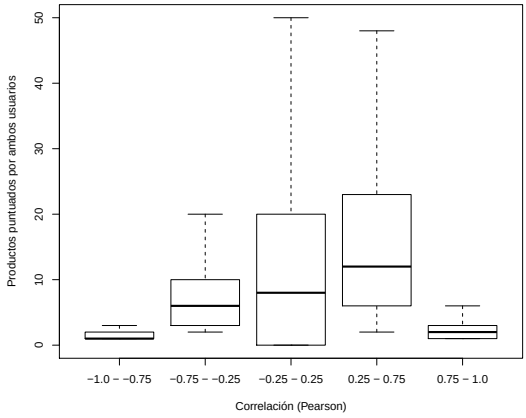


Figura 4.2: Número de productos valorados por ambos usuarios, según la correlación de Pearson. Se ocultan los datos atípicos para mayor claridad.

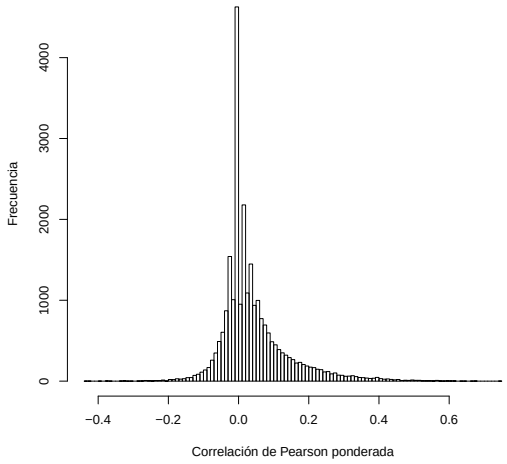


Figura 4.3: Histogramas para las correlaciones de Pearson ponderadas.

está relacionado con las diferencias entre los usuarios. Dos usuarios con diferentes gustos verán (y valorarán) probablemente películas distintas. Además, como se muestra en la Figura 4.5, para muchos pares de usuarios que han valorado un elevado número de productos, el porcentaje de productos que ambos han puntuado es también bajo. En este caso, puede estar relacionado con el hecho de que a ambos les guste un mismo género, director, etc. en particular, pero que discrepen en muchos otros. Otras veces, esas películas valoradas en común pueden ser películas muy populares.

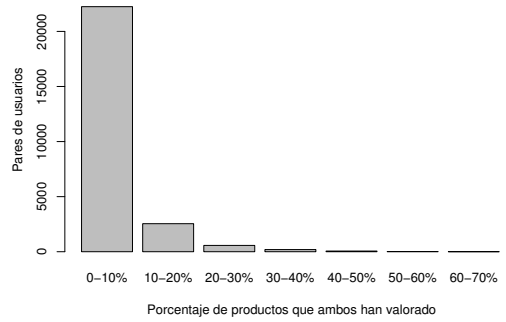


Figura 4.4: Número de pares usuario-usuario, según el porcentaje de productos que han valorado ambos (con respecto al número total de productos valorados).

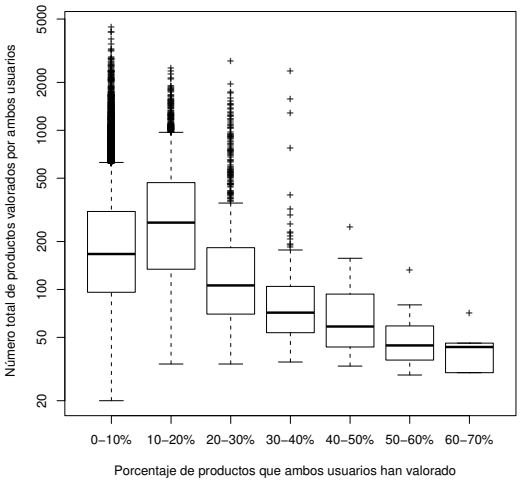


Figura 4.5: Número total de productos valorados por ambos usuarios, según el porcentaje de productos que ambos han valorado.

Parece lógico pensar que las aproximaciones basadas en productos son una mejor alternativa desde el punto de vista de los datos, porque las diferencias naturales entre los usuarios relacionadas con la naturaleza humana y la variedad en los gustos no están necesariamente presentes en los productos. Sin embargo, como se puede ver en la Figura 4.6, las correlaciones entre productos tienen el mismo problema.

Por tanto, el estudio de los datos parece confirmar que las medidas de similitud tradicionales basadas en correlaciones no siempre son la mejor alternativa. En particular, cuanto más dispersos son los datos, peor se comportan estas medidas. Estas observacio-

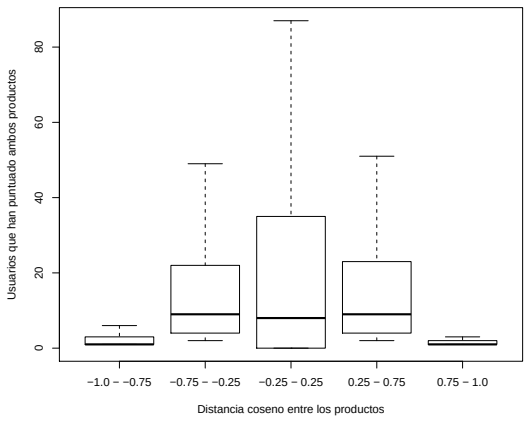


Figura 4.6: Usuarios que han valorado ambos productos según la distancia coseno.

nes nos conducen a pensar que aproximaciones menos estrictas a la hora de relacionar a los usuarios pueden obtener mejor precisión.

4.3.2. Productos populares

Como algunos autores han mostrado [Cremonesi et al., 2010], la mayoría de las puntuaciones en los conjuntos de datos comerciales se condensan en un pequeño número de productos populares. En particular, tanto en el conjunto de datos de Movielens 10M como en el de Netflix, el 20 % de las puntuaciones pertenecen aproximadamente un 2 % de los productos solamente. Normalmente, no es útil recomendar estos productos populares, ya que probablemente el usuario ya los conozca. Esta observación no se tiene en cuenta cuando se está tratando la tarea de predicción, porque el sistema tendrá que predecir una puntuación para estas películas si el usuario lo solicita.

Sin embargo, estos productos populares pueden no ser tan útiles para calcular la similitud entre los usuarios. Parece lógico pensar que no todos los productos proporcionan la misma información sobre dicha similitud. Por ejemplo, una coincidencia en una película muy popular es menos importante que una coincidencia en una película antigua y difícil de encontrar. La razón es que las películas populares son consideradas positivamente por la mayoría de los usuarios, lo cual es lógico ya que una mala película difícilmente llega a ser popular. Así, muchos usuarios estarán de acuerdo en la puntuación acerca de esas películas, pero eso no significa que los usuarios sean realmente similares. Esto se puede comprobar si analizamos la distribución de las puntuaciones para productos populares y no populares. En la Figuras 4.7 y 4.8 aparece la distribu-

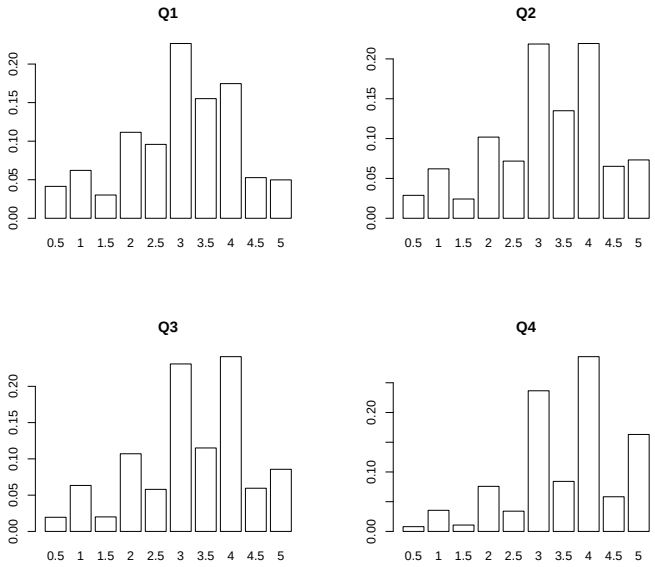


Figura 4.7: Distribución de puntuaciones para cada cuartil del número de puntuaciones, en el *dataset* de MovieLens 10M. Q1 son los productos con menos puntuaciones, mientras que Q4 son los productos más populares.

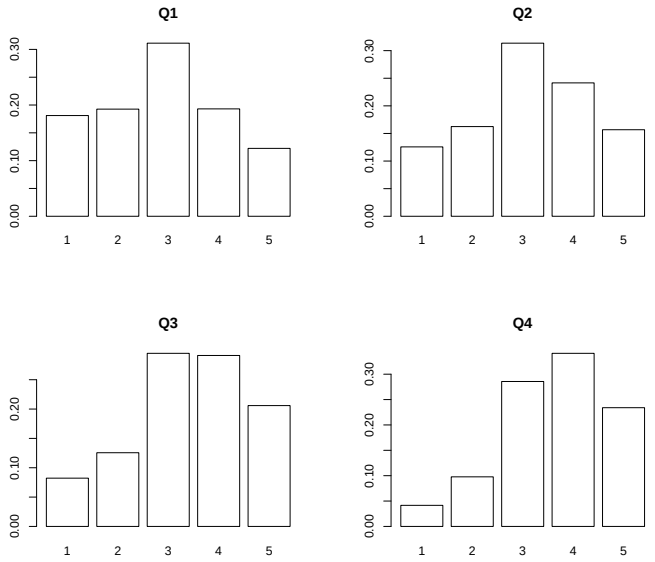


Figura 4.8: Distribución de puntuaciones para cada cuartil del número de puntuaciones, en el *dataset* de Netflix. Q1 son los productos con menos puntuaciones, mientras que Q4 son los productos más populares.

ción de las puntuaciones, clasificados según el número de usuarios que han valorado el producto. En particular, hemos usado los cuartiles del número de puntuaciones para

permitir la comparación entre los conjuntos de datos de Movielens 10M y Netflix. En ambos conjuntos de datos se puede ver que la mayoría de los productos del cuarto cuartil (Q4), que son los elementos más populares, recibieron buenas puntuaciones. Por otro lado, los productos del primer cuartil (Q1) recibieron también muchas puntuaciones malas. Es decir, la mayoría de los usuarios tienden a estar de acuerdo en su opinión acerca de las películas populares, pero muestran opiniones muy diferentes sobre las películas menos populares. Así, una coincidencia en un producto popular realmente no significa que dos usuarios sean similares, mientras que una coincidencia en un producto raro no popular puede significarlo. Las medidas de similitud deberían tener esto en cuenta. Sin embargo, que nosotros sepamos, no se ha llevado ningún estudio acerca de la utilidad de estos productos populares para calcular la similitud entre usuarios.

4.3.3. La importancia del problema de cold-start

Las medidas tradicionales no manejan demasiado bien los usuarios que han valorado pocos productos. Eso sucede, tanto con usuarios que acaban de empezar a usar el sistema como con usuarios que han valorado pocos productos, a pesar de haber estado usando el sistema durante mucho tiempo, bien sea porque no han usado demasiado el sistema, bien porque son reticentes a realizar valoraciones de los productos. Esto se conoce con el nombre del problema del cold-start, y más concretamente, el problema del nuevo usuario.

Esto es un problema especialmente serio porque muchos usuarios están en esta situación. Como se observa en la Figura 4.9, el 20 % de los usuarios sólo proporcionan el 2 % y el 3 % de las puntuaciones en los conjuntos de datos de Netflix y Movielens 10M, respectivamente. En tal situación, medidas de similitud tradicionales, como la correlación de Pearson, tendrán muy poca información con la que trabajar, lo cual conducirá a resultados imprecisos. Se necesita, por tanto, desarrollar un medida de similitud suficientemente simple para que se puedan obtener resultados razonablemente precisos para dichos usuarios.

4.4. Mejorando los algoritmos kNN

A partir del análisis del conjunto de datos que presentamos en la sección anterior se puede ver que:

- No todos los productos tienen la misma importancia a la hora de calcular la similitud entre usuarios. En particular, los productos populares suelen recibir

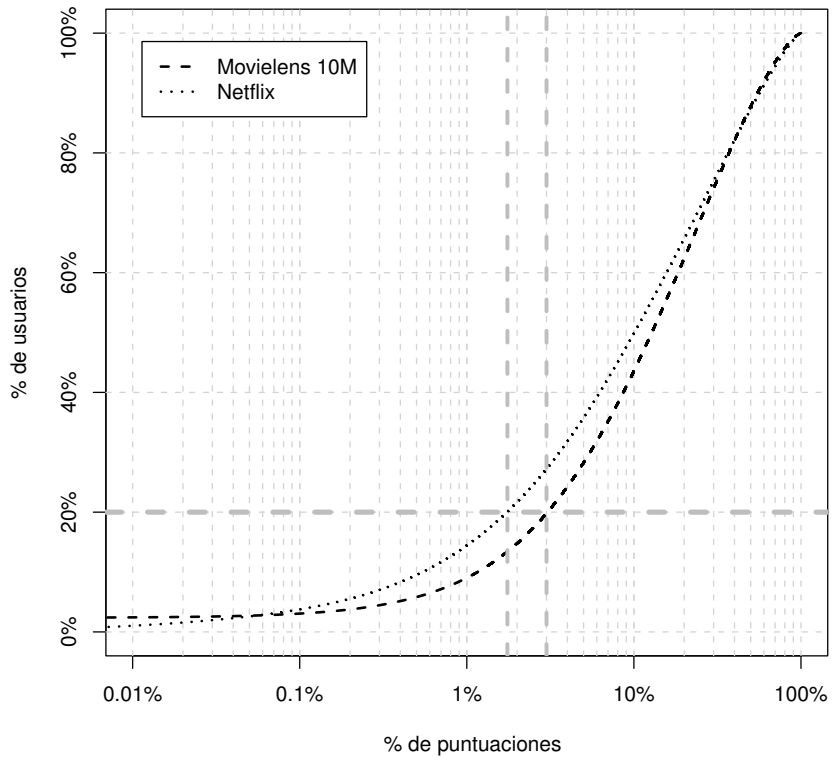


Figura 4.9: Distribución de las puntuaciones según el porcentaje de usuarios.

buenas puntuaciones, así que coincidencias en estos productos no son realmente un indicador de la concordancia en los gustos de los usuarios.

- Las medidas de similitud tradicionales, como la correlación de Pearson, son demasiado estrictas y conducen a resultados imprecisos porque la mayoría de los usuarios sólo puntúan un pequeño número de productos en común. Estas medidas intentan encontrar usuarios muy similares. Estos usuarios, como se ha comentado en la Sección 4.3.1, son muy difíciles de encontrar en los dominios de recomendación de películas. De hecho, aproximaciones más sencillas [Cacheda et al., 2011a; Lemire, 2003] alcanzan similares o incluso mejores precisiones, especialmente ante situaciones de baja densidad de datos, donde las aproximaciones basadas en correlaciones no se comportan nada bien. Esto es particularmente importante cuando nos enfrentamos al problema del *cold-start*.

Ya que las medidas de similitud tradicionales presentan varias limitaciones, proponemos un nueva medida de similitud para tener en cuenta los puntos que acabamos de mencionar. Esta nueva medida está fundamentada en dos criterios: los productos en común y las coincidencias en las puntuaciones.

El primer criterio se usa para tratar con el problema del *cold-start*, es decir, con usuarios con pocas puntuaciones. Las medidas tradicionales son muy estrictas y intentan calcular la similitud usando el valor numérico real de las puntuaciones. En este caso, los datos disponibles son demasiado dispersos, y, por tanto, usar los valores numéricos en estas condiciones no es apropiado, porque no hay suficientes productos en común para calcular de un modo adecuado este tipo de medida. En su lugar, proponemos tener en cuenta sólo los productos valorados en común por cada pareja de usuarios. Con las medidas tradicionales basadas en correlaciones, es probable que dos usuarios que compartan pocas puntuaciones presenten una alta similitud. Sin embargo, las coincidencias en las puntuaciones pueden ser simplemente debidas al azar (puede que no sean realmente usuarios similares). La popularidad de estas películas es una posible razón, por ejemplo. Teniendo en cuenta el número de productos valorados en común nuestra medida evita este problema.

De hecho, como se observa en la Figura 4.10, con esta medida se alcanza una gran similitud no sólo con usuarios que han valorado pocos o muy pocos productos en común. Ésta es una diferencia importante con respecto a la correlación de Pearson, y una de las principales ventajas de esta medida.

Conforme el número de productos en común aumenta, es posible empezar a usar técnicas más complejas además de esta medida, para así extraer tanta información como sea posible. En particular, si el número de coincidencias en las puntuaciones entre dos usuarios es alta, probablemente es porque son realmente usuarios similares. Todas estas consideraciones apuntan a que cuantos más productos en común haya entre dos usuarios, más útiles serán las coincidencias en las puntuaciones. Así, proponemos una medida de similitud alternativa que tiene en cuenta este hecho. Sin embargo, no todas las coincidencias son igual de significativas. Como se mencionó previamente, una coincidencia en una película muy popular explica peor el comportamiento de un usuario que una coincidencia en una película muy extraña. De este modo, al medir la similitud entre dos puntuaciones es útil ponderarla por la importancia del producto en cuestión. De hecho, como se muestra en la Figura 4.11, esta medida también mejora la correlación de Pearson: incrementa el número de productos valorados en común por los pares de usuarios que presentan alta similitud. Sin embargo, la mejora no es tan buena como con la similitud del número de productos valorados en común.

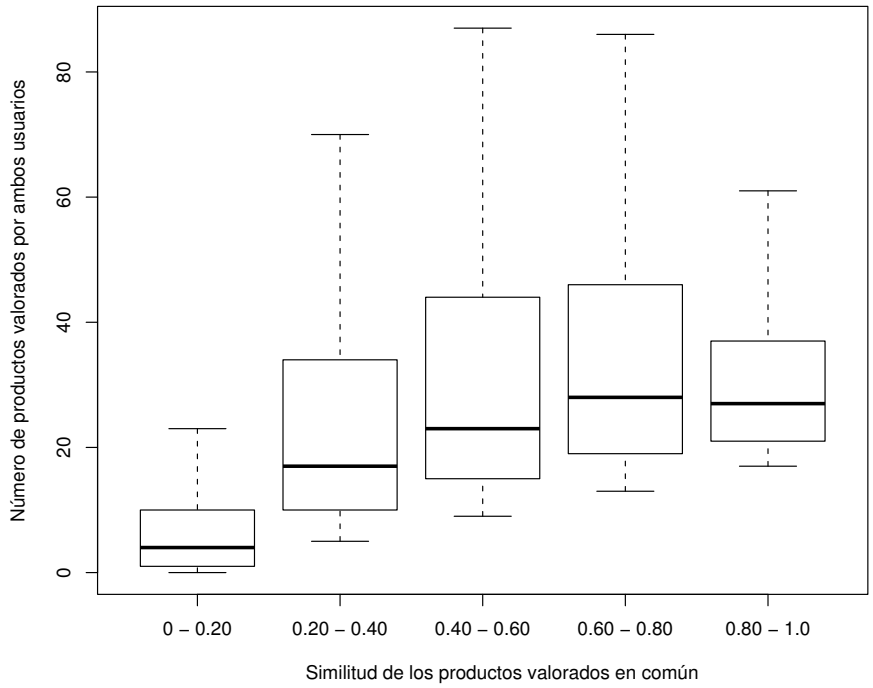


Figura 4.10: Número de productos valorados por ambos usuarios, usando la similitud de los productos valorados en común.

Por tanto, proponemos una medida de similitud que combina el número de productos valorados en común con la coincidencia real en las puntuaciones. De este modo, abordamos las limitaciones mencionadas de las medidas tradicionales.

Más formalmente, nuestra medida de similitud se define del siguiente modo:

$$s(a,u) = \beta * s_{common}(a,u) + (1-\beta) * s_{coinc}(a,u) \tag{4.1}$$

en donde el parámetro β pondera la contribución de los dos términos en los que se divide la medida de similitud, s_{common} y s_{coinc}.

El primer término, cuya definición se muestra en la Ecuación 4.2, representa la similitud de acuerdo a los productos en común.

$$s_{common}(a,u) = \frac{|I_a \cap I_u|}{|I_a|} \tag{4.2}$$

Con respecto a ello, es de mencionar que se usa una aproximación asimétrica, es decir, $s_{common}(u_1,u_2) \neq s_{common}(u_2,u_1)$. Por ejemplo, supongamos un usuario u_1 que ha valorado diez productos y un usuario u_2 que ha valorado cien productos. Si tienen diez productos en común, este valor representa el 100 % de los productos de u_1 y el

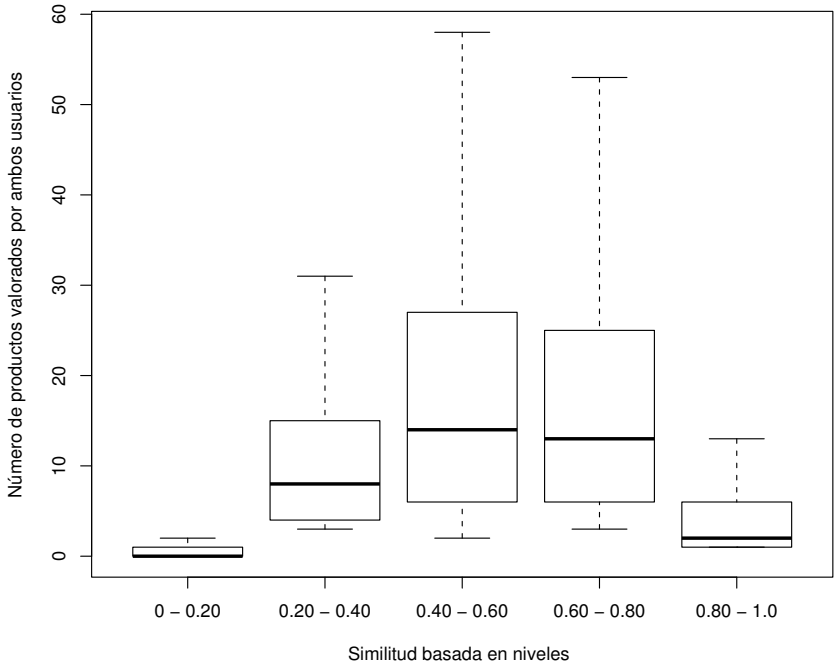


Figura 4.11: Número de productos valorados por ambos usuarios, usando la similitud de los productos basada en niveles.

10 % de los de u_2. Por tanto, si la medida fuese simétrica, u_1 presentaría muy baja similitud con todos salvo los usuarios en situación de *cold-start* que hayan valorado los mismos productos. Esto supondría un problema porque se podrían recomendar pocos productos nuevos. Con una aproximación asimétrica incrementamos las oportunidades de los usuarios en situación de *cold-start* sin hacer que los usuarios más activos paguen las consecuencias.

Por otro lado, las coincidencias en las puntuaciones se tienen en cuenta en el segundo término de la Ecuación 4.1, y se calculan de acuerdo a la siguiente fórmula:

$$s_{coinc}(a,u) = \frac{\sum_{i \in I_a \cap I_u} s_{ratings}(r_{ai}, r_{ui}) * iuf(i)}{\sum_{i \in I_a \cap I_u} iuf(i)} \tag{4.3}$$

En esta Ecuación 4.3 iuf (*inverse user frequency* representa la importancia de un producto. Cuantos más puntuaciones recibe, menor es su importancia; es decir, los productos populares no son adecuados para comparar el comportamiento de dos usuarios diferentes. El valor de iuf se calcula del siguiente modo:

$$iuf(i) = log\frac{|U|}{|U_i|} \tag{4.4}$$

Finalmente, hemos analizado dos modos sencillos de calcular la similitud entre las puntuaciones ($s_ratings$ en la Ecuación 4.3). En la primera, llamada medida basada en distancia, calculamos la diferencia entre dos puntuaciones dividida por el rango de todas las posibles puntuaciones. Cuanto mayor sea esta diferencia, menor será la similitud.

$$s_{ratings}(r_1, r_2) = 1 - \frac{|r_1 - r_2|}{r_{max} - r_{min}} \tag{4.5}$$

En la segunda alternativa, llamada medida basada en niveles, las puntuaciones se clasifican en dos grupos, siendo un grupo el de las *buenas puntuaciones* y el otro el de las *malas puntuaciones*. Decidimos usar como umbral de relevancia $\theta = 4$. Por tanto, dos puntuaciones se considerarán similares si pertenecen al mismo grupo, tal y como aparece definido en la Ecuación 4.6.

$$s_{ratings}(r_1, r_2) = \begin{cases} 1 & r_1 < \theta \wedge r_2 < \theta \\ 1 & r_1 \geq \theta \wedge r_2 \geq \theta \\ 0 & otherwise \end{cases} \tag{4.6}$$

4.5. Experimentos

4.5.1. Diseño de experimentos

Los experimentos han sido ejecutados usando el conjunto de datos de Movielens 10M, que fue analizado previamente en la Sección 4.3.

En lo que respecta a la metodología, hemos dividido el conjunto de datos en un subconjunto de entrenamiento y en un subconjunto de evaluación del siguiente modo: primero, hemos escogido aleatoriamente 10,000 puntuaciones de todo el conjunto de datos como subconjunto de evaluación. De los datos restantes, hemos usado diferentes porcentajes de puntuaciones como subconjuntos de entrenamiento. En particular, hemos variado en intervalos de 10 % los porcentajes, desde el 10 % hasta el 90 %. Hemos evaluado la tarea de predicción usando algoritmos kNN. Hemos usado métricas de precisión en la predicción como *Mean Absolute Error* (MAE) y *Root Mean Squared Error* (RMSE). Los resultados que se presentan en las siguientes secciones serán la media de los valores obtenidos para todas las puntuaciones del subconjunto de evaluación.

4.5.2. La influencia de los parámetros β y N

En primer lugar hemos estudiado la influencia del parámetro β, que pondera la contribución de cada una de las medidas de similitud propuestas. Con valores grandes de este parámetro, el valor de similitud entre dos usuarios depende en gran medida del porcentaje de productos que ambos hayan valorado, sin influir tanto la coincidencia en las propias puntuaciones. Por otro lado, con valores de β pequeños, la importancia de las coincidencias en las puntuaciones se ve incrementada.

Esperamos que con un pequeño porcentaje de puntuaciones como subconjunto de entrenamiento los mejores resultados se alcancen con un valor de β grande. La razón es que con una baja densidad de datos, cuanto más compleja es la medida de similitud, más problemas tiene esta medida para proporcionar valores precisos.

De hecho, estos valores que esperábamos se confirman en nuestros experimentos. Como se puede ver en la Figura 4.12, el MAE (RMSE presentan un comportamiento similar y no se muestra) más pequeño se alcanza con $\beta = 0.90$. De dicha figura también se puede apreciar que el impacto de β es más grande cuando la medida de similitud hace uso del factor basado en distancia. Cuando se usa el factor basado en niveles, el impacto no es especialmente importante. Esto también era lo esperado y, de hecho, confirma lo que se acaba de comentar: cuanto más compleja es una medida de similitud, peores resultados obtiene con baja densidad de datos. Cuando se usa la similitud basada en niveles, independientemente de los valores de β escogidos, la función de similitud usada es siempre simple. Es decir, un β grande únicamente reemplaza a una medida ya de por sí simple con una más simple todavía, así que el impacto no es demasiado alto. Sin embargo, cuando se usa la medida basada en distancia, más compleja, el impacto del parámetro β es más grande. Así, podemos concluir que cuando la información es extremamente dispersa, no tiene sentido tener una medida de similitud compleja, porque no hay información con la que pueda trabajar. En esos casos, la información es con frecuencia malinterpretada, conduciendo a malos resultado. Por ello, es buena idea usar una medida más simple. En particular, los resultados muestran que nuestra elección, basada en los productos valorados en lugar de en la puntuación concreta, es una buena alternativa.

Por otro lado, con mayor densidad las medidas de similitud más complejas empiezan a funcionar mejor, puesto que tienen una mayor cantidad de datos con los que trabajar, tal y como se puede comprobar en la Figura 4.13. Con el 90 % de las puntuaciones como subconjunto de entrenamiento, los mejores resultados se alcanzan con bajos valores de β, lo cual prueba que en ese caso las medidas más complejas superan a las alternativa simple basada en el número de productos valorados en común. Aunque no se representa

en la Figura 4.13, los resultados de MAE presentan un comportamiento similar.

En cuanto a la influencia del parámetro N, que controla el número de vecinos, los resultados son comparables con los presentados por las aproximaciones tradicionales basadas en usuarios. Es decir, los resultados empeoran notablemente cuando N toma valores bajos. Estos son los resultados esperados: incluso con medidas de similitud complejas, los usuarios más correlados no siempre son los mejores vecinos. Lo mismo sucede, por supuesto, si la medida de similitud escogida es más sencilla. Por consiguiente, tener más vecinos es útil para reducir el impacto de un vecino mal escogido. En los resultados presentados en este capítulo, el valor de N usado ha sido 50.

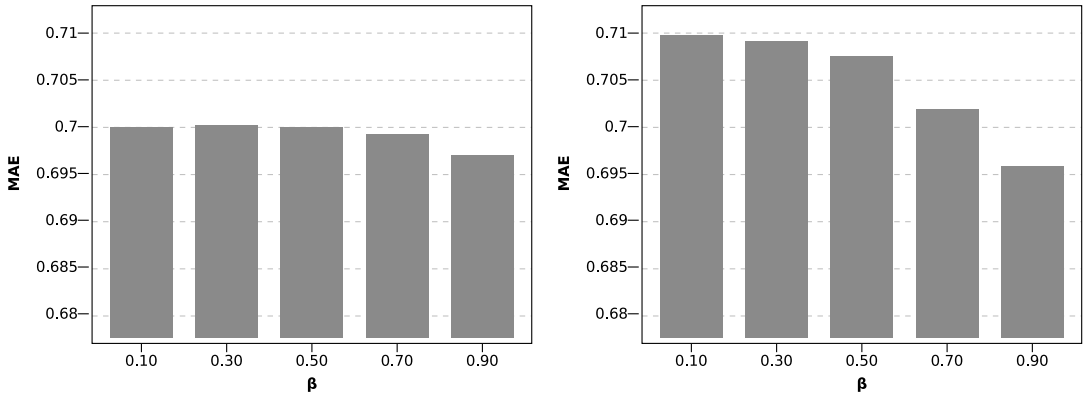


Figura 4.12: Resultados de MAE para distintos valores de β con el 10 % de las puntuaciones como subconjunto de entrenamiento, con la similitud basada en niveles (izquierda) y la similitud basada en distancia (derecha).

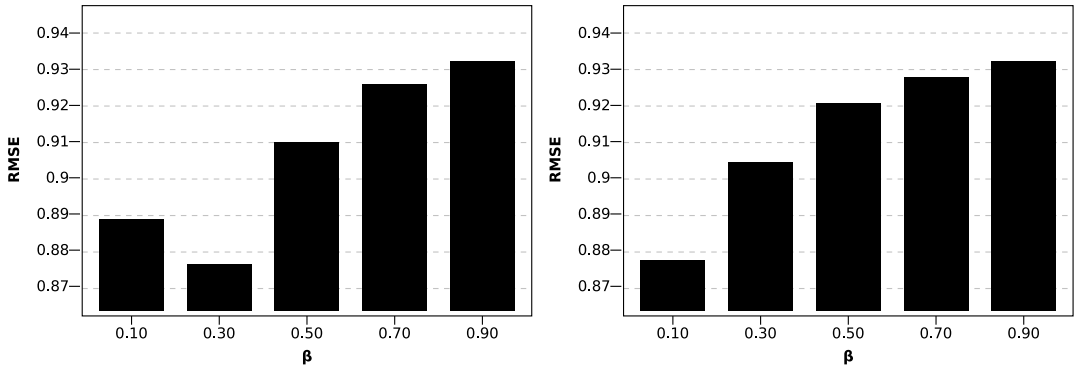


Figura 4.13: Resultados de RMSE para distintos valores de β con el 10 % de las puntuaciones como subconjunto de entrenamiento, con la similitud basada en niveles (izquierda) y la similitud basada en distancia (derecha).

4.5.3. Comparación con las medidas basadas en correlaciones

Hasta ahora en este capítulo hemos mostrado la importancia del parámetro β y su impacto en función de la información disponible. A continuación nos centraremos en cuál es la mejora real de las medidas simples, como la presentada en este capítulo, cuando se comparan con las técnicas tradicionales basadas en correlaciones. Para evaluar este punto, hemos comparado nuestra medida de similitud con dos medidas ampliamente usadas: la correlación de Pearson [Resnick et al., 1994] y la distancia coseno [Breese et al., 1998].

En la Figura 4.14 mostramos los resultados obtenidos cuando la información disponible es escasa. Se puede comprobar como nuestra aproximación simple mejora los resultados obtenidos por las medidas tradicionales. En particular, la correlación de Pearson muestra muy malos resultados en esas condiciones. La variante basada en distancias y la basada en niveles obtienen resultados muy similares, como era de esperar, porque sólo un 10 % de la similitud final depende de ellos.

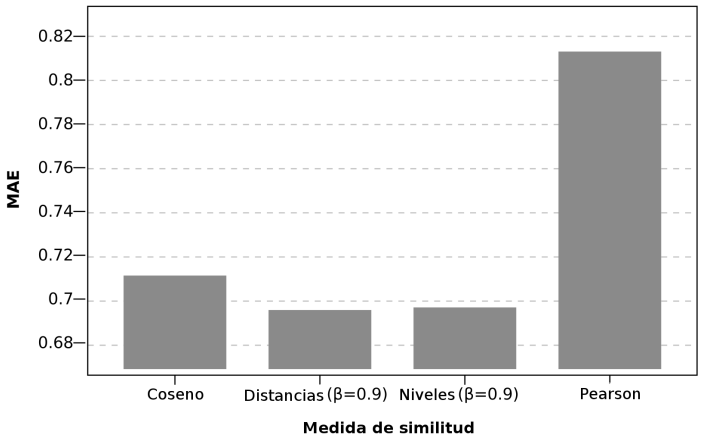


Figura 4.14: Comparación de distintas medidas de similitud, con el 10 % de las puntuaciones como conjunto de entrenamiento.

Por otro lado, con el 90 % de las puntuaciones en el subconjunto de entrenamiento, la distancia coseno obtiene mejores resultados, como se muestra en la Figura 4.15.

Los resultados obtenidos son los esperados y confirman lo que habíamos dicho previamente. Las medidas tradicionales son demasiado estrictas, y, cuando la información disponible es escasa, esto provoca que aparezcan resultados erróneos. Sin embargo, los resultados también muestran que la medida basada en distancias y la basada en niveles, que habían sido propuestas como alternativas al número de puntuaciones en común cuando la información no es tan dispersa, no mejoran a las medidas tradicio-

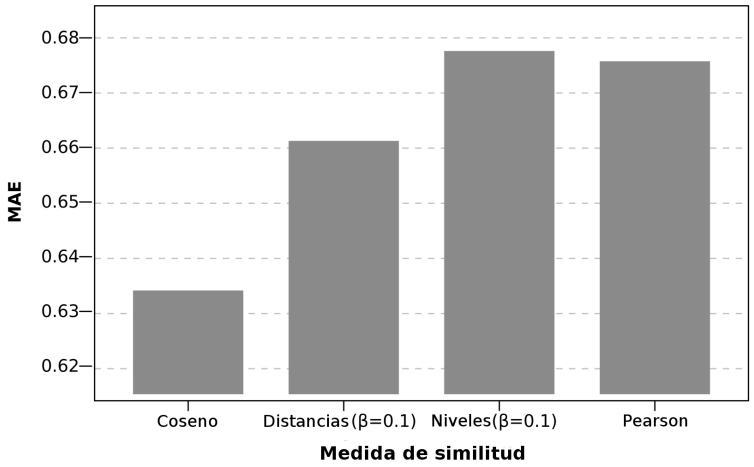


Figura 4.15: Comparación de distintas medidas de similitud, con el 90 % de las puntuaciones como subconjunto de entrenamiento.

nales. Así, una mejor aproximación podría ser combinar nuestra medida simple basada en el número de productos valorados en común con una aproximación tradicional más compleja.

4.5.4. Impacto de *Inverse User Frequency*

También hemos estudiado el impacto de *iuf*, que ha sido usado para tener en cuenta la popularidad de los productos para poder ponderar así su impacto en la medida de similitud entre usuarios. Sin este factor, usuarios diferentes podrían ser incorrectamente considerados similares simplemente porque están de acuerdo en los productos populares. Así, usando este factor se espera mejorar el cálculo de la similitud entre usuarios.

Sin embargo, los resultados muestran cómo el impacto de este factor es casi imperceptible. Sólo cuando el 90 % de las puntuaciones forman parte del subconjunto de entrenamiento, se observa una ligera mejora, como se muestra en la Figura 4.16. De hecho, el impacto en RMSE es incluso menor. Es decir, tener en cuenta *iuf* en el cálculo de la similitud entre usuarios no es que tenga impacto negativo, pero no consigue ninguna mejora significativa.

Sin embargo, es complicado evaluar si tiene un impacto nulo sólo en la precisión en la predicción, o también en el cálculo de la similitud. Puede ser posible que el factor *iuf* mejore realmente la medida de similitud. Sin embargo, al final, se usan varios usuarios, incluso con baja similitud, como parte del vecindario, así que la mejora posible no impacta en la precisión de la predicción. Como ya hemos mencionado, es necesario usar varios vecinos para poder reducir así el impacto de un vecino mal escogido. Así, en este

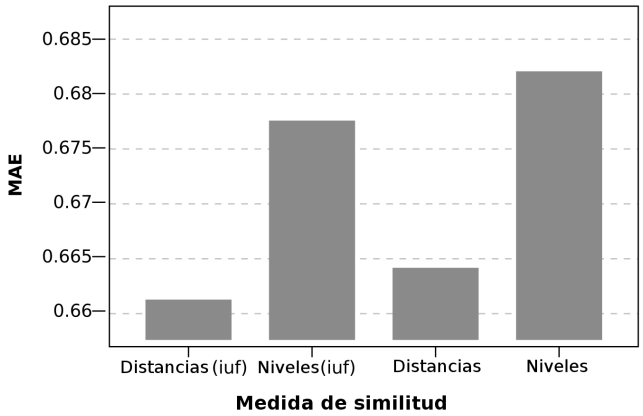


Figura 4.16: Impacto en MAE del factor *iuf*, tanto en la medida de similitud basada en distancia como en la medida basada en niveles, con el 90 % de puntuaciones como subconjunto de entrenamiento.

caso, el factor *iuf* no es determinante.

4.6. Conclusiones y trabajos futuros

En este capítulo hemos mostrado como una metodología en el diseño, que incluye un análisis de los datos del dominio, puede ser de ayuda a la hora de desarrollar algoritmos de filtrado colaborativo. En particular, hemos confirmado que muchos de los problemas presentes en los algoritmos tradicionales basados en usuarios pueden explicarse por medio de un estudio de los datos. Hemos propuesto también varias mejoras a esos algoritmos, como una medida de similitud basada en el porcentaje de productos valorados en común por cada par de usuarios, o el uso de *inverse user frequency* (*iuf*) para ponderar la contribución de cada producto en función de su popularidad. Hemos mostrado como la medida tiene un impacto ligeramente positivo cuando la información disponible es escasa, pero también que *iuf* no mejora significativamente la precisión en la predicción.

Como trabajo futuro, planeamos combinar las medidas de similitud tradicionales con una medida simple basada en el porcentaje de productos valorados en común. Nos centraremos también en una medida de similitud nueva basada en un β dinámico, de acuerdo a las conclusiones obtenidas en este capítulo. Es decir, usaremos diferentes β para cada usuario dependiendo del número de productos que haya valorado.

Además, las conclusiones obtenidas a partir de nuestro análisis del conjunto de datos pueden ser usadas para desarrollar nuevos modelos que se ajusten a los datos mejor que las aproximaciones existentes.

Capítulo 5

Mejora de la tarea de recomendación en sistemas dispersos

Los algoritmos de filtrado colaborativo se basan en las opiniones de usuarios con gustos o intereses similares al usuario para el cual se está realizando la recomendación. Esto provoca que alcancen mejores precisiones que otras alternativas, como los algoritmos basados en contenido. Sin embargo, este hecho también es el causante de una de su principales limitaciones: el problema del *cold-start*. En el Capítulo 4 explicamos dicho problema, muy relacionado con la dispersión de datos, y propusimos una mejora para los algoritmos kNN en predicción. En esta capítulo nos centraremos en la otra gran tarea de los sistemas recomendadores, la tarea de recomendación, y en cómo afrontar el problema del nuevo usuario (un tipo de situación de *cold-start*) mediante el uso de la expansión del perfil. En concreto, proponemos diversos métodos para combinar estas técnicas, aprovechando sus beneficios y disminuyendo sus limitaciones. De este modo intentamos mejorar el comportamiento de los algoritmos kNN también en la tarea de recomendación, ofreciendo recomendaciones precisas a un usuario que acabe de llegar al sistema o que haya valorado pocos productos.

5.1. Introducción

Las técnicas de filtrado colaborativo [Goldberg et al., 1992], que recomiendan productos basándose en las opiniones de otros usuarios, son muy populares dados sus buenos resultados en muchos dominios. En estas técnicas cada usuario se representa por medio de un perfil, formado por sus opiniones acerca de los productos que ha valorado. El sistema buscará usuarios con un perfil similar, y recomendará productos

valorados con una puntuación alta por usuarios similares. Obviamente, el perfil del usuario es incompleto, es decir, sólo contiene las valoraciones de un grupo reducido de productos, que son aquéllos que el usuario conoce y que ha decidido puntuar. Cuanto más grande es el perfil, más información tiene el sistema acerca del usuario. Esto trae consigo generalmente mejores recomendaciones. Por otro lado, cuando hay poca información disponible o ésta es muy dispersa, el sistema recomendador genera malas recomendaciones.

En particular, las técnicas de filtrado colaborativo sufren el llamado problema del nuevo usuario, un tipo de cold-start [Schein et al., 2002] que ya hemos comentado en la Sección 4.2 y que afecta a usuarios con pocas puntuaciones en su perfil. Ayudar a estos usuarios, sin embargo, es un aspecto crítico en cualquier sistema recomendador, pues puede suponer el éxito o el fracaso del mismo.

Tradicionalmente, este problema ha sido abordado con métodos híbridos, que combinan información colaborativa con otras fuentes de datos, como información demográfica, información que los usuarios no siempre están dispuestos a proporcionar y que ofrece una personalización limitada en muchos casos. En cuanto a técnicas puramente colaborativas, se han propuesto diferentes aproximaciones para tratar este problema. Todas ellas se pueden dividir en dos grandes grupos según el usuario intervenga o no en ellas. Cuando hacen partícipe al usuario éste proporciona al sistema nuevas valoraciones a cambio de poder recibir mejores recomendaciones. De este modo, todo usuario dispondrá al menos del número de valoraciones que el sistema le solicite en su registro. La elección de qué elementos presentar, como ya se ha comentado en el Capítulo 4, recibe el nombre de proceso de elicitación.

Sin embargo, a veces no es adecuado optar por esta solución. Por ejemplo, como sucedía con la información demográfica comentada anteriormente, algunos usuarios pueden considerar inoportuno que el sistema les solicite que valoren productos cuando se están registrando. Por esta razón, muchos trabajos en este ámbito han propuesto soluciones en las que el usuario no se ve involucrado.

En este capítulo abordaremos el uso de técnicas de expansión del perfil que propusimos en [Formoso et al., 2013a], y que están basadas en las técnicas de expansión de consultas.

Para tratar el problema se proponen diversas aproximaciones que incrementan automáticamente el perfil de los nuevos usuarios, lo que trae consigo mejores recomendaciones. Estas técnicas se pueden aplicar fácilmente a los algoritmos kNN y, además, no necesitan la intervención del usuario. Pueden ser habilitadas sólo en determinadas situaciones, como cuando el perfil del usuario no supere un determinado tamaño, por

ejemplo. Además, están basadas únicamente en las puntuaciones disponibles de los usuarios. Esta aproximación parte de la idea de que el problema del nuevo usuario puede ser abordado eficientemente sin la necesidad de molestar al usuario requiriéndole nueva información.

En este capítulo en primer lugar presentaremos las técnicas de expansión de perfil. A continuación explicaremos las técnicas de orden superior, centrándonos en qué ventajas aprovechan de los algoritmos presentados anteriormente, y cómo mitigan sus limitaciones. Después, realizaremos unos experimentos y analizaremos los resultados de los métodos propuestos, para finalizar comentando las conclusiones obtenidas y futuros trabajos.

5.2. Expansión de perfil

Como se mostraba en la Ecuación 3.1, en los algoritmos de filtrado colaborativo cada usuario $u \in U$ tiene asociado un perfil, P_u, compuesto de una lista de pares producto-puntuación, donde cada par está formado por un producto $i \in I$ que el usuario ha valorado, y su correspondiente puntuación v_{ui}

Los algoritmos kNN usan el perfil para calcular la similitud entre el usuario activo y el resto de usuarios del sistema. Los usuarios más parecidos, como ya hemos comentado previamente, se denominan vecinos y, dichos vecinos se usan para calcular la recomendación.

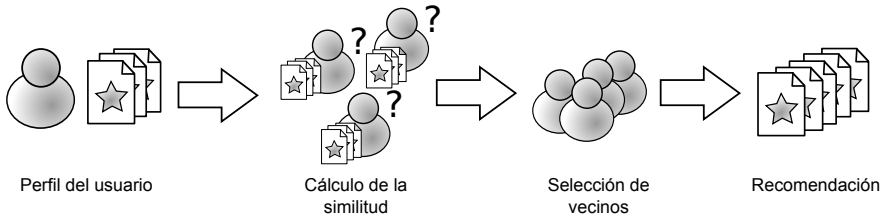


Figura 5.1: Comportamiento general de los algoritmos kNN, que usan el perfil del usuario para seleccionar los vecinos.

Se trata de un proceso secuencial que comienza con el perfil del usuario, tal y como se muestra en la Figura 5.1. Si el perfil es muy pequeño, el cálculo de la similitud no funcionará correctamente, ya que no hay información suficiente sobre los intereses del usuario. De hecho, un error en el cálculo de la similitud tendrá un fuerte impacto en los pasos siguientes, es decir, tanto en la selección del vecindario como en la elaboración de la lista de recomendación. A esto hay que unir que, como mostramos en el Capítulo 3, los algoritmos kNN no se comportan bien en entornos dispersos.

Para reducir el efecto de los entornos dispersos sobre los algoritmos kNN, decidimos encarar el problema del nuevo usuario proponiendo las técnicas de expansión de perfil [Formoso et al., 2013a]. La idea es incrementar el tamaño del perfil original añadiendo nuevos productos, para poder así mejorar los pasos siguientes. Para ello nos basamos en técnicas de expansión de consultas utilizadas en Recuperación de Información. Estas técnicas se emplean habitualmente cuando la consulta de búsqueda está incompleta o es ambigua, y el sistema no tiene información suficiente para satisfacerla correctamente. Se trata, por tanto, del mismo problema que los sistemas recomendadores tienen con los usuarios nuevos. Estas aproximaciones se suelen clasificar en análisis local y análisis global [Baeza-Yates y Ribeiro-Neto, 2008]. Mientras que el análisis local se centra en los documentos recuperados ante una consulta para proporcionarle términos relevantes adicionales a la consulta original, las técnicas globales buscan dichos términos en toda la colección de documentos.

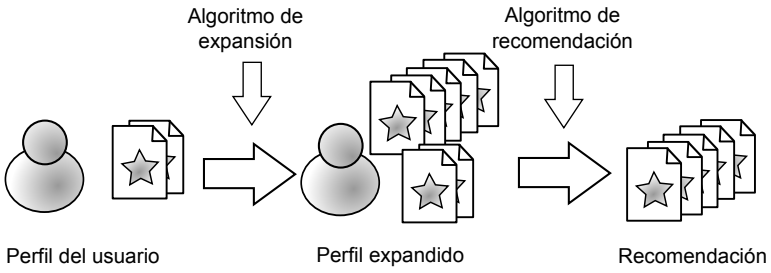


Figura 5.2: Expansión de perfil.

En resumen, tal y como se muestra en la Figura 5.2, cuando se usa la expansión del perfil, el proceso de recomendación consiste en un perfil inicial de usuario al que se le añaden nuevos productos por medio de un algoritmo de expansión. Este nuevo perfil es la entrada para el algoritmo de recomendación, que se encarga de calcular la lista de productos final.

Formalmente, el *perfil expandido*, P'_u, se define como la unión del perfil original con hasta l productos adicionales, tal y como se muestra en la Ecuación 5.1.

$$P'_u = P_u \cup \left\langle \left(i_{s+1}, r_{ui_{s+1}}\right), \left(i_{s+2}, r_{ui_{s+2}}\right), \ldots, \left(i_{s+l}, r_{ui_{s+l}}\right) \right\rangle \tag{5.1}$$

Por su parte, $P^e_u = \left\langle \left(i_{s+1}, r_{ui_{s+1}}\right), \left(i_{s+2}, r_{ui_{s+2}}\right), \ldots, \left(i_{s+l}, r_{ui_{s+l}}\right) \right\rangle$. Es decir, P^e_u es el conjunto de pares producto-puntuación que añadiremos al perfil. Además, I_u, según lo explicado en la Sección 3.1, es el conjunto de productos valorados por el usuario, e I^e_u es el conjunto de productos añadidos al perfil. Por supuesto, $I^e_u \cap I_u = \emptyset$.

Nos centraremos en técnicas colaborativas, ya que aprovechan el perfil de usuario

existente y no requieren que el usuario proporcione información adicional. Teniendo en cuenta las definiciones previas, los algoritmos de expansión se pueden clasificar atendiendo a diferentes criterios. Por un lado, como sucede con las técnicas de expansión de consultas, pueden ser locales o globales. Mientras que las técnicas locales se basan en los resultados de la recomendación actual y, por tanto, en el perfil del usuario activo, las técnicas globales usan información disponible independientemente del perfil para el cual se esté calculando la expansión.

Desde otro punto de vista, las técnicas también pueden ser basadas en usuarios o basadas en productos. Las primeras escogen los productos entre aquéllos valorados por un conjunto de usuarios, mientras que las segundas las eligen a partir de un conjunto de productos dado.

Según estas dos clasificaciones, en las siguientes secciones explicaremos los diferentes métodos de expansión de perfil (PE): expansión global basada en productos (IG), expansión local basada en productos (IL), expansión global basada en usuarios (UG) y, finalmente, expansión local basada en usuarios (UL).

5.2.1. Expansión global basada en productos

Las técnicas globales basadas en productos intentan encontrar productos parecidos a aquéllos que están presentes en el perfil del usuario. Como se muestra en la Figura 5.3, la expansión se realiza antes que cualquier otro cálculo. Para poder encontrar los productos similares, estas técnicas usan información global disponible en el sistema.

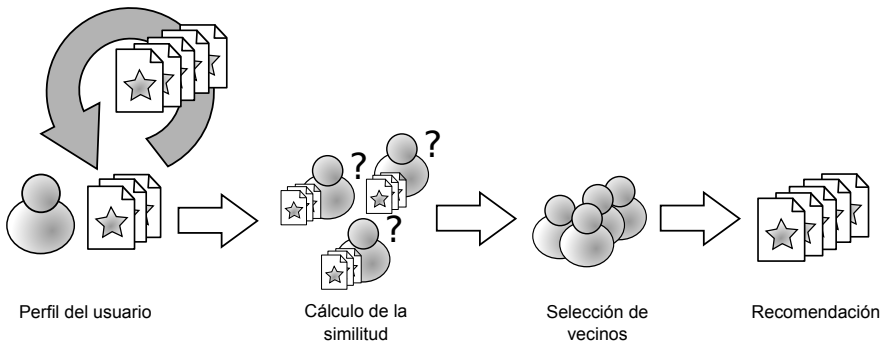


Figura 5.3: Expansión de perfil global basada en productos.

Como algoritmo de expansión usamos un algoritmo kNN basado en productos [Sarwar et al., 2001], también con la distancia coseno como medida de similitud.

Los l productos más similares se seleccionan para ser añadidos al perfil. A cada producto escogido se le asigna la puntuación que el usuario dio al producto al que es similar. En caso de que se haya escogido un producto que es similar a varios productos

del perfil, la puntuación se calculará como una media ponderada en función de las respectivas similitudes.

5.2.2. Expansión local basada en productos

La técnica local basadas en productos se centra en los productos recomendados al usuario, tal y como se muestra en la Figura 5.4. Por tanto, se necesita una recomendación inicial que se empleará para realizar la expansión. Después de ésta, se hará una segunda recomendación usando ya el perfil expandido.

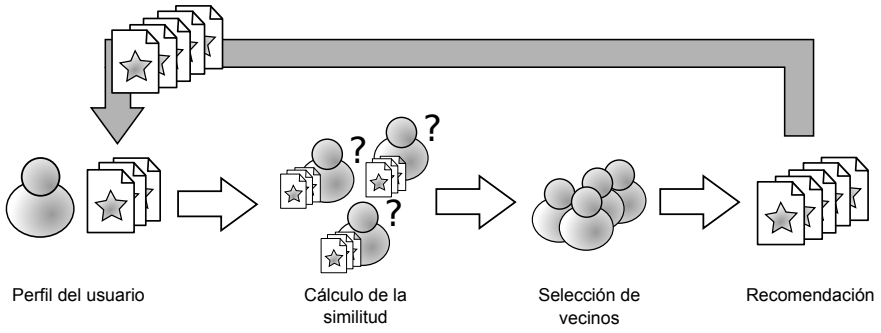


Figura 5.4: Expansión de perfil local basada en productos.

Aunque existen diversas aproximaciones posibles, en el trabajo mencionado empleamos una aproximación sencilla que consiste en expandir el perfil con los l productos mejor valorados de la lista de recomendación.

Sin embargo, en [Formoso et al., 2013a] quedó demostrado que esta aproximación no se comporta bien. De hecho, llega a empeorar los resultados de los algoritmos sin expansión en muchos casos. La razón por la que esto sucede es porque se están expandiendo el perfil con productos recomendados, y éstos, al haber poca información acerca de los usuarios, puede que sean malas recomendaciones.

5.2.3. Expansión global basada en usuarios

Este tipo de técnicas usan información global sobre los usuarios para expandir el perfil. La idea es añadir al perfil productos valorados por usuarios similares. En la práctica, para esta aproximación se podría emplear información adicional, como datos demográficos. Cuando sólo se usa información colaborativa para calcular la similitud entre usuarios, como se pretende en este capítulo, esta técnica es muy parecida a las aproximaciones locales basadas en usuarios que se presentará a continuación. Dado que el objetivo es mostrar soluciones puramente colaborativas esta opción no será evaluada y se nombra únicamente.

5.2.4. Expansión local basada en usuarios

Como se muestra en la Figura 5.5, las técnicas locales basadas en usuarios se centran en los vecinos del usuario actual. Los productos que se escogen para expandir el perfil se seleccionan entre aquéllos valorados por los vecinos.

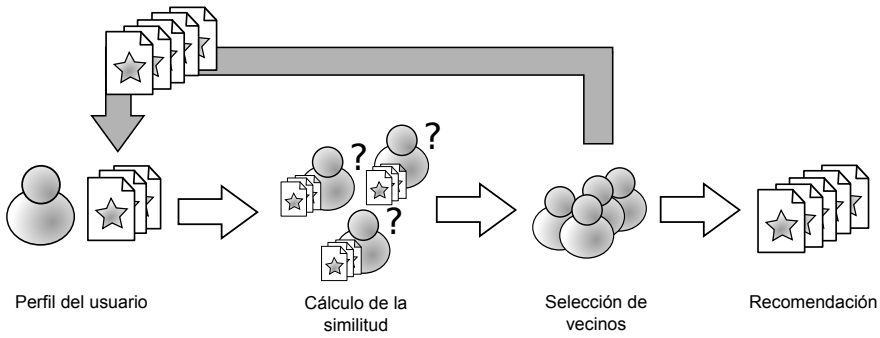


Figura 5.5: Expansión de perfil local basada en usuarios.

Aunque se pueden emplear diversas técnicas, se han estudiado las siguientes aproximaciones:

- Productos mejor puntuados (TR). El perfil se expande con los l productos mejor puntuados. Si un producto fue valorado por varios vecinos, la puntuación se obtiene ponderando por la similitud de cada uno de ellos, análogamente a como se hace en las técnicas locales basadas en productos. La única diferencia es que pueden ser añadidos productos con cualquier puntuación, no sólo los buenos. En [Formoso et al., 2013a] comprobamos que este método no funciona demasiado bien, debido, en gran medida, a las mismas razones por las que las técnicas locales basadas en productos funcionaban mal: el algoritmo no tiene suficiente información para seleccionar los mejores productos para un usuario dado.

- Productos más puntuados (MR). El perfil se expande con los l productos más valorados por los vecinos, sin importar la puntuación o la similitud con cada vecino.

- Vecinos locales (LN). Con esta técnica se expande el perfil con los productos más parecidos a aquéllos que figuran en el perfil del usuario teniendo en cuenta sólo las puntuaciones dadas por los usuarios del vecindario.

- Agrupamiento local de usuarios (LC). Este método intenta capturar las relaciones entre productos, pero teniendo en cuenta sólo las puntuaciones dadas por el

vecindario del usuario activo. Se basa en la técnica de la matriz de asociación local [Attar y Fraenkel, 1977] usada para la expansión de consultas. Para cada producto $i \in I_u$, la similitud entre él y otro producto $j \in I$, es decir, $\varsigma(i,j)$, se calcula según la Ecuación 5.2.

$$\varsigma(i,j) = \frac{\sum_{n \in N(u)} r_{ni} r_{nj}}{|N(u)|} \tag{5.2}$$

5.3. Expansión de perfil de order superior

En la sección anterior hemos presentado distintas técnicas de expansión de perfil para tratar el problema del nuevo usuario.

Cuando se seleccionan elementos no relacionados con los del perfil del usuario, las técnicas de expansión de perfil pueden inducir errores que afecten a la recomendación final. Además, aunque se seleccionen correctamente los elementos, los errores pueden venir también causados por una incorrecta predicción de la valoración asignada.

La idea de la expansión de perfil de rango superior es combinar las técnicas de expansión de perfil para aprovechar sus beneficios y minimizar sus errores. Usando más de un algoritmo para expandir el perfil, la expansión será más heterogénea y, al mismo tiempo, seguirá estando relacionada con el perfil inicial del usuario.

Hemos considerado dos tipos de combinaciones: serie y paralela.

5.3.1. Serie

La Figura 5.6 representa el proceso de expansión en serie. Como se puede apreciar, se usan dos algoritmos, uno a continuación de otro. Primero, el perfil inicial del usuario es expandido usando el algoritmo 1. El perfil expandido será la entrada del algoritmo 2. Finalmente, el algoritmo principal hará uso de la salida de éste para calcular la lista de recomendación. Por tanto, los algoritmos se ejecutan a modo de cadena. Este tipo de combinación puede ser considerada como el modo más cercano al concepto original de expansión de perfil, en donde un algoritmo es usado para incrementar la información disponible para otro algoritmo. Cuando la información es muy escasa, el primer algoritmo debería funcionar correctamente y minimizar errores. Un error en el primer paso del proceso es arrastrado a lo largo de las fases restantes, por lo que se hace significativo.

La diferencia entre tener un algoritmo con un tamaño de expansión dado y tener el mismo algoritmo dos veces en un proceso de expansión serie con la mitad del tamaño

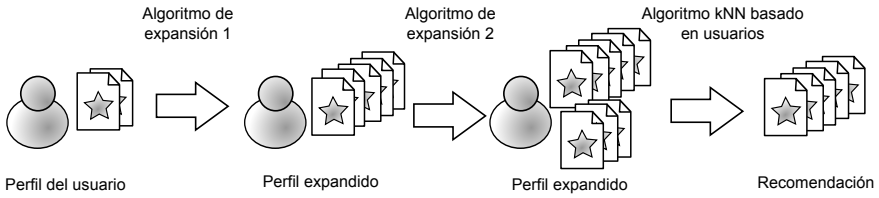


Figura 5.6: Alternativa en serie.

de expansión es que los elementos que ocupan las últimas posiciones en la lista de expansión calculada con el algoritmo de referencia es más probable que sean no relevantes y que, de esa manera, conduzcan a peores recomendaciones. Lógicamente cuanto mayor sea el tamaño de la expansión, mayor será también la probabilidad de que los últimos elementos de la lista de recomendación sean no-relevantes. Así que, en general, es preferible un proceso en serie ejecutando dos veces el mismo algoritmo frente a la ejecución del algoritmo de referencia una única vez con una expansión del doble de tamaño.

5.3.2. Paralela

Como ya se ha comentado, las técnicas de expansión de perfil pueden inducir errores tanto a la hora de seleccionar los elementos como cuando se predicen las valoraciones, y dichos errores pueden afectar a la recomendación final. Esta aproximación intenta minimizar estos errores usando la misma entrada (el perfil inicial del usuario) para los algoritmos de expansión, tal y como se muestra en la Figura 5.7.

Los perfiles de expansión obtenidos se combinan para dar como resultado un único perfil, que será la entrada del algoritmo recomendador. En este proceso un mismo elemento puede aparecer en más de un perfil expandido, con la misma valoración o con valoraciones diferentes. Cuando las valoraciones son distintas, la valoración final será la media de las valoraciones. En cuanto a la selección de los elementos, consideramos dos alternativas:

- Suma. El conjunto de elementos en el perfil final se forma a partir de la unión de los elementos de cada una de las expansiones. Por ello, cuando se combina el mismo algoritmo más de una vez, el resultado es la expansión con el tamaño más grande.

- Cruce. El conjunto de elementos en el perfil final se forma a partir de la intersección de los elementos de cada una de las expansiones. En este caso, cuando el mismo algoritmo se combina más de una vez, el resultado es la expansión con el tamaño más pequeño.

A diferencia de la aproximación en serie, si un algoritmo comete un error, este error no afecta a la expansión calculada por el otro algoritmo de expansión. A cambio el segundo algoritmo tiene a su disposición menos información; únicamente el perfil inicial del usuario.

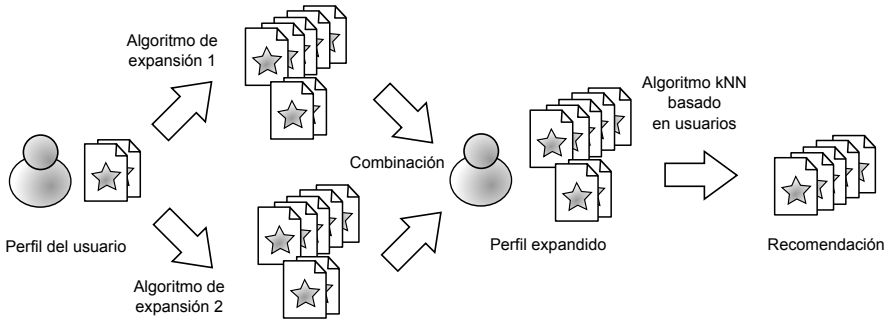


Figura 5.7: Alternativa en paralelo.

5.4. Experimentos

5.4.1. Diseño de experimentos

Los experimentos han sido ejecutados en un dominio de recomendación de películas. En concreto, hemos usado una versión reducida del conjunto de datos Netflix [Bennett y Lanning, 2007], en donde cada película se asocia con una película de Internet Movie Database (IMDb)[1]. De este modo, podemos realizar manualmente también una evaluación de los resultados y comparar los resultados de algoritmos basados en contenido. El conjunto de datos final consta de 8.362 elementos y 478.458 usuarios, que han realizado 48.715.350 valoraciones entre 1 y 5, ambos inclusive.

En cuanto a la metodología, hemos seleccionado aleatoriamente 1000 usuarios de evaluación. El problema del nuevo usuario ha sido simulado usando una estrategia Given-N [Breese et al., 1998]. Es decir, se han escogido aleatoriamente N valoraciones como parte del subconjunto de entrenamiento. El resto de valoraciones forman parte del subconjunto de evaluación. Nos centramos en $N = 2$, ya que se trata de la situación en la que la información es más escasa y las técnicas de expansión de perfil pueden ser más útiles. Además, estudiamos la evolución de los algoritmos según N. De los demás usuarios, hemos seleccionado aleatoriamente un 90 % de sus valoraciones para el subconjunto de entrenamiento.

[1] http://www.imdb.com

En relación a los algoritmos, primero estudiaremos cómo se compartan los algoritmos simples de expansión de consultas. Después, abordaremos los distintos modos de combinar dichas técnicas. El algoritmo principal es un algoritmo kNN basado en usuario. El número de vecinos, k, que hemos utilizado es 25, número obtenido tras un proceso de validación cruzada.

Ya que nos centramos en la tarea de recomendación, el algoritmo devolverá una lista de elementos. Para evaluar los resultados, consideramos 4 como umbral de relevancia. Hemos usado métricas tradicionales como precisión y MAP. En particular, prestaremos especial atención a la precisión en los primeros 5 elementos de la lista (P@5), porque considera aquellos elementos a los que el usuario probablemente preste atención y no todos los elementos de la lista, como hace MAP.

5.4.2. Técnicas de expansión de perfil

La Tabla 5.1 muestra los resultados de P@5 con diferentes tamaños de expansión (l) para un algoritmo kNN sin expansión y para las técnicas presentadas en [Formoso et al., 2013a] que mejores resultados obtuvieron, tanto globales basadas en productos, como locales basadas en usuarios.

Las técnicas globales basadas en productos muestran mejoras significativas con respecto a los algoritmos tradicionales sin expansión de perfil cuando el número de puntuaciones disponibles para los usuarios es muy pequeño, puesto que los algoritmos kNN no funcionan satisfactoriamente en estos casos. De hecho, la precisión en las primeras posiciones de la lista de recomendación es particularmente buena. Conforme el tamaño del perfil (N) de los usuarios considerados es más grande, el beneficio de este tipo de técnicas es menor, como es de esperar.

En cuanto a la influencia del tamaño de la expansión, l, los mejores resultados de esta técnica se obtienen cuando l es pequeño, especialmente cuando el tamaño de la lista de recomendación es también pequeña. Entre $l = 2$ y $l = 5$ no hay ninguna diferencia significativa. De hecho, los resultados de MAP para $l = 5$ son mejores, de ahí que sean los que se muestren en la Figura 5.8. Conforme l aumenta, el perfil se expande con productos que no son particularmente parecidos a los del perfil del usuario, conduciendo esto a malos resultados.

Por otro lado, la Tabla 5.1 también muestra los resultados de P@5 para las mejores técnicas locales basadas en usuarios: productos más puntuados, vecinos locales y agrupamiento local de usuarios. A pesar de que la técnica de los productos más puntuados sólo considera los productos con mayor número de valoraciones del vecindario y de que tanto la técnica de los vecinos locales como la de agrupamiento local de usuarios tratan

de capturar las similitudes entre productos, todas ellas presentan resultados similares de P@5. Sin embargo, la técnica de los productos más puntuados es preferible dada su simplicidad, a pesar de no presentar tan buenos resultados con la métrica MAP.

Los algoritmos locales basados en usuarios también mejoran significativamente la precisión de los nuevos usuarios. Muestran unos resultados particularmente buenos cuando el tamaño de los perfiles es muy pequeño, como sucedía con los algoritmos de expansión global basada en productos, siendo el método agrupamiento local de usuarios y el de los productos más valorados los que obtienen mejores resultados de precisión en los primeros elementos de la lista de recomendación.

En resumen, cuando se usa la expansión global basada en elementos la lista de recomendación tiene menos especificidad, porque toda la información disponible es usada, pero también presenta menor número de errores, ya que la mayoría de los elementos que son similares a los que aparecen en el perfil del usuario probablemente serán escogidos por el algoritmo. Por tanto, obtienen buenos valores para MAP, puesto que todo los elementos de la lista (no sólo los que están en las primeras posiciones) están relacionados con los elementos del usuario.

Por otro lado, con los algoritmos locales basados en usuario, probablemente se encontrarán los elementos más similares, porque los vecinos tenderán a valorarlos. Sin embargo, en el vecindario también pueden figurar elementos que no están en absoluto relacionados con los del usuario actual. Entonces, estos algoritmos presentarán buena precisión en las primeras posiciones de la lista, pero un MAP no tan bueno.

De aquí en adelante, en vista de los resultados, las técnicas con $l = 5$ se usarán como técnicas de referencia.

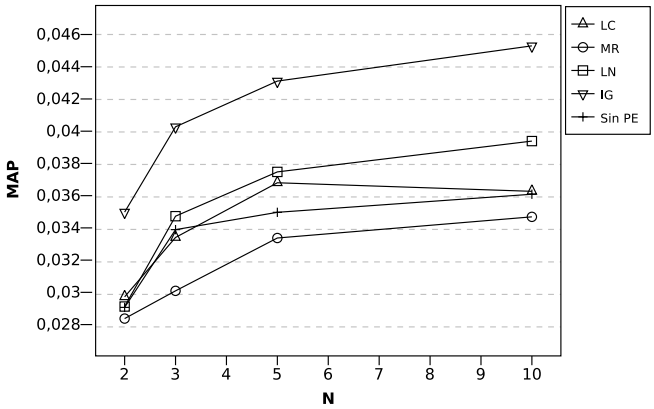


Figura 5.8: Evolución de MAP para los *baselines* con $l = 5$.

Algoritmos	l	$N = 2$		$N = 10$	
		P@5	MAP	P@5	MAP
IG	2	0.125	0.029	0.123	**0.040**
	5	0.122	0.035	0.118	**0.045**
	10	0.100	**0.038**	0.097	**0.046**
	15	0.086	**0.039**	0.076	**0.044**
	20	0.078	**0.038**	0.071	**0.043**
	50	0.044	0.034	0.048	0.037
LC	2	**0.136**	0.025	**0.148**	0.035
	5	**0.140**	0.030	**0.140**	0.036
	10	0.123	0.034	0.128	**0.040**
	15	0.112	0.036	0.118	**0.041**
	20	0.102	**0.038**	0.110	**0.042**
	50	0.077	**0.040**	0.085	0.039
LN	2	0.134	0.023	**0.140**	0.037
	5	**0.139**	0.029	**0.141**	0.039
	10	0.128	0.032	**0.140**	0.038
	15	0.121	0.035	**0.141**	0.037
	20	0.111	**0.037**	0.138	0.037
	50	0.083	**0.041**	0.137	0.037
MR	2	**0.138**	0.024	**0.146**	0.031
	5	**0.147**	0.029	0.139	0.035
	10	0.134	0.032	0.127	0.038
	15	0.117	0.033	0.123	0.037
	20	0.109	**0.036**	0.115	0.038
	50	0.076	**0.040**	0.087	**0.042**
Sin PE	-	0.078	0.029	**0.140**	0.036

Tabla 5.1: P@5 y MAP para los *baselines*. Se destacan los mejores valores.

5.4.3. Expansión de perfil de orden superior

La idea principal de nuestros experimentos es averiguar si los algoritmos de orden superior superan los resultados de los algoritmos simples de expansión de perfil. Además, entre las distintas alternativas, estamos interesados en evaluar cuál se comporta mejor y en conocer sus beneficios y inconvenientes. Para poder lograr esto, se han realizado distintos experimentos. En ellos, se han usado diferentes tamaños de expansión (2, 5, 10 y 15) para comprobar cómo las combinaciones de l afectan a los resultados. También nos interesa conocer cómo evolucionan los algoritmos de acuerdo a N, es decir, a la

cantidad de información disponible.

5.4.3.1. Serie

En las combinaciones en serie, dos técnicas de expansión de perfil son usadas una a continuación de la otra. Aunque los algoritmos de la Sección 5.4.2 obtienen sus mejores resultados con $l = 5$, los mejores resultados combinándolos en serie se obtienen con $l = 2$, si bien también presentan buenos resultados con $l = 5$. La Tabla 5.2 muestra los resultados de P@5 y MAP para combinaciones con $l = 2$.

Algoritmos		$N = 2$		$N = 10$	
1	2	P@5	MAP	P@5	MAP
LC	LC	0.153	0.026	0.146	0.034
	LN	0.147	0.026	0.148	0.036
	MR	**0.156**	0.025	**0.153**	0.036
	IG	0.149	**0.032**	0.147	**0.040**
LN	LC	0.154	0.027	**0.151**	0.035
	LN	0.142	0.024	0.140	0.037
	MR	**0.158**	0.025	**0.151**	0.033
	IG	**0.156**	**0.032**	0.127	**0.041**
MR	LC	**0.163**	0.026	**0.153**	0.034
	LN	0.152	0.025	0.146	0.032
	MR	**0.155**	0.024	**0.151**	0.034
	IG	**0.160**	**0.030**	0.137	0.039
IG	LC	0.139	**0.030**	0.144	**0.040**
	LN	0.137	**0.031**	0.128	**0.041**
	MR	0.145	0.029	0.143	**0.041**
	IG	0.133	**0.035**	0.129	**0.045**
Sin PE		0.078	0.029	0.140	0.036

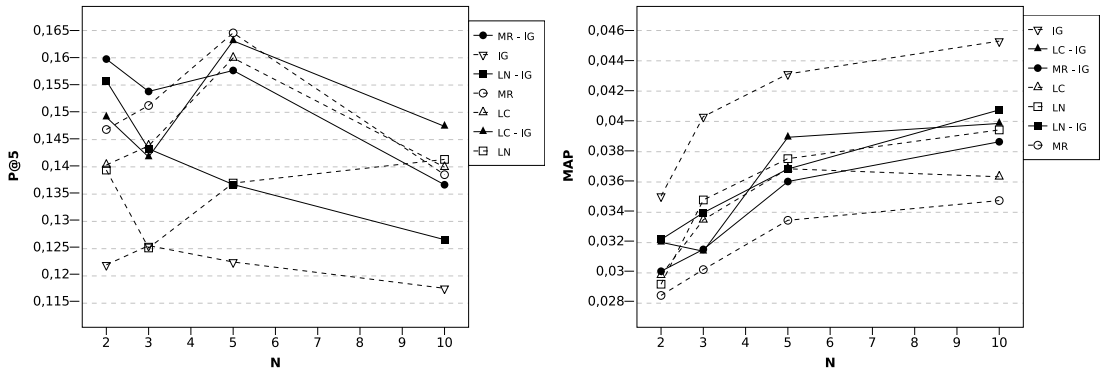
Tabla 5.2: P@5 y MAP para las combinaciones en serie, con $l = 2$. Se destacan los mejores valores.

A diferencia de las combinaciones en paralelo, en este caso el orden en la ejecución sí que puede tener relevancia. Así que, primero, nos centraremos en cómo influye el orden de los algoritmos en los resultados. Se puede comprobar cómo la técnica de vecinos locales presenta mejores resultados cuando se ejecuta en primer lugar. Por contra, el algoritmo global considerado funciona mejor cuando es usado de segundo algoritmo, mejorando así los resultados de MAP con respecto a los algoritmos de referencia. Las

técnicas de los algoritmos más puntuados y de agrupamiento local de usuarios funcionan mejor como primero o segundo en función del algoritmo con el que son combinados. Estas observaciones son en cierta medida predecibles y lógicas, porque cuando la información sobre un usuario es escasa, la información global es más propensa a errores, ya que es más difícil seleccionar los productos que son verdaderamente similares a aquéllos contenidos en el perfil del usuario. Sin embargo, cuando se usa información local, hay menos información disponible, pero ésta tiende a estar relacionada con el perfil del usuario. De aquí en adelante, ya que dos algoritmos distintos se pueden combinar usando dos órdenes diferentes, sólo presentaremos aquél con el que se obtengan mejores resultados.

Para tratar el conjunto de posibles combinaciones, las dividiremos en tres grupos: primero, aquéllas que combinen la técnica global con cualquier otro algoritmo; segundo, aquéllas que combinen dos técnicas locales diferentes; y, tercero, aquéllas que combinen un algoritmo consigo mismo.

La Figura 5.9 muestra los resultados con P@5 y MAP de la combinación de una técnica local con una técnica global. Cuando $N = 2$, es decir, cuando la información disponible acerca del usuario actual es extremadamente escasa, este tipo de combinaciones mejora las precisiones de los algoritmos de referencia. En particular, combinar la técnica de los productos más puntuados con la técnica global mejora en casi un 10 % los resultados de precisión anteriores. También los valores para MAP son mejores. De este modo, combinando la técnica de los productos más puntuados con la técnica global no sólo se mejora la precisión, sino que también los resultados de MAP obtenidos con la técnica de los productos más puntuados. Ya que parten de distinto tipo de información,



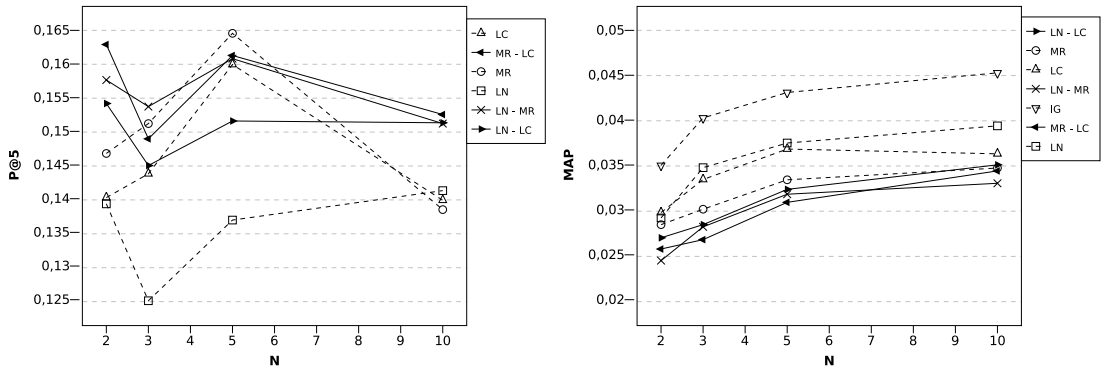
(a) P@5 **(b)** MAP

Figura 5.9: Evolución de P@5 y MAP según N, usando combinaciones en serie con la técnica global basada en productos y $l = 2$.

el usar ambos permite obtener mayor cantidad de información sobre el usuario, lo cual, finalmente, provoca que la lista de recomendación pueda ser más precisa. Debido a la misma razón, al combinar la técnica de vecinos locales con la técnica global, siendo N pequeña, la mejora en precisión es importante, en comparación con la que se obtenía con ambos algoritmos individualmente.

El comportamiento de las técnicas de los productos más puntuados y de los vecinos locales con la aproximación global es muy similar. La precisión tiende a disminuir de acuerdo al incremento de N. Esto es debido a que la escasez de información no es tan crítica cuando $N = 10$. De hecho, mientras el algoritmo sin expansión puede obtener una precisión en torno a 0.14, las alternativas de orden superior pueden inducir algún error a la hora de predecir la valoración, y ésta puede llevar a una pérdida de precisión en la recomendación.

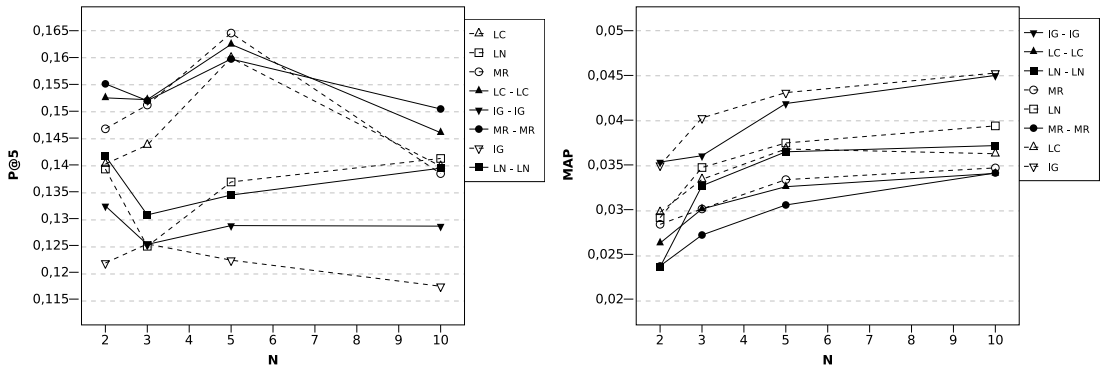
Con respecto a las combinaciones en serie de algoritmos locales, también presentan buenos resultados, tal y como se puede apreciar en la Figura 5.10. De nuevo, es con $N = 2$ cuando la mejora es más destacable, incrementando así la utilidad de las técnicas de expansión de perfil cuando está disponible muy poca información acerca del usuario. Dado que el algoritmo de los productos más puntuados originalmente alcanzaba una alta precisión, las combinaciones con este algoritmo obtienen también altas precisiones. En este caso, con $N = 10$ también se consiguen buenos resultados. Sin embargo, en contraposición a cuando las técnicas globales participan en la combinación, los valores de MAP son peores. De cualquier modo, como fue mencionado en la Sección 5.4.1, en términos generales consideramos más útil P@5 que MAP cuando se trata de una tarea de recomendación, aunque esta afirmación depende del contexto particular del sistema.



(a) P@5 **(b)** MAP

Figura 5.10: Evolución de P@5 y MAP según N, usando combinaciones en serie con las técnicas locales basadas en usuarios y $l = 2$.

Finalmente, un algoritmo puede ser combinado consigo mismo también. Los resultados con estas combinaciones se muestran en la Figura 5.11. Todos los algoritmos salvo la técnica de vecinos locales mejoran los resultados en precisión cuando son ejecutados siguiendo un proceso en serie. Estos resultados confirman que es preferible dicho proceso en serie a ejecutar el algoritmo individualmente duplicando el tamaño de expansión, tal y como se mencionó en 5.3.1. Los resultados de MAP son sólo ligeramente inferiores a los obtenidos con los algoritmos de referencia.



(a) P@5 **(b)** MAP

Figura 5.11: Evolución de P@5 y MAP según N, usando el mismo algoritmo dos veces y $l = 2$.

Como resultado general, las combinaciones en serie de los algoritmos locales muestran muy buenos resultados en P@5, mientras que las combinaciones con técnicas globales alcanzar mejores resultados en MAP. La técnica de los productos más puntuados con la técnica de agrupamiento local de usuarios o la técnica global son particularmente intersantes, ya que tienen una mejora del 10 % en P@5 sobre los algoritmos locales, y de cerca del 110 % sobre el algoritmo sin expansión de perfil. De hecho, la combinación de la técnica de los productos más puntuados con cualquier otra alternativa mejora los resultados de referencia. La razón es que se complementan entre sí: mientras la técnica de productos más puntuados se centra en aquellos items que reciben una mayor cantidad de valoraciones por parte de los vecinos, las demás se centran en buscar similitudes entre los productos valorados por los vecinos o entre el resto de productos disponibles, respectivamente.

5.4.3.2. Paralela

La variantes paralelas consideradas son la técnica del cruce y la técnica de la suma. La primera alternativa consiste en hacer una intersección de los perfiles expandidos

obtenidos con diferentes técnicas para así poder crear un único perfil expandido. Debido a esto, los perfiles expandidos pueden no tener elementos en común si tenemos en cuenta muy pocos productos como expansión. Esto podría causar que el paso de expansión de perfil resultase inútil. Así, para esta variante los tamaños de expansión han sido incrementados con respecto a la alternativa en serie, de tal modo que sea más probable que dos técnicas diferentes proporcionen algún elemento en común. Hay que tener en cuenta que son de naturaleza diversa, por lo que los elementos obtenidos no tienen por qué coincidir. Mientras que $l = 5$ es suficiente para obtener buenos resultados cuando sólo se consideran los algoritmos locales, no sucede lo mismo cuando los globales intervienen en las combinaciones. De hecho, en todo el conjunto de datos puede haber muchos elementos similares a los que se encuentran en el perfil del usuario activo, y es probable que sólo unos pocos de ellos hayan sido valorados por los usuarios vecinos. Esta es la razón por la que decidimos usar un tamaño de expansión de perfil más grande ($l = 100$). Esta decisión fue corroborada por los resultados.

La Tabla 5.3 muestra los resultados con P@5 y MAP. Los valores de MAP son bastante buenos, pero los resultados obtenidos con P@5 no superan los logrados con las alternativa en serie.

Algoritmos		Cruce				Suma			
1	2	$N = 2$		$N = 10$		$N = 2$		$N = 10$	
		P@5	MAP	P@5	MAP	P@5	MAP	P@5	MAP
LC	LN	**0.146**	0.029	0.141	0.038	0.137	0.025	0.147	0.034
	MR	**0.143**	0.029	**0.152**	0.036	**0.144**	0.027	**0.152**	0.034
LN	MR	**0.147**	0.028	0.141	0.038	0.138	0.024	**0.148**	0.032
IG	LC	0.128	0.029	**0.148**	0.038	**0.142**	**0.031**	0.140	**0.041**
	LN	0.139	0.028	0.141	0.039	0.134	**0.030**	0.125	**0.040**
	MR	0.133	0.027	0.143	0.038	**0.144**	**0.030**	**0.148**	**0.040**

Tabla 5.3: P@5 y MAP para las alternativas en paralelo. La alternativa del cruce con $l = 5$ para todos los algoritmos salvo el global basado en productos, que usa $l = 100$. La alternativa de la suma con $l = 2$ para todos los algoritmos. Se destacan los mejores resultados.

Por otro lado, la técnica de la suma agrupa las expansiones obtenidas con los distintos algoritmos. Hemos empleado $l = 2$ como tamaño de expansión. Como se muestra en la Tabla 5.3 los resultados no mejoran tampoco los obtenidos con la alternativa en serie. De hecho, ninguna combinación obtiene un valor de P@5 mejor que el que obtiene

la técnica de los productos más puntuados por sí sola.

Por tanto, a pesar de que con las alternativas en paralelo el segundo algoritmo no se ve afectado por el error cometido por el primero, los resultados con las combinaciones en serie son mejores, ya que el segundo algoritmo dispone de más información

5.5. Conclusiones

En este capítulo mejoramos los resultados obtenidos con las técnicas de expansión de perfil, que tratan el problema del nuevo usuario en los algoritmos de filtrado colaborativo. En particular, para lograr esta mejora hemos presentado dos modos de combinar las técnicas originales: la combinación en serie y la combinación en paralelo.

Los experimentos han mostrado como la alternativa en serie funciona mejor que la paralela. La razón es que en la alternativa en serie el segundo algoritmo se puede aprovechar de la información proporcionada por el primer algoritmo. Sin él, el segundo quizá no podría obtener dicha información. Esto no sucede con la alternativa en paralelo, donde los algoritmos usan la misma información.

En cuanto a las diferentes combinaciones en serie, los mejores resultados de P@5 se obtienen en aquéllas en donde participa la técnica de los productos más puntuados. Además, el algoritmo global incrementa el MAP obtenido con las técnicas locales cuando son usadas conjuntamente. De este modo, la combinación de la técnica global con la de los productos más puntuados alcanza muy buenos resultados, principalmente cuando N es pequeña.

Asimismo, cuando dos métodos locales distintos se combinan, los resultados de P@5 alcanzan buenos resultados con todos los valores estudiados para N. Sin embargo, no sucede lo mismo con los valores para MAP, que no sobrepasan los obtenidos con la técnica de los productos más puntuados, los más bajos entre todas las alternativas consideradas.

Otro modo de combinar las técnicas de expansión de perfil es ejecutando el mismo algoritmo dos veces consecutivas. En este caso, los resultados obtenidos con este tipo de combinaciones mejoran los resultados obtenidos cuando los mismos algoritmos son ejecutados una sola vez, con la excepción de la técnica de vecinos locales.

En el futuro, planeamos estudiar otros algoritmos de filtrado colaborativo aparte los algoritmos kNN. Se puede optar por una solución recursiva, en la que en cada iteración se emplea un algoritmo diferente de filtrado colaborativo. Además, es nuestra intención probar la combinación de más de algoritmos diferentes. Finalmente, se podría añadir un número diferente de productos al perfil en función de cada usuario, siguiendo diferentes

parámetros como el tamaño del perfil, la popularidad de los productos que en él figuran, etc.

Con respecto al estudio de la tarea de recomendación, los trabajos sobre las técnicas de expansión de perfil pueden ser continuados desde muchos frentes. En particular, aparte de los algoritmos basados en vecinos se podrían usar otros algoritmos de filtrado colaborativo. Además, la expansión de orden superior permite múltiples combinaciones. Se puede optar por una solución recursiva, usando diferentes algoritmos de filtrado colaborativo en diferentes iteraciones de la expansión. Aunque hasta el momento sólo hemos propuesto combinaciones dos a dos de algoritmos de expansión, el número de algoritmos combinados puede ser ampliado, sobre todo teniendo en cuenta que también sería interesante el estudio de nuevas fuentes de información como datos demográficos, las relaciones entre usuarios en las redes sociales, etc.

Otro aspecto interesante que abordar en el futuro acerca de la expansión de perfil es el de añadir un número diferente de productos a la expansión en función del tamaño del perfil de cada usuario, puesto que no todos los usuarios del sistema tienen por qué estar en situación de *cold-start*.

Aunque hemos tratado la posibilidad de realizar expansiones del perfil del usuario activo, existe también la alternativa de expandir la matriz globalmente, es decir, aumentar los perfiles de todos los usuarios o, al menos, de aquéllos que no hayan valorado un número mínimo de productos.

También puede ser estudiado cómo influyen en los resultados finales las medidas de similitud de los algoritmos kNN tanto del algoritmo de recomendación, como de los algoritmos de expansión. El número de vecinos también puede ser modificado según la similitud que tengan con el usuario actual.

Como se puede comprobar, la expansión del perfil supone un tema que abre un gran abanico de futuros trabajos posibles, que podrían ser de gran utilidad para afrontar el problema de la dispersión de los datos.

Capítulo 6

Expansión de perfil basada en contenido

Las técnicas de filtrado colaborativo sufren el problema de cold-start. La expansión de perfil se ha presentado como un modo de abordar este problema sin molestar al usuario con peticiones de información adicionales. Estas técnicas añaden más información al perfil del usuario. Hasta el momento, se ha optado por un enfoque meramente colaborativo, si bien técnicas basadas en contenido pueden ser también una buen opción para el problema a tratar. Para ello es necesario analizar previamente el conjunto de datos del que se dispone y proponer una solución acorde a los resultados de dicho análisis. La solución propuesta emplea tanto información colaborativa como información de contenido.

6.1. Introducción

Un sitio de comercio electrónico puede disponer de una enorme variedad de artículos diferentes. Estos artículos son difícilmente conocidos en su totalidad por los usuarios, que, a su vez, poseen variedad de gustos y de necesidades a satisfacer. Para incrementar sus beneficios, muchos de estos sitios recomiendan artículos en función de los gustos de los usuarios. Con este propósito, hacen uso de sistemas recomendadores, que han llegado a ser muy populares en los últimos años. Las tareas que acometen pueden abarcar desde la predicción de puntuaciones hasta la recomendación de listas de productos [Herlocker et al., 2004].

Las técnicas de recomendación basadas en filtrado colaborativo construyen un perfil que contiene información acerca de los gustos del usuario. Sin embargo, cuando esta información es escasa (situación de nuevo usuario), ya sea porque el usuario acaba de

llegar al sistema o porque simplemente ha valorado muy pocos productos, generalmente los sistemas de recomendación no son capaces de lograr una recomendación suficientemente precisa para poder ser útil para el usuario. Como se explica en la Sección 4.3, el 20 % de los usuarios sólo proporcionan un 2 % de las valoraciones en el conjunto de datos de Netflix [Bennett y Lanning, 2007]. Este es un dato muy relevante, puesto que ayudar a estos usuarios con pocas puntuaciones puede significar la diferencia entre una experiencia exitosa o fallida para ellos.

Las alternativas consideradas en la literatura se centran en incrementar el tamaño del perfil del usuario, es decir, la información disponible acerca de los gustos del usuario. Sin embargo, a veces estas aproximaciones no proporcionan a los usuarios recomendaciones suficientemente personalizadas. En otras ocasiones, se les solicita información adicional a los usuarios, que no siempre facilitan, bien sea porque simplemente les molesta introducir más información, por motivos de privacidad, etc. Es decir, cuanto más información se le exige al usuario, más piensa éste que el sistema sabe demasiado sobre él, por lo que evita proporcionar más datos.

Las técnicas de expansión de perfil se han presentado como un medio para afrontar este problema sin solicitar al usuario más información. Los productos para la expansión se obtienen localmente, de los resultados de la recomendación actual, o globalmente, es decir, se buscan similitudes entre todos los usuarios o productos.

En este capítulo nos centramos de nuevo en la tarea de recomendación, puesto que numerosas aplicaciones hoy en día presentan a cada usuario una lista de productos acorde a sus intereses. De hecho, no sólo abordamos el problema del nuevo usuario por medio de las técnicas de expansión de perfil, sino que también proponemos una situación extrema donde todos los usuarios presentan pocas puntuaciones (situación de nuevo sistema). Para ello emplearemos de nuevo una expansión de perfil, pero esta vez haciendo uso de información de contenido. Los sistemas basados en contenido, como ya se ha explicado en la Sección 2.4, construyen perfiles de usuario obtenidos a partir de las características de contenido de los productos que los usuarios puntúan.

El capítulo está organizado del siguiente modo. En la próxima sección explicamos nuestra aproximación con información basada en contenido. La metodología y los experimentos se describen en la Sección 6.3. Posteriormente, discutimos los resultados obtenidos y exponemos las conclusiones alcanzadas.

6.2. Expansión de perfil basada en contenido

La idea de aplicar técnicas basadas en contenido a la expansión de perfil está justificada por el hecho de que son capaces de encontrar relaciones semánticas entre los productos. Cuando la información acerca de un usuario es escasa, la información de contenido puede ser relativamente importante. Dicha información puede ser representada por medio de un conjunto de características.

El usuario dispondrá de un perfil por cada una de las características consideradas. A partir de estos perfiles de contenido, la técnica propuesta busca productos similares a aquéllos que el usuario ha valorado. Para lograr esto, primero identificamos por medio de un análisis de los datos las características que mejor definen los gustos del usuario, así como el modo de comparar el perfil de contenido con la información de contenido acerca de un producto concreto. Una vez que los productos son seleccionados, son añadidos al perfil del usuario como si realmente los hubiese valorado, por lo que sus puntuaciones han de ser estimadas. Después, el sistema recomendador se ejecuta del modo habitual. A continuación, analizamos cada una de las principales fases de la expansión de perfil basada en contenido.

6.2.1. Selección de características

El objetivo de la expansión es encontrar productos suficientemente similares a los que aparecen en el perfil inicial del usuario. Primero es necesario analizar las características disponibles para poder determinar así cuáles de ellas son más útiles para comparar los productos o, en otras palabras, qué características son las más relevantes para determinar los gustos de un usuario y caracterizar los productos.

En este capítulo, como en los anteriores, estamos centrados en un dominio de recomendación de películas. En particular, usamos un conjunto de datos filtrados de Netflix, que será descrito en la Sección 6.3. Para identificar las características más útiles para describir el perfil de un usuario, hemos realizado un análisis manual seleccionando aleatoriamente 25 usuarios. Además, se escogieron dos productos por cada usuario para conformar los perfiles iniciales. Este primer análisis nos condujo hacia varias conclusiones que fueran confirmadas con un análisis automático de 1000 usuarios (también aleatoriamente escogidos). La media y la desviación típica del número de productos valorados por cada usuario son 99.93 y 135.66, respectivamente. De hecho, sólo el 33 % de los perfiles tienen más de 100 productos.

A continuación, se expone un breve análisis con las principales conclusiones extraídas para cada característica y la motivación para su utilización en la expansión

del perfil. Entre ellas, reparto, país, director y género han sido seleccionadas para ser usadas por nuestro algoritmo.

- Género. Cada película se categoriza de acuerdo a ciertos parámetros, que están relacionados principalmente con el tipo de las escenas que más abundan en ella. De este modo, se considera que una película se ajusta a un conjunto de moldes. Sin embargo, dicha categorización no es para nada sencilla. La razón es que la pertenencia a ciertos géneros como acción, comedia, drama, etc. es en gran medida subjetiva. De hecho, hay varios aspectos que no están incluidos en la característica de género. Por ejemplo, películas que tienen los mismos géneros pueden considerar diferentes temas, culturas... Por tanto, esta característica permite clasificar las películas en grupos, aunque esta clasificación presenta ciertas limitaciones, ya que es no puramente objetiva. A pesar de ello, esta es la característica de contenido que a priori mejor define los intereses de un usuario y las similitudes entre películas.

  Si un valor de género aparece más frecuentemente en un perfil de usuario que en todo el conjunto de datos, indica la preferencia de ese usuario por ese género en concreto. De otro modo, si el género aparece menos frecuentemente, suele significar que el usuario no está interesado en ese tipo de películas. Conforme el tamaño del perfil excede más el tamaño medio, la distribución de los valores de género del usuario se van acercando más a los del conjunto de datos. Además, las películas que pertenecen a los mismos géneros y tienen calidad parecida suelen recibir valoraciones parecidas también.

- País. Una película da una percepción sobre una cultura en particular, un modo de vida, etc. Principalmente, el país de la película es lo que determina esta percepción. Esto puede acentuar las diferencias entre películas. Por ejemplo, entre las películas indias hay muchos musicales (principalmente películas de Bollywood). Sin embargo, estos musicales pueden diferir mucho de los musicales franceses. Los usuarios no suelen ver películas de todos los países, sino que de sólo un subconjunto de ellos.

  En el conjunto de datos analizado, cuando un perfil inicial presenta películas de un determinado país, dicho país tiende a aparecer frecuentemente a lo largo de todo el perfil.

- Director. Cada director tiene su propio estilo de rodar las películas. Esto puede ser o no del agrado de un determinado usuario. En los perfiles estudiados, los usuarios no suelen valorar películas del mismo director. Sin embargo, esto no

implica que esta característica no pueda ser útil. Por ejemplo, si un usuario ha valorado "La ventana indiscreta" (dirigida por Alfred Hitchcock), es razonable pensar que valoraría "Vértigo" de forma similar. Por tanto, esta película se convierte en candidata para la expansión. Es interesante mencionar que los directores no aparecen más de una vez en los perfiles iniciales. De hecho, los directores que aparecen en el perfil inicial raramente aparecen de nuevo en el perfil, aunque, cuando aparecen son valorados de forma parecida.

- Reparto. Tal y como sucedía con los directores, las actrices y los actores no suelen aparecer repetidos en los perfiles estudiados. Siguiendo el mismo razonamiento, cuando una película en un perfil presenta una coincidencia relevante en cuanto a su reparto con otra película no valorada todavía por ese usuario, parece razonable expandir el perfil con ella. Sin embargo, una actriz o un actor presentes en el perfil inicial raramente aparecen de nuevo en el perfil completo, a pesar de que el perfil inicial contenga alrededor de 43 actores, por término medio. La razón es que cada uno sólo aparece 4.09 veces de media, con una desviación típica 3.87.

  Sin embargo, los usuarios tienden a valorar de forma semejante las películas de un mismo actor. Esto se debe a que, en general, las películas de los mismos actores no varían excesivamente entre ellas, ya que muchos actores y actrices están encasillados en determinados roles. Por tanto, esta característica puede ser útil para estimar las valoraciones de los productos.

- Año de estreno. Muchos trabajos han mostrado cómo las valoraciones de los usuarios varían con el tiempo de diferentes modos [Koren, 2010b; Lathia et al., 2010]. Un usuario puede ver sólo películas recientes mientras que otro puede ver películas clásicas. El año de estreno de las películas valoradas por un usuario puede ayudar a caracterizar su preferencia por un periodo de tiempo particular. Sin embargo, como estamos tratando con el problema de cold-start, esta información es demasiado escasa para poder estimar de una forma precisa la preferencia de un usuario por un determinado periodo de tiempo. Hemos decidido no incluir esta característica por esa razón.

- Guionista. Los guionistas y los directores se comportan de modo similar. Además, no hay valores repetidos en los perfiles estudiados. Debido a estas dos razones, hemos decidido no considerar esta característica tampoco.

- Tipo. Hay varias clases de artículos en los conjuntos de datos de películas: series, episodios, películas, etc. Cada elemento acepta un único valor para esta

característica. Aunque la mayor parte de los productos son películas, hay algunos usuarios que están interesados en episodios de una serie, por ejemplo. Incluir esta característica no aportaba ninguna diferencia significativa en los resultados, por lo que no ha sido considerada.

6.2.2. Representación de las características

Después de haber seleccionado las características, la siguiente fase es decidir cómo representarlas. Denotamos con F el conjunto de características de contenido que hemos tenido en cuenta. Sea L^f el conjunto de los valores o niveles que puede tomar una determinada característica f. Por ejemplo, considerando el género como una característica del conjunto F, L^{genero} contendrá valores como acción, comedia, drama, etc. Un producto, por tanto, puede presentar un subconjunto de todos los niveles que puede tomar una característica, por lo que podemos definir L_i^f como los niveles que una característica toma para una película $i \in I$.

De forma similar a la definición del perfil colaborativo en la Ecuación 3.1, denotamos con P_u^f el perfil de contenido de un usuario $u \in U$ para una característica de contenido $f \in F$. Cada perfil está formado por n pares de un valor $l_i \in L^f$ y su correspondiente peso w_{ui}, siendo n el número de niveles para la característica f, $\left|L^f\right|$. La definición se muestra en la Ecuación 6.1.

$$P_u^f = \langle (l_1, w_{ul_1}), (l_2, w_{ul_2}), \ldots, (l_n, w_{ul_n}) \rangle \tag{6.1}$$

En esta primera aproximación, los pesos son simplemente el número de veces que cada valor de la característica f aparece en el perfil del usuario. Por ejemplo, si un usuario u_1 ha valorado una comedia y una película que es drama y comedia, entonces el peso para el valor comedia será 2, para drama será 1, y 0 para el resto de valores.

De modo análogo, P_i^f es el perfil de contenido para un producto $i \in I$ y una característica de contenido $f \in F$.

6.2.3. Medidas de similitud

Una vez que las características han sido seleccionadas y su representación ha sido detallada, el siguiente paso es estudiar cómo se puede calcular la similitud entre dos productos usando estas características. Para acometer esta tarea, 25 usuarios ya no son suficientes por lo que se ha utilizado todo el conjunto de datos.

En este capítulo proponemos dos medidas de similitud: la tradicional distancia coseno y una función heurística que llamamos *similitud ponderada*.

6.2.3.1. Distancia coseno

Esta medida de similitud mide la distancia entre el perfil de contenido de un usuario y el de un producto según la siguiente ecuación:

$$s^f(u,i) = \frac{\sum_{l \in L^f} w_{ul} w_{il}}{\sqrt{\sum_{l \in L^f} w_{ul}^2} \sqrt{\sum_{l \in L^f} w_{il}^2}} \tag{6.2}$$

6.2.3.2. Similitud ponderada

En la Sección 6.2.1 se mencionó que si un valor aparece más de una vez en el perfil inicial, probablemente aparecería más veces en el perfil completo. Por ejemplo, para la característica de género, hay 4.54 valores diferentes en el perfil inicial por término medio, y 5.06 si se incluyen los valores repetidos. Cuando esta repetición tiene lugar en el perfil del usuario, hay de media un 43.46 % de los productos del perfil completo que comparten dicho valor. El resto de valores aparecen en menos del 30 % de los productos.

La similitud ponderada representa este hecho, tal y como se aprecia en la Ecuación 6.3, donde u es un usuario, i un producto, w_{ul} es el peso que toma el nivel l en P_u^f, y N es el número de productos valorados por el usuario u.

$$s^f(u,i) = max \left\{ \frac{w_{ul}}{N} \mid (l, w_{ul}) \in P_u^f \wedge l \in L^f \right\} \tag{6.3}$$

Siguiendo con el ejemplo anterior, la similitud entre el usuario u_1 y un producto i_1, con los valores de comedia, drama y romance para la característica de género, será el máximo de los pesos asociados a los 3 valores de género dividido por el número de películas que el usuario u_1 ha valorado (en esta caso 2). Ya que los pesos son 2/2, 1/2 y 0/2, respectivamente, la similitud será 1.

6.2.4. Selección de productos

Los productos se ordenan de acuerdo a un cierto peso. Ya que se usan diversas características, se pueden combinar las similitudes obtenidas para producir un único valor para ordenar la lista definitiva. En particular, hemos escogido las característica de género y país para la selección de productos, ya que concluimos en el análisis que éstas eran las características más adecuadas para determinar los gustos de un usuario.

La similitud de contenido, w_c, entre un producto y un perfil de usuario será una combinación ponderada de las similitudes calculadas para cada característica. Entonces, los productos con un peso más alto serán escogidos. Hemos decidido emplear un método heurístico para ello.

Además, tenemos como objetivo maximizar el tamaño de la expansión, ya que, en general, cuanto más información se tiene acerca de un usuario, más precisa es la recomendación. A diferencia de las técnicas de elicitación, en donde los usuarios explícitamente valoran productos sugeridos por el sistema, las técnicas de expansión incrementan el tamaño del perfil del usuario sin la intervención de éste. Lógicamente, esto puede causar errores, que debe ser minimizados para que la precisión no se vea decrementada excesivamente. Por tanto, la selección del número de elementos a expandir se convierte en una decisión clave.

6.2.4.1. Información colaborativa

Los algoritmos kNN basados en usuario que emplearemos para la tarea de recomendación buscan similitudes entre usuarios. Cuando existen pocos productos en un perfil, no es sencillo encontrar suficientes usuarios en el sistema que compartan con el usuario actual un mínimo número de productos. Por tanto, el vecindario estará basado en pocos usuarios o en usuarios con poca similitud con el usuario activo. Con un vecindario así es complicado alcanzar precisión a la hora de calcular la lista de recomendación.

De forma análoga, una vez calculada la expansión, los productos presentes en ella puede que no aparezcan en otros perfiles de usuario a pesar de ser muy similares a los del perfil inicial. De nuevo, el vecindario se basará en pocos usuarios, a pesar de que el perfil contenga más productos. Así, parece una buena idea considerar la *popularidad* de los productos a la hora de escoger los elementos a incluir en la expansión. De este modo, no sólo expandiremos con productos similares, sino que también con productos que aparecen frecuentemente en los perfiles de usuario, de tal modo que el vecindario pasa a ser suficientemente grande.

Modelaremos la popularidad con una función lineal. Particularmente, la popularidad se calcula dividiendo el número de puntuaciones que un producto ha recibido entre el número total de usuarios.

En resumen, calcularemos un peso total, w_t, considerando tanto la similitud de contenido como la popularidad. Este peso se describe en la Ecuación 6.4, en donde w_c es el peso de contenido, w_p es el peso de la popularidad normalizada con una normalización de máximos y mínimos y α es un parámetro que controla la contribución de cada peso.

$$w_t = \alpha w_c + (1 - \alpha) w_p \tag{6.4}$$

A diferencia de la recomendación, en donde la popularidad puede causar varios problemas cuando sólo se recomiendan productos populares, como un descenso en el

coverage, novedad o diversidad, en la expansión de perfil estos problemas pueden ser reducidos, ya que estas técnicas se usan antes de la recomendación.

6.2.5. Estimación de la puntuación

Después de seleccionar los productos con los que expandir, es necesario calcular la puntuación para cada uno de ellos. En particular, hay determinados estudios, como [Herlocker et al., 1999], en donde se muestra que los usuarios puntúan de modos muy diferentes.

Se pueden considerar distintas aproximaciones para determinar la puntuación de cada producto. Podemos tener en cuenta información colaborativa (desde la media del producto hasta una combinación ponderada de las puntuaciones dadas por los vecinos más similares al usuario actual, por ejemplo). Pero también la información de contenido puede ser de ayuda. La contribución de cada característica para determinar la puntuación puede variar de la contribución para seleccionar los productos con los que expandir. Hemos seleccionado la combinación de género, reparto y director basándonos en las conclusiones extraídas del análisis hecho para la selección de características. Es importante indicar que no se ha establecido ningún umbral de relevancia para la puntuación de los productos de la expansión, a diferencia de lo que sucede en las recomendaciones de productos, en donde se proponen sólo productos buenos al usuario, es decir, por encima de un determinado umbral de puntuación.

Han sido propuestas varias alternativas para poder determinar si la información de contenido es útil o no para estimar las puntuaciones.

6.2.5.1. Propuesta colaborativa

La aproximación colaborativa que hemos usado es la puntuación media de la película. Generalmente obtiene resultados aceptables con las métricas que se emplean normalmente en la tarea de predicción, como, por ejemplo, MAE.

6.2.5.2. Propuestas basadas en contenido

Aparte de la puntuación media, hemos probado tres propuestas basadas en contenido: simple, ponderada y z-score. Para ello definimos un perfil de puntuaciones para cada característica de contenido, en donde cada par está formado por un nivel $l_i \in L^f$, y un valor asociado a él, v_i, como se muestra en la Ecuación 6.5. Este valor tendrá diferentes significados en función del tipo de propuesta. Denotamos con R_u^f el perfil de puntuaciones de contenido de un usuario $u \in U$. Este perfil está compuesto de n pares nivel-valor, siendo n la cardinalidad de $\left|L^f\right|$.

$$R_u^f = \left\langle \left(l_1, v_{ul_1}^f\right), \left(l_2, v_{ul_2}^f\right), \ldots, \left(l_n, v_{ul_n}^f\right) \right\rangle \tag{6.5}$$

De este modo obtenemos un valor para cada nivel de una característica. Cada uno de estos valores se basa en la similitud entre el perfil de contenido del usuario y los niveles de contenido del producto para el que se está calculando la puntuación.

Por otro lado, también necesitamos definir el conjunto de productos que presentan un particular nivel de una característica en un perfil de contenido de un usuario, tal y como se muestra en la Ecuación 6.6.

$$I_{ul}^f = \{i \mid i \in I_u, l \in L_i^f\} \tag{6.6}$$

La puntuación final se calculará como la combinación lineal de las puntuaciones parciales de acuerdo a los diferentes pesos asignados a cada característica. A continuación, explicamos las propuestas basadas en contenido consideradas.

- Simple. Los valores numéricos del perfil de puntuaciones se calculan directamente a partir de las puntuaciones del perfil colaborativo, del siguiente modo:

$$v_{ul}^f = \frac{\sum_{i \in I_{ul}} r_{ui}}{\left|I_{ul}^f\right|} \tag{6.7}$$

  Para cada característica se calcula una predicción de la puntuación que daría el usuario al producto j según la Ecuación 6.8.

$$\hat{r}_{uj}^f = \frac{\sum_{l \in L_j^f \cap L_u^f} v_{ul}^f}{\left|I_{ul}^f\right|} \tag{6.8}$$

- Ponderada. En la propuesta simple todas las puntuaciones predichas oscilarán entre la máxima y la mínima puntuaciones que aparecen en el perfil del usuario. Esta aproximación intenta resolver este problema y tiene en consideración tanto la puntuación media de los productos del perfil como la puntuación media del producto para el que se está calculando la puntuación. Así, los valores numéricos del perfil de puntuaciones no serán puntuaciones sino que diferencias entre puntuaciones y medias de puntuaciones. Es decir, normaliza las puntuaciones con respecto a la media, tal y como figura en la Ecuación 6.9.

$$v_{ul}^f = \frac{\sum_{i \in I_{ul}} r_{ui} - \bar{r}_i}{\left|I_{ul}^f\right|} \tag{6.9}$$

Las puntuaciones de contenido se calculan de acuerdo a la Ecuación 6.10.

$$\hat{r}^f_{uj} = \bar{r}_j + \frac{\sum_{l \in L^f_j \cap L^f_u} v^f_{ul}}{\left|I^f_{ul}\right|} \tag{6.10}$$

- Z-score. Esta aproximación no sólo tiene en cuenta la media, sino que también la desviación típica. Las Ecuaciones 6.11 y 6.12 muestran cómo se realizan los cálculos para esta propuesta.

$$v^f_{ul} = \frac{\sum_{i \in I_{ul}} \frac{r_{ui} - \bar{r}_i}{\sigma_{r_i}}}{\left|I^f_{ul}\right|} \tag{6.11}$$

$$\hat{r}^f_{uj} = \bar{r}_j + \sigma_{r_j} \frac{\sum_{l \in L^f_j \cap L^f_u} v^f_{ul}}{\left|I^f_{ul}\right|} \tag{6.12}$$

6.3. Metodología y experimentos

Los experimentos han sido ejecutados en un dominio de recomendación de películas usando el *dataset* de Netflix [Bennett y Lanning, 2007] ya empleado en otras ocasiones en esta tesis y analizado en la Sección 4.3. Descargamos la información de contenido de Internet Movie Database[1]. Sin embargo, ya que no todas las películas en Netflix tenían su correspondencia en IMDb, se filtró Netflix, tal y como sucedió en el capítulo anterior. De esta manera, el conjunto de datos final está formado por 8,362 películas y 478,458 usuarios, que han dado 48,715,350 puntuaciones entre 1 y 5, ambos inclusive. Fueron eliminados los valores de contenido que sólo aparecían una vez en el conjunto de datos.

En nuestros experimentos hemos seleccionado aleatoriamente 1000 usuarios para la evaluación. Hemos simulado el problema del nuevo usuario usando una estrategia Given-N [Breese et al., 1998]. Más específicamente, hemos escogido aleatoriamente dos puntuaciones por cada usuario de evaluación. Para completar el conjunto de entrenamiento, de los usuarios restantes hemos utilizado un 10 % de sus puntuaciones, para simular así una situación de sistema nuevo (ver Sección 4.2). De este modo podemos evaluar cómo se comportan los algoritmos con poca información sobre los usuarios y, por tanto, con datos muy dispersos.

[1] http://www.imdb.com

6.3.1. Evaluación y métricas

En esta sección comparamos la técnica de expansión basada en contenido con las técnicas de expansión básicas presentadas en el capítulo anterior que mejores resultados obtuvieron en [Formoso et al., 2013a], es decir, la técnica global basada en productos y la local basada en usuarios. La primera de ellas busca productos similares a aquéllos que pertenecen al perfil del usuario actual. Para este propósito se emplea un algoritmo kNN basado en productos. Un producto formará parte de un vecindario si ha sido valorado de forma similar por un número mínimo de usuarios en común con el producto actual. Por otra parte, las técnicas locales basadas en usuarios escogen los productos para expandir entre aquéllos que los vecinos del usuario actual han valorado. En particular, en este capítulo emplearemos la aproximación de los productos más puntuados, es decir, se expandirá el perfil del usuario con los productos más valorados en el vecindario. Además, estudiamos también el comportamiento de un algoritmo kNN basado en usuarios sin expansión de perfil.

Como medida de similitud para la selección de vecinos usamos la distancia coseno, y para la predicción de las puntuaciones una aproximación ponderada, tanto en los algoritmos de expansión como en los de recomendación, siempre y cuando fuese necesario. Tras un proceso de validación cruzada, escogimos $k = 25$ como número de vecinos.

Como se mencionó previamente, nos centramos en la tarea de recomendación. Cuando evaluamos los algoritmos de recomendación, aislar un problema concreto supone una ardua tarea. Analizar cómo se comporta el algoritmo en las distintas fases del proceso de recomendación (expansión y recomendación) puede ayudar a determinar de qué problema concreto se trata. Más aún, cuando el sistema es nuevo y contiene muy poca información.

6.3.1.1. Expansión

Un perfil de usuario se expande por medio de un algoritmo de expansión. Sea cual sea el tipo de algoritmo, devolverá una lista de productos con elementos adicionales para el perfil de usuario. A estos productos se les asignará una puntuación. Por tanto, las técnicas de expansión de perfil proporcionan a los algoritmos de recomendación un perfil en el que la información acerca del usuario ha sido incrementada. Dicha información puede presentar errores, lógicamente. Para simplificar la identificación de la fuente de esos errores, hemos identificado dos subfases en el proceso de expansión:

- La selección de productos. Para la evaluación de la expansión de perfil, la relevancia o no relevancia de un producto viene determinada por su presencia o su

ausencia en el perfil del usuario, con independencia de su puntuación. Nosotros pretendemos obtener un perfil expandido con tantos productos en común con el perfil del usuario como sea posible. Debido a estas razones, hemos considerado la métrica de Precision-at-n ($P@n$), siendo n el tamaño de la expansión, para asegurar la calidad de la selección de los productos.

- La estimación de las puntuaciones. Otro de nuestros objetivos es determinar cuán precisa es la asignación de puntuaciones y cómo evoluciona esta precisión según varía la lista de expansión. Aunque se pueden usar varias métricas para medir la precisión, usaremos la popular métrica MAE. Lógicamente, no hemos tenido en cuenta para esta métrica aquellos productos que no pertenecen al perfil del usuario, ya que su error absoluto no podría ser calculado.

6.3.1.2. Recomendación

A partir del perfil obtenido con el algoritmo de expansión, el algoritmo de recomendación produce una lista de sugerencias. De nuevo emplearemos la métrica $P@n$, es decir, el ratio de productos relevantes en las primeras n posiciones de la lista de recomendación. Pensamos que esta métrica es la que mejor se ajusta a nuestras necesidades, porque en el sistema no estamos interesados en saber cuán precisa es la asignación de puntuaciones para todos los productos, sino en saber si los productos mostrados al usuario son realmente relevantes o no. A diferencia de la evaluación en la expansión en donde considerábamos relevantes todos los productos presentes en el perfil original del usuario, fuese cual fuese su puntuación, en la recomendación sólo consideramos relevantes aquellos productos cuya puntuación supere el valor de 4, nuestro umbral de relevancia.

6.4. Resultados

Como ya se ha expuesto previamente, el proceso de recomendación involucra dos pasos: la expansión y la recomendación. A continuación mostramos los resultados obtenidos en cada fase.

6.4.1. Expansión

El algoritmo de expansión es una técnica basada en contenido. La Figura 6.1 muestra la precisión y el *recall* en expansión para las diferentes características. Como se ha explicado en la Sección 6.2.3, consideramos dos medidas de similitud: la distancia coseno y la medida ponderada. Se puede ver cómo las curvas para las características de género

y país son mejores que las curvas para las características de reparto y director. Es decir, las características de género y país se repiten más frecuentemente que las otras dos características en el perfil de usuario, como ya había sido confirmado en el análisis manual.

La medida de similitud ponderada funciona ligeramente mejor que la distancia coseno, a pesar de ser más simple. En situaciones extremas, en donde la información es escasa y el perfil del usuario no está bien definido, funciones tradicionales como la distancia coseno o la correlación de Pearson son demasiado estrictas y conducen a resultados erróneos, como ya explicamos en el Capítulo 4. Debido a este hecho, la medida de similitud ponderada será usada de aquí en adelante.

La Tabla 6.1 muestra la precisión de las expansiones calculadas con las aproximaciones basadas en contenido y con las puramente colaborativas. Como se puede ver en los resultados, la precisión varía con el tamaño de la expansión del perfil. De hecho, cuánto más grande es el tamaño de la expansión, más pequeña es la precisión. Esto tiene sentido porque los primeros elementos en la lista de expansión son los que reciben un mayor peso por parte del algoritmo. Por tanto, estos resultados implican que los algoritmos están funcionando correctamente.

Por otro lado, con la Tabla 6.1 también podemos estudiar el comportamiento de

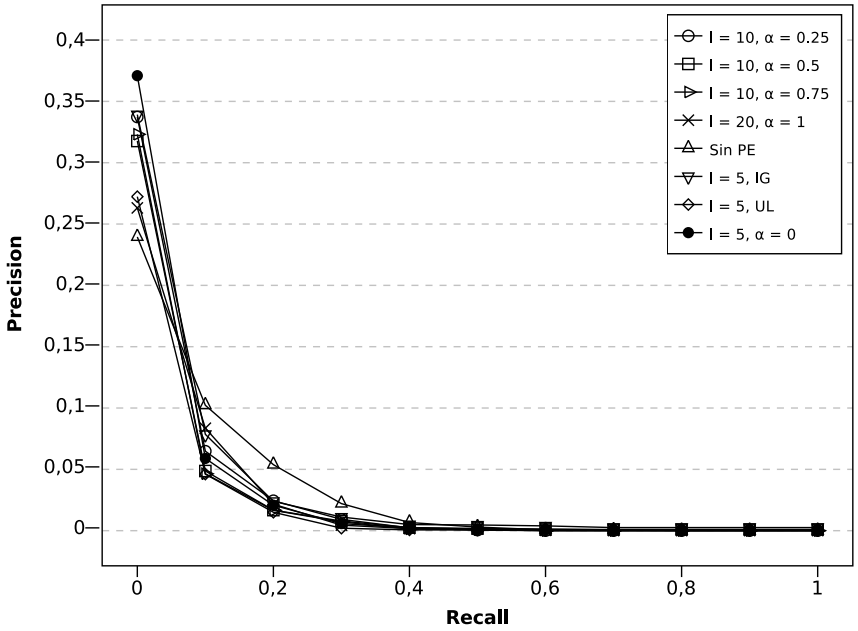


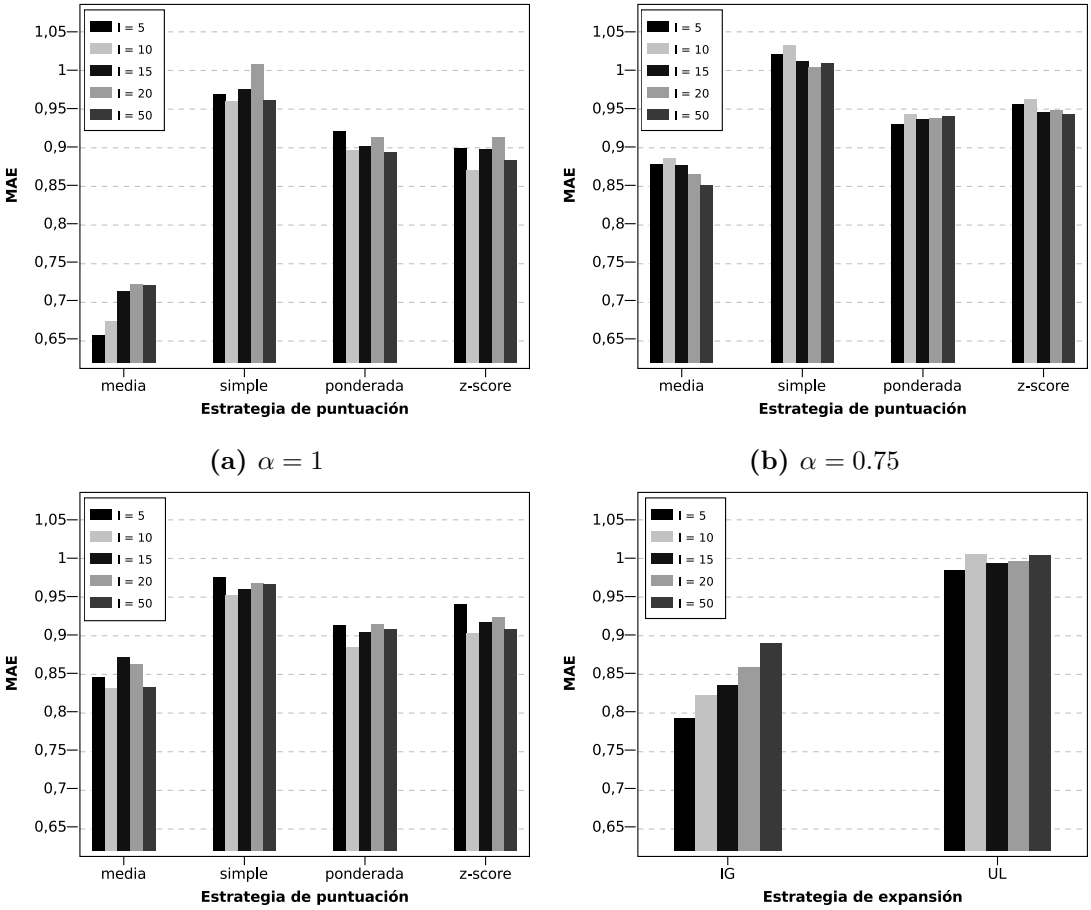
Figura 6.1: Curvas P-R en expansión para diferentes características.

la aproximación basada en contenido conforme varía α, el parámetro que controla la contribución de los pesos de contenido y popularidad. Con $\alpha = 1$ (sin popularidad) nuestra aproximación basada en contenido presenta una baja precisión. Aunque realmente estamos expandiendo con productos similares a aquellos que figuran en el perfil del usuario, estos productos son raros en el conjunto de datos. Por tanto, la precisión se reduce. De hecho, cuando se tiene en cuenta la popularidad, esta precisión se incrementa significativamente. Por ejemplo, con $\alpha = 0.75$ (es decir, 75 % de información de contenido y 25 % de popularidad) la información de contenido todavía tiene una gran importancia en el peso final del producto. Pero, al estar usando la popularidad, todos los empates en peso a la hora de escoger los productos con los que expandir son resueltos en favor de aquéllos que son más populares. De este modo, lógicamente, la precisión se incrementa, porque los productos populares tienen más probabilidad de aparecer en los perfiles de los usuarios de evaluación.

l	$\alpha = 1$	$\alpha = 0.75$	$\alpha = 0$	IG	UL
5	0.032	**0.349**	**0.401**	**0.350**	0.266
10	0.019	0.312	**0.365**	0.324	0.236
15	0.017	0.295	**0.348**	0.308	0.227
20	0.014	0.277	0.334	0.299	0.215
50	0.014	0.218	0.291	0.256	0.184

Tabla 6.1: Precisión en expansión para diferentes técnicas de expansión de perfil, según el número de productos añadidos al perfil, l. Destacamos los mejores valores.

Con respecto a la precisión en la predicción de la puntuación, en la Figura 6.2 se muestra que el mejor MAE en expansión se alcanza con la estrategia colaborativa, independientemente del valor de α. Además, con α igual a uno la media colaborativa incluso mejora el MAE obtenido con las aproximaciones colaborativas, en donde la técnica global presenta un buen MAE (principalmente cuando $l = 5$). Las estrategias z-score y ponderada obtienen mejores resultados que la estrategia simple. Para entender la razón de las diferencias entre las propuestas, analizamos individualmente cómo son las puntuaciones de cada usuario del conjunto de evaluación, confirmando que la propuesta colaborativa y las basadas en contenido se comportan de modo diferente. Con $\alpha = 0.75$ y $\alpha = 0$, mientras que la estrategia colaborativa tiende a subestimar las puntuaciones reales, el resto de estrategias tienden a sobreestimar. Con $\alpha = 1$, este comportamiento varía, porque sólo se considera la información de contenido para seleccionar los productos para expandir. En este caso, la estrategia colaborativa y la



(a) $\alpha = 1$ **(b)** $\alpha = 0.75$

(c) $\alpha = 0$ **(d)** Colaborativa

Figura 6.2: MAE en expansión para diferentes técnicas de expansión de perfil, según el número de productos añadidos al perfil, l.

estrategia simple sobreestiman, mientras que tanto las estrategia ponderada como la z-score subestiman.

6.4.2. Recomendación

Una vez evaluado que el algoritmo de expansión, el siguiente paso es evaluar el algoritmo de recomendación.

La Tabla 6.2 muestra los resultados obtenidos en P@5 por la aproximación basada en contenido con $\alpha = 1$. Se mejora ligeramente la precisión obtenida por el algoritmo kNN sin expansión de perfil. La estrategia z-score con una expansión de 20 productos ($l = 20$) es el mejor caso. En particular, esta estrategia para la asignación de puntuacio-

nes, aparte de usar información de contenido, también usa información colaborativa. Sin embargo, la expansión global basada en productos y la expansión local basada en usuarios mejoran estos resultados. La razón es que no sólo usan información colaborativa, sino que también los algoritmos son más complejos. Aunque la información colaborativa es importante para determinar la calidad de un producto, estos resultados muestran que la información de contenido para la estimación de puntuaciones es también útil. De hecho, las Tablas 6.3 y 6.4 confirman esto, ya que las aproximaciones de contenido presentan las mejores precisiones. Sin embargo, aunque la información de contenido ayuda en la estimación de las puntuaciones, las peores precisiones se obtienen cuando la información de contenido es la única que se usa para la selección de los productos.

l	Media	Simple	Ponderada	Z-score	IG	UL
5	0.103	0.100	0.096	0.096	**0.181**	0.161
10	0.102	0.099	0.101	0.100	0.165	0.161
15	0.101	0.101	0.104	0.103	0.149	0.154
20	0.107	0.101	0.109	0.112	0.133	0.138
50	0.089	0.076	0.090	0.091	0.110	0.120
Sin PE	0.100	0.100	0.100	0.100	0.100	0.100

Tabla 6.2: P@5 en recomendación para diferentes técnicas de expansión de perfil, según el número de productos añadidos al perfil, l, con $\alpha = 1$. Destacamos los mejores valores.

l	Media	Simple	Ponderada	Z-score
5	0.178	0.172	0.178	0.178
10	0.176	**0.182**	**0.183**	**0.184**
15	0.167	0.164	0.172	0.170
20	0.129	0.125	0.126	0.128
50	0.117	0.118	0.121	0.121

Tabla 6.3: P@5 en recomendación para diferentes técnicas de expansión de perfil, según el número de productos añadidos al perfil, l, con $\alpha = 0.75$. Destacamos los mejores valores.

La pregunta que surge ahora es si la información de contenido ayuda en la selección de los productos. Las características de contenido no pueden identificar la calidad. Cuando dos películas comparten los mismos valores para todas sus características de contenido, no es posible distinguir una película buena de una mala. Así, sería útil considerar más datos para ayudar a estas técnicas a determinar cuándo añadir un producto concreto a la lista de expansión. Por tanto, decidimos considerar información

l	Media	Simple	Ponderada	Z-score
5	**0.228**	**0.225**	**0.233**	**0.230**
10	0.167	0.190	0.187	0.186
15	0.116	0.171	0.161	0.160
20	0.074	0.117	0.098	0.099
50	0.141	0.120	0.115	0.117

Tabla 6.4: P@5 en recomendación para diferentes técnicas de expansión de perfil, según el número de productos añadidos al perfil, l, con $\alpha = 0$. Destacamos los mejores valores.

colaborativa, y, más concretamente, popularidad.

En la Sección 6.2.4.1 se mencionó la popularidad como un modo de mejorar la expansión en situaciones de cold-start. Las Tablas 6.3 y 6.4 confirman esto. Con $\alpha = 0.75$ los resultados son incluso mejores a los que presentan los algoritmos globales basados en productos. De hecho, cuanto más grande es el tamaño de expansión, más similares son los resultados a los obtenidos por el algoritmo que considera sólo popularidad.

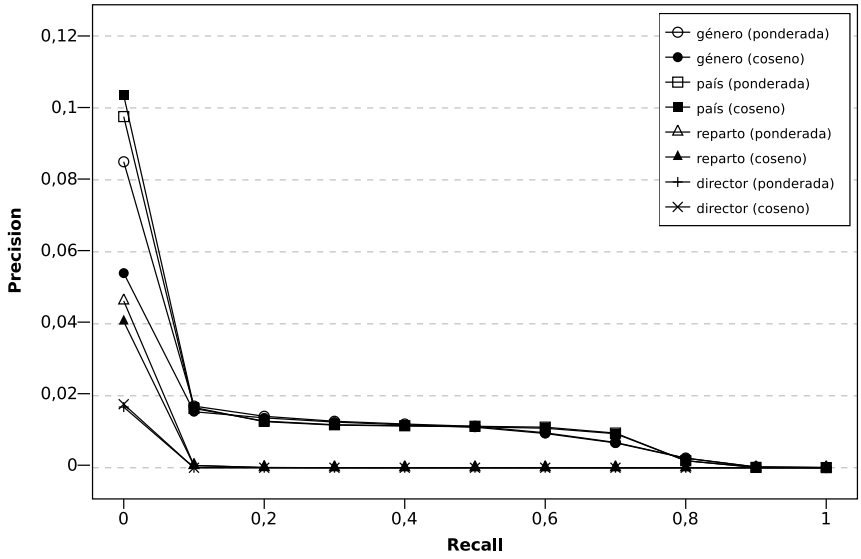


Figura 6.3: Curvas P-R en recomendación para distintos algoritmos con su número óptimo de productos añadidos al perfil, l.

La Figura 6.3 también muestra la mejora obtenida con popularidad. Se puede comprobar que cuanto mayor peso se le da a la popularidad, mayor es la precisión de la recomendación. Sin embargo, es necesario estudiar la razón por la que esto sucede. Analizando el conjunto de datos, es posible darse cuenta de que algunas películas (popu-

lares) han sido valoradas por cerca del 50 % de los usuarios. Estos productos populares pueden ser recomendados, siempre y cuando sus puntuaciones alcancen el umbral de relevancia (producto popular no implica necesariamente buen producto). Si sólo se recomiendan estos productos, se obtienen buenos resultados, ya que es muy probable que formen parte de los perfiles de los usuarios. De hecho, en la expansión cuando $\alpha = 0$ (sólo popularidad), la precisión alcanza el valor de 0.60 cuando el *recall* es muy bajo. No obstante, esta expansión puede causar que el algoritmo de recomendación se sesgue hacia los productos más populares. Los resultados dejan de estar personalizados. Esta alta precisión se debe a la limitación de la evaluación *offline*, porque un producto sólo se considera relevante si aparece en el perfil del usuario.

Por otro lado, usando $\alpha = 1$ los resultados son ligeramente mejores que con el algoritmo sin expansión de perfil, y muy similares a los obtenidos con el algoritmo local basado en usuarios, cuando el *recall* es bajo. En realidad, en la tarea de recomendación sólo una pequeña cantidad de productos relevantes se le mostrará al usuario actual. Así, aunque es importante la curva en su totalidad, se debe poner especial atención a la primera parte de ella. Si comparamos las curvas del algoritmo sin expansión con las curvas para los algoritmos con expansión se puede ver cómo la primera decrece más lentamente que las otras.

Finalmente, es importante destacar otro aspecto en las diferencias entre el comportamiento con $\alpha = 0$ y con $\alpha = 1$. Con $\alpha = 0$ el algoritmo alcanza una precisión de 0.40 en expansión, la cual disminuye en recomendación (0.22). Por otro lado, con $\alpha = 1$ la precisión es de 0.03 en expansión, pero se incrementa hasta 0.11 en recomendación. Además, el MAE que se obtiene en expansión es ligeramente mejor cuando el algoritmo no usa popularidad, como ya se mencionó previamente. Así, a pesar de alcanzar altas precisiones en expansión, los productos usados para la expansión con popularidad no están personalizados, y, en consecuencia, al usar el algoritmo kNN la precisión en la recomendación decrece. En contraste, aunque al usar sólo información de contenido la precisión en expansión es baja, esta precisión se incrementa en la recomendación, ya que los productos son similares a los del perfil del usuario. Es decir, algunos productos podrían ser considerados relevantes a pesar de no estar en el perfil del usuario. Sin embargo, la evaluación *offline* no permite distinguir estos casos.

6.5. Conclusiones

La expansión de perfil puede ser vista como un conjunto de técnicas para aliviar el problema del usuario nuevo. En este capítulo hemos mostrado cómo estas técnicas pue-

den ser usadas para abordar también el problema de un nuevo sistema. Además, hemos identificado las diferentes fases del proceso de expansión de perfil y cómo evaluarlas. De hecho, el análisis de cada fase ayuda a simplificar el análisis de la siguiente.

Particularmente, hemos propuesto una técnica basada en contenido en la que las características han sido seleccionadas usando un análisis manual. Se han empleado varias estrategias para calcular los pesos y las puntuaciones para los productos de la expansión. Se ha usado una similitud ponderada para cada característica. Además, las estrategias basadas en contenido para calcular las puntuaciones funcionaron ligeramente mejor en recomendación que la media de las puntuaciones, la alternativa colaborativa considerada. Por tanto, hemos mostrado cómo la información de contenido puede ser útil no sólo para calcular la lista de expansión, sino también para calcular las puntuaciones.

Por otro lado, también hemos mencionado la popularidad como una característica adicional. Hemos mostrado que incluir esta característica mejora los resultados, pero conduce a ciertos problemas. Primero, sesga los resultados hacia los productos más populares. Además, los resultados no están personalizados. De este modo, sería interesante llegar a un consenso entre popularidad y las características de contenido según el nivel de personalización deseado en los resultados. La popularidad puede ser adecuada cuando el problema del nuevo sistema sea extremo y los perfiles de usuario sean demasiado difíciles de obtener.

En cuanto a los trabajos futuros, se podría llevar a cabo un análisis más profundo de las diferencias entre expandir sólo el usuario actual y realizar una expansión global. Algunos aspectos relacionados con los algoritmos kNN, como las medidas de similitud o el número de vecinos, pueden ser también estudiados no sólo en el algoritmo recomendador, sino también en los algoritmos de expansión para distintas aproximaciones.

Parte III

Un dominio diferente: la recomendación de consultas

Capítulo 7

Introducción a la recomendación de consultas

En esta segunda parte de la tesis estudiaremos cómo se comportan los algoritmos de filtrado colaborativo en un nuevo entorno: la recomendación de consultas. Para ello, en primer lugar es necesario introducir algunos conceptos básicos de este nuevo ámbito, así como las principales contribuciones del estado del arte.

Hay que tener en cuenta que en la actualidad existe una cantidad ingente de información disponible en la web que evoluciona día a día. Sin embargo, aunque pública, dicha información es desestructurada. Es por ello que son necesarias herramientas para manejar, recuperar y filtrar toda esta información disponible. Una de estas herramientas son los motores de búsqueda o buscadores web, que permiten, entre otras funciones, consultar qué páginas web están relacionadas con un determinado tema. Sin embargo, no siempre es sencillo formular la consulta exacta que permite obtener la información deseada.

Por otro lado, el principal objetivo de un motor de búsqueda es ayudar al usuario a satisfacer sus necesidades de información de un modo eficiente. En este sentido, cualquier ayuda proporcionada al usuario para reducir el tiempo invertido tiene una gran importancia. De hecho, los principales motores de búsqueda, además de poder responder a las peticiones en unos pocos cientos de milisegundos, normalmente ofrecen al usuario algunas sugerencias, en forma de consultas, que están de algún modo relacionadas con la información que el usuario necesita. Estas *sugerencias* tienen como finalidad dirigir al usuario en la dirección correcta y en la mayoría de los casos ahorrarle tanto tener que realizar un excesivo número de consultas como consumir demasiado tiempo en la búsqueda. Las recomendaciones suelen estar semánticamente relacionadas con la consulta original. Se obtienen principalmente mediante modelos entrenados con

la información presente en *query logs*, que no son más que registros de la actividad de los diferentes usuarios en el propio motor de búsqueda. En general, podemos recibir recomendaciones para una consulta cuando ésta ya haya sido previamente enviada al motor de búsqueda por cualquiera de los usuarios del servicio.

En la siguiente sección explicaremos qué es un *query log* y qué puede aportar su estudio. También nos centraremos en cómo son los procesos de búsqueda llevados a cabo por los usuarios en un motor de búsqueda. A continuación, describiremos los tipos de consultas existentes. Después, indicaremos diferentes modos de identificar las tareas que un mismo usuario ha llevado a cabo en un buscador a partir de la información contenida en un *query log*. Finalmente, abordaremos cómo el buscador web ayuda al usuario por medio de recomendaciones de consultas.

7.1. *Query logs*

Las grandes compañías de búsqueda web mantienen ficheros de *log* con información acerca de la interacción de los usuarios con el propio motor de búsqueda. Dicha información incluye principalmente un identificador de usuario o de la sesión del navegador (obtenido generalmente a partir de una autenticación previa o una *cookie* [Kristol y Montulli, 1997], respectivamente), las diferentes consultas enviadas y un *timestamp* que indica el momento en el que fueron realizadas dichas consultas. Aparte de ello, también puede incluir contenido adicional como el número de páginas devueltas por el motor de búsqueda ante cada una de las búsquedas, las distintas URLs sobre las que el usuario ha hecho clic, así como la posición de las mismas entre los documentos devueltos, pequeños fragmentos del contenido de las páginas o *snippets*, etc.

Los motores de búsqueda se pueden mejorar con el análisis de los *query logs* [Baeza-Yates, 2004, 2005; Silvestri, 2010]. El análisis de la distribución de las consultas (y sus términos), el tiempo de llegada de cada consulta, los resultados con clic, etc. pueden permitir comprender mejor cómo son los procesos de búsqueda, qué tipos de consultas se realizan, con qué finalidad, la longitud media de las consultas, el número de resultados visitados, etc. Pero no sólo eso. También permite el desarrollo de nuevas técnicas de sugerencia de consultas para hacer más sencillo el proceso de búsqueda a los usuarios (ver la Sección 7.5), técnicas de reclasificación de los documentos que forman parte del resultado de una búsqueda determinada [Joachims, 2002], técnicas para realizar caché de consultas [Markatos, 2000; Puppin et al., 2006; Shokouhi et al., 2007], índices de búsqueda [Baeza-Yates, 2004] y mucho más.

En la literatura podemos encontrar trabajos acerca del estudio de diversos *query*

logs desde finales de los años 90: AltaVista [Silverstein et al., 1999], Excite [Wolfram, 2000], AlltheWeb [Jansen y Spink, 2005], TodoCL [Spink et al., 2001, 2002], AOL [Beitzel et al., 2007], Yahoo! [Skobeltsyn et al., 2008], Dogpile [Jansen et al., 2007c], etc. Esta información disponible para la comunidad científica dista mucho de ser todo lo abundante y reciente que desearíamos. Por norma general, los investigadores cuentan con pequeños *query logs* públicos que algunas compañías hacen accesibles y que abarcan periodos de tiempo generalmente pequeños, con no demasiados usuarios. Dos ejemplos entre los pocos *query logs* públicos disponibles son los *query logs* de Microsoft y AOL. En enero de 2006 Microsoft Research publicó un *query log* de alrededor de 15 millones de consultas recogidas a lo largo de un mes. En ese mismo año, AOL publicó otro *query log* con más de 30 millones de consultas enviadas por 650000 usuarios a lo largo de 3 meses. Sin embargo, este *query log*, pensado para la comunidad científica, presentaba un importante problema: vulneraba la intimidad de muchos de sus usuarios, puesto que, aunque no identificaban a los usuarios directamente, gracias al contenido de las propias consultas era posible identificar a muchos de ellos. Esto hizo que la compañía retirase los datos, si bien ya había sido replicado y distribuido en numerosos sitios web.

7.2. Proceso de búsqueda

Antes de pensar en cómo recomendar consultas a un usuario es necesario pensar en cómo realiza una búsqueda una persona. De hecho el comportamiento de los usuarios y su actividad diaria puede ser revelada a partir de las consultas que envían a los motores de búsqueda [Jansen y Spink, 2006; Lee et al., 2005].

En el artículo [Marchionini, 2006] el autor distingue distintos tipos de búsqueda: la búsqueda de información y la búsqueda exploratoria. El primer tipo se satisface rápidamente con pequeñas cantidades de información (es decir, fechas, números, nombres, etc.) y referencia a ejemplos de búsqueda simple como el tiempo que va a hacer hoy en nuestra ciudad, la fecha de un determinado evento, etc. El segundo se puede, a su vez, subdividir en tareas de aprendizaje y tareas de investigación. Las tareas de aprendizaje suponen un esfuerzo por parte del usuario, que requiere de varias interacciones con el motor de búsqueda (por tanto, de más tiempo) para poder comprender la información recuperada. Las tareas de investigación, sin embargo, requieren generalmente largos periodos de tiempo y múltiples interacciones con el motor de búsqueda.

En [O'Day y Jeffries, 1993] los autores determinaron que el proceso de búsqueda de información consta a su vez de diversas búsquedas. Además, los resultados de una búsqueda suelen acarrear nuevas necesidades de información y, por consiguiente, nuevas

búsquedas en distintas direcciones.

Tal y como se indica en [Baeza-Yates y Ribeiro-Neto, 2011], muchos investigadores han desarrollado modelos de cómo los usuarios realizan su tarea de búsqueda. El modelo clásico descrito en [Sutcliffe y Ennis, 1998] es un modelo cíclico que está formado por las siguientes actividades: identificación del problema, expresión de las necesidades de información, formulación de la consulta y evaluación de los resultados. El usuario, cuando identifica el problema, envía su consulta al motor de búsqueda. Generalmente, el texto introducido no supera las tres palabras [Jansen et al., 2005, 2007c]. La eficiencia de los motores de búsqueda de hoy en día permite a los usuarios comenzar el proceso de búsqueda con consultas genéricas e ir refinándolas en función de los resultados obtenidos hasta lograr la información deseada [Bates, 1990; Hertzum y Frøkjær, 1996; Marchionini, 1995]. De esta manera, el usuario intenta averiguar si el motor de búsqueda es capaz de proporcionarle los resultados necesarios. En caso de que esto no sea así, el usuario reformula la consulta. Si los resultados son prometedores, el usuario tiende a acceder a las páginas que le parecen más relevantes y a refinar la búsqueda en los propios sitios a los que accede, siguiendo una estrategia bastante común conocida como *orienteering* u orientación [O'Day y Jeffries, 1993; Teevan et al., 2004], puesto que el usuario va orientando y reformulando la consulta hacia el propósito que busca. Se ha demostrado que la combinación de las consultas al motor de búsqueda junto con la estrategia de orientación es la solución que mejor resultados obtiene [Bilal, 2002; Pollock y Hockley, 1997].

El modelo estándar de búsqueda de información tiene como suposición subyacente que la necesidad de información es estática, y el proceso de búsqueda consiste simplemente en un refinamiento continuo de una consulta hasta encontrar la información deseada [Baeza-Yates y Ribeiro-Neto, 2011]. Modelos más recientes se inclinan por una aproximación más dinámica, en donde el usuario aprende conforme busca, y la necesidad de información se va ajustando en concordancia. Este proceso dinámico se conoce también como modelo de búsqueda de la *recolección de bayas* [Bates, 1989].

Otros modelos de búsqueda como los presentados en [Bates, 1979; Pirolli, 2007; Russell et al., 1993] caracterizan los procesos desde el punto de vista de las estrategias y cómo son las tomas de decisión de los usuarios.

Otro aspecto importante a tener en cuenta en un proceso de búsqueda, independientemente del modelo definido, es que los usuarios a menudo buscan información que ya han buscado previamente, y esto hace que su comportamiento a la hora de realizar la búsqueda sea diferente [Barreau y Nardi, 1995; Jones et al., 2002; Milic-Frayling et al., 2004].

Por otro lado, existen trabajos [Hölscher y Strube, 2000; Lazonder et al., 2000; White et al., 2005] que abordan las diferencias en las estrategias de búsqueda entre un usuario novicio y uno experto en el asunto, así como la relación entre el dominio del proceso de búsqueda, y el dominio de un determinado tema [Christine Jenkins y Wiedenbeck, 2003]. De este modo, la persistencia en la búsqueda [Patterson et al., 2001] o el tiempo dedicado a ella pueden determinar el éxito o el fracaso final a la hora de encontrar la información deseada [Tabatabai y Shore, 2005].

7.3. Consultas

Los usuarios, cuando hacen uso de un motor de búsqueda, emplean diferentes tipos de consultas. La clasificación más clásica y popular [Broder, 2002] (refinada más tarde en [Rose y Levinson, 2004]) agrupa dichas consultas en informacionales, navegacionales y transaccionales. Las consultas informacionales tienen como objetivo satisfacer una necesidad de información. Con las consultas navegacionales se busca encontrar la dirección de un sitio web para que así el usuario puede navegar en él. Por su parte, cuando un usuario emplea una consulta transaccional lo que busca es realizar alguna acción interactiva como puede ser comprar una entrada para un concierto, descargar un libro, etc. El problema que surge es que no es fácil determinar cuál es la intención de una consulta concreta, y en muchos casos es ambiguo. Por ejemplo, cuando escribimos el nombre de un equipo de baloncesto, no se sabe si queremos obtener información acerca de la historia del equipo, acceder a su página web, o ver un partido en streaming. Es por ello que se han desarrollado diferentes técnicas para determinar automáticamente la intención de una determinada consulta [Baeza-Yates et al., 2006; Jansen et al., 2007a; Kang y Kim, 2003; Lee et al., 2005]. Asimismo, como ya se ha dicho, existen búsquedas que el usuario realiza frecuentemente. Las consultas que el usuario repite una vez tras otra, sobre todo consultas navegacionales, se conocen como consultas de rebúsqueda [Jones et al., 2001, 2002; Teevan et al., 2004] y suponen un problema a la hora de predecir el comportamiento de un usuario.

Otra característica importante a partir de la cual se pueden clasificar las consultas es el tema de las mismas. Aunque se han desarrollado diversas técnicas para clasificar las consultas según su tema [Broder et al., 2007], la propia ambigüedad inherente al lenguaje humano, la falta de rigor a la hora de escribir una consulta, etc. hacen esta tarea complicada. De hecho, cuando se tienen necesidades que satisfacer o tareas que realizar, en determinadas ocasiones expresarlas por medio de una consulta resulta complicado. A menudo, dicha expresión sólo engloba parte de las necesidades, por lo que se convierte

en algo imperfecto. Es un reto para el usuario ser capaz de elaborar una consulta que conduzca a los resultados que busca. En el mejor de los casos, si el usuario es capaz de expresar con perfección su consulta, aparece otra dificultad: interpretar correctamente los resultados obtenidos.

Hay estudios que demuestran lo difícil que es para los usuarios determinar la relevancia de un documento para un determinado tema, sobre todo cuando el usuario sabe poco sobre él [Croft et al., 2001; Ruthven y Lalmas, 2003; Spink et al., 1998a; Vakkari, 2000; White et al., 2005]. Es por ello que a menudo el ranking del documento supone el principal punto de apoyo para determinar su relevancia para los usuarios [Granka et al., 2004; Joachims et al., 2005]. De hecho, esto ha sido probado en trabajos de *eye tracking* como [Joachims et al., 2007], en donde se muestra la predilección de los usuarios por los primeros resultados, y la confianza en el orden de los resultados devueltos por el motor de búsqueda. También supone un problema determinar la cantidad de documentos relevantes accedidos y si existen más documentos relevantes en la colección [Sutcliffe y Ennis, 1998; Tabatabai y Shore, 2005].

7.4. Segmentación de sesiones

En el análisis de consultas hay que tener en cuenta que un usuario generalmente no envía al motor de búsqueda consultas independientes, sino que están relacionadas entre sí. Normalmente en un *query log* las consultas aparecen asociadas bien a un usuario, bien a un identificador de sesión. A la hora de trabajar con *query logs* a menudo no es suficiente con tener todas las consultas asociadas a un mismo usuario por lo que dichas consultas se separan en diferentes conjuntos con cierta relación. A este hecho se le denomina *segmentación*. Aunque el concepto de sesión parece intuitivo, existe divergencia en la literatura acerca de su definición. Principalmente se puede ver bien como la secuencia de consultas enviadas por un usuario durante un día, bien la secuencia de consultas enviadas por un usuario desde que inicia hasta que abandona el navegador o bien la secuencia de consultas comprendidas entre periodos de inactividad [Gayo-Avello, 2009]. Si bien nosotros nos decantamos por las definiciones que aparecen en [Hagen et al., 2013] de sesiones física y lógica, y misión de búsqueda, el concepto de *sesión* tiene muchas definiciones alternativas [Hansen y Shriver, 2001; He y Göker, 2000; He et al., 2002; Jansen y Spink, 2003; Jansen et al., 1998, 2000; Silverstein et al., 1999; Spink et al., 1998b; Wen y Zhang, 2003; Wolfram, 2000; Xiong y Agichtein, 2007]. Así, dos consultas consecutivas pertenecerán a una *sesión física* si el lapso de tiempo transcurrido entre ambas no supere un determinado umbral. Por tanto, se trata de una

definición basada meramente en el concepto tiempo. En una misma sesión física, sin embargo, pueden existir consultas que pertenezcan a diferentes necesidades de información. Por ejemplo, un mismo usuario desde un mismo navegador puede estar buscando simultáneamente información acerca de un determinado problema en su teléfono móvil e información sobre las fiestas gastronómicas en Galicia. Dichas consultas pueden estar incluso intercaladas. Aquí surge el concepto de *sesión lógica* [Baeza-Yates, 2007], también llamada cadena de consultas [Radlinski y Joachims, 2005], que representa a aquellas consultas pertenecientes a una misma sesión física que comparten una misma necesidad de información. No obstante, la finalidad de distintas sesiones lógicas de un mismo usuario puede ser igual, por lo que aparece otro concepto: la misión de búsqueda [Jones y Klinkner, 2008]. Dicho concepto engloba a todas las consultas de un usuario con la misma necesidad de información, sea cual sea el intervalo de tiempo que las separa.

En la literatura se han propuesto diversas técnicas para la segmentación de un *query log*. Principalmente estas técnicas se pueden clasificar en: basadas en tiempo, basadas en el léxico y basadas en información semántica.

7.4.1. Basadas en tiempo

Los usuarios tienden a realizar ráfagas de consultas en periodos cortos de tiempo y después dejar periodos relativamente largos de inactividad. Por tanto, para detectar los límites de sesión, en [Silverstein et al., 1999] los autores sugirieron la aplicación de los límites temporales. Este método es muy popular debido a su simplicidad y ha sido ampliamente utilizado con diferentes valores para el umbral de tiempo [Anick, 2003; Catledge y Pitkow, 1995]: 5 minutos [Downey et al., 2007], entre 10 y 15 minutos [Göker y He, 2000; He et al., 2002], entre 20 y 40 minutos [Huang et al., 2004a], entre 1 y 50 minutos [He y Göker, 2000], 30 minutos [Radlinski y Joachims, 2005], 60 (e incluso 120) minutos [Buzikashvili y Jansen, 2006]. Existen otras soluciones, como la presentada en [Chen et al., 2002], en donde se propone un umbral adaptativo que depende tanto del usuario como de la tarea.

En [Murray et al., 2006] afirmaron que la aplicación del mismo umbral para todos los usuarios no es necesariamente apropiado para cada uno de ellos en todas las circunstancias y, en consecuencia, propusieron una técnica que considera un umbral para cada usuario a través de un algoritmo basado en clasificación aglomerativa jerárquica [Jain y Dubes, 1988], donde cada consulta comienza siendo una sesión para que después gradualmente cada sesión se vaya uniendo en sesiones más generales que las engloben.

7.4.2. Basadas en el léxico

Otros investigadores han sugerido la idea de utilizar las características léxicas de las consultas para determinar si existe un cambio de tema y, por tanto, un límite en la sesión lógica. Normalmente este tipo de técnicas no se emplean solas. A menudo se combinan con técnicas basadas en tiempo.

Los patrones de búsqueda son empleados habitualmente para poder detectar un cambio de sesión lógica en este tipo de técnicas. A este respecto, se han propuesto varias clasificaciones de los patrones de búsqueda [He et al., 2002; Huang y Efthimiadis, 2009; Jansen et al., 2007d; Lau y Horvitz, 1999; Ozmutlu y Çavdur, 2005; Ozmutlu et al., 2006; Spink et al., 2000]. Dichos patrones de búsqueda indican la relación entre dos consultas consecutivas y facilitan la detección de los límites de las sesiones lógicas. Para ello los siguientes patrones son de interés:

1. Repetición. La primera y la segunda consultas son iguales.

2. Especialización. La segunda consulta trata el mismo tema que la primera, pero busca información más concreta.

3. Generalización. La segunda consulta busca información menos concreta que la primera.

4. Reformulación. Tanto la primera como la segunda consultas se refieren al mismo tema y comparten términos en común, aunque presentan también términos diferentes.

5. Nuevo. La segunda consulta trata sobre un tema diferente al de la primera.

Cuando se utilizan patrones de búsqueda para detectar sesiones lógicas hay dos enfoques principales: bien se puede suponer que el patrón de búsqueda *nuevo* implica siempre un límite de sesiones [Jansen et al., 2007b], bien se puede obtener información estadística de los *query logs* para saber la probabilidad de que este patrón de búsqueda en realidad implique un cambio de sesión lógica en función de la diferencia de tiempo entre dos consultas sucesivas [He et al., 2002]. Dada la naturaleza del patrón de búsqueda *nuevo*, el primer enfoque implica que muchas consultas relacionadas se dividen en distintas sesiones lógicas. Diferentes enfoques intentan resolver parcialmente este problema. Así, en [He y Göker, 2000; He et al., 2002] combinaron tanto patrones temporales como patrones de búsqueda para decidir si dos consultas han de pertenecer o no a la misma sesión lógica. En concreto, emplean la teoría de Dempster-Shafer sobre

probabilidades precalculadas, como también se propone en [Ozmutlu y Çavdur, 2005; Ozmutlu et al., 2006]. Por su parte, en [Radlinski y Joachims, 2005] emplearon clasificadores SVM (método de aprendizaje supervisado). Se han propuesto otros métodos para hacer tal combinación, como redes neuronales, regresión lineal múltiple, simulación de Monte Carlo, probabilidades condicionales...

Debe tenerse en cuenta que los patrones de búsqueda están determinados por medio de comparaciones léxicas usando medidas de similitud entre cadenas de texto. Cuando dos consultas no comparten ningún término en común, suele ser indicador de un cambio de tema y, por tanto, de una nueva sesión lógica [Jansen et al., 2007b]. Por ejemplo, la distancia Levenshtein [Jones y Klinkner, 2008] determina cuántos caracteres se necesitan cambiar en una determinada consulta para transformarla en otra. El coeficiente Jaccard [Lucchese et al., 2011], por su parte, mide el solape entre consultas, ya sea entre caracteres o palabras.

Así, por ejemplo, en [Shi y Yang, 2006] desarrollaron el método de la ventana de deslizamiento dinámico de segmentación que se basa en tres restricciones temporales: máximo tiempo entre dos consultas consecutivas pertenecientes a la misma sesión lógica, máximo tiempo de inactividad dentro de la misma sesión y máxima longitud de una sesión lógica simple. Empíricamente establecieron a 5 minutos, 24 horas y 60 minutos dichos valores, respectivamente. Esto significa que dos consultas sucesivas con un espacio más corto de 5 minutos deben pertenecer a la misma sesión lógica (siempre y cuando toda la sesión dure 24 horas máximo) y dos consultas con un espacio de más de 60 minutos pertenecen a diferentes sesiones. Esas consultas con un intervalo de tiempo entre 5 y 60 minutos se comparan utilizando la distancia Levenshtein para decidir si son lo suficientemente similares como para pertenecer a la misma sesión lógica o no.

Otro método diferente es el método geométrico presentado en [Gayo-Avello, 2009], un ejemplo de combinación de técnicas de tiempo y léxicas.

Los métodos léxicos presentan el problema de la discordancia en el vocabulario, es decir, la existencia de consultas relacionadas que no tienen ningún término en común. Por ejemplo, aunque dos consultas como *recuperación de información* y *Baeza-Yates* es probable que tengan el mismo objetivo, usando sólo medidas basadas en distancia léxica no sería posible determinar la relación semántica.

7.4.3. Basadas en información semántica

Las técnicas basadas en información semántica abordan el problema de la discordancia en el vocabulario. Para ello, una opción es enriquecer las consultas con información adicional. En [Shen et al., 2005] se comparan las consultas en sí con una representación

extendida de las mismas. Estas representaciones extendidas se forman a partir de la concatenación de los *snippets* de los primeros resultados proporcionados por el motor de búsqueda a una consulta específica. En el estado del arte figuran trabajos que emplean los primeros 10 resultados, 50, 100 o incluso 500 resultados. Estos métodos abarcan desde aquéllos que miden el solape entre los conjuntos de URLs, hasta la distancia coseno entre títulos de las páginas y *snippets* o incluso la distancia coseno entre los documentos en sí [Buzikashvili y Jansen, 2006] . Sin embargo, todo esto es bastante costoso. Así, otra opción, usada en [Lucchese et al., 2011], es emplear el *framework ESA* [Gabrilovich y Markovitch, 2007] para buscar las similitudes semánticas en otra colección de datos auxiliar como Wikipedia o Wiktionary. Ejemplo de grafos son los denominados query flow graphs [Boldi et al., 2008, 2009], que son modelos para representar los datos recogidos en un *query log* que permiten hacer una segmentación de las consultas aplicando el problema del vendedor viajante.

7.4.4. Detección de misiones

La detección de misiones es una tarea más compleja que la segmentación de sesiones lógicas, puesto que implica la existencia de multitareas, ya que un usuario puede realizar tareas de un modo entrelazado, incluso durante sesiones físicas diferentes. Pocos son los trabajos que abordan este problema, pudiendo destacar entre ellos [Hagen et al., 2013; Jones y Klinkner, 2008; Kotov et al., 2011; Lucchese et al., 2011, 2013].

7.5. Sugerencia de consultas

En los motores de búsqueda los usuarios no solamente tienen problemas a la hora de elaborar la consulta correctamente, sino que también al emplear algunas de las herramientas de las que disponen. Es por ello que herramientas que son presentadas al usuario y en las que su intervención es menos activa son de gran ayuda. Un claro ejemplo es la sugerencia de consultas [Baeza-Yates y Ribeiro-Neto, 2011]. De hecho, la especificación de consultas experimentó un cambio muy positivo desde que sugerencias dinámicas, también llamadas auto-completar o auto-sugerencias, fueron incluidas como parte de las interfaces de los motores de búsqueda [White y Marchionini, 2007]. La información que usan para dar este tipo de sugerencias es mínima, abarcando únicamente los caracteres que ha enviado el usuario y, en mayor o menor medida, el historial de consultas del usuario, información geográfica o características de personalización de los usuarios. Dichas sugerencias se presentan antes de que la consulta sea enviada por completo al servidor por lo que se trata de un proceso que ha de ser eficiente, así como

eficaz, teniendo en cuenta diversos aspectos como la diversidad de las propuestas, la corrección ortográfica de las consultas, la temporalidad o la personalización. También existe otro tipo de sugerencias encaminadas a enmendar errores ortográficos en las consultas, como puede ser la inversión de letras [Kukich, 1992]. Para ello se analizan los *query logs* y se aprovecha la *sabiduría de las masas* y la reformulación de consultas de usuarios previos para proporcionar las soluciones a dichos errores [Ahmad y Kondrak, 2005; Chen et al., 2007; Cucerzan y Brill, 2004; Li et al., 2006]

Por otro lado, la recomendación de consultas o *query suggestion*, se refiere a la proposición de nuevas consultas que le realiza el motor de búsqueda al usuario al haber enviado una consulta. Estas recomendaciones estarán relacionadas de alguna manera con la consulta original, de tal modo que le resulte más sencillo al usuario encontrar aquello que busca. A diferencia de las sugerencias dinámicas, en este caso se dispone de más información: consultas completamente formuladas, resultados con sus *snippets*, clics, etc. Estas fuentes de información pueden ser usadas en diferente grado para crear técnicas para la sugerencia de consultas. Así, algunas soluciones emplean el conocimiento del contexto; es decir, las recomendaciones se basan en el orden de los documentos obtenidos para medir similitudes entre consultas [Raghavan y Sever, 1995], en la intersección entre conjuntos de resultados obtenidos para diferentes consultas [Fitzpatrick y Dent, 1997], o la información obtenida a partir de los *snippets* usados a modo de consultas [Sahami y Heilman, 2006]. Otras soluciones ignoran el contenido y usan como fuente de información los documentos comunes entre consultas sobre los que se ha hecho un clic [Beeferman y Berger, 2000]. El uso de información obtenida a través de los datos de clics (*click-through data*) para medir la similitud entre consultas también se puede encontrar en [Cucerzan y White, 2007; Zhang y Nasraoui, 2006]. El problema que tiene este tipo de aproximaciones es que el número de clics por consulta es reducido [Baeza-Yates, 2005]. Existen otras alternativas que consideran el flujo de consultas. Otros trabajos usan información extraída de la sesión del usuario para presentar sugerencias [Fonseca et al., 2005; Huang et al., 2003]. En lugar de utilizar una única consulta, este tipo de aproximaciones consideran el camino de búsqueda llevado por el usuario para poder así determinar de un modo más adecuado la finalidad de una consulta determinada [Fonseca et al., 2003; Zhang y Nasraoui, 2006]. Se tienen en cuenta, por tanto, las sesiones de búsqueda. Por último, existen también propuestas híbridas [Baeza-Yates et al., 2004b,c, 2007b], que tienen en cuenta varias de las fuentes de información disponibles.

En cuanto a los tipos de técnicas, en la literatura podemos encontrar desde la sugerencia simple de las consultas que más frecuentemente se realizaron en el pasado, hasta el uso de soluciones basadas en técnicas de clustering que pueden usar, a dife-

rentes niveles, información disponible en *query logs* [Baeza-Yates et al., 2004a,c, 2007b; Beeferman y Berger, 2000; Wen et al., 2001]. Algunas se centran en obtener reglas de asociación a partir de los *query logs* [Fonseca et al., 2003]. Otros emplean la reformulación de las consultas enviadas por usuarios previos [Jones et al., 2006], o incluso dejan a los usuarios escoger las técnicas para medir la similitud entre las consultas [Zaïane y Strilets, 2002].

Finalmente, se han empleado también técnicas de expansión de consultas, como [Cui et al., 2002]. Éstas se pueden dividir en técnicas de *relevance feedback* y *query expansion*. Ambas asumen que una consulta no es más que un primer intento para conseguir el objetivo final del usuario, por lo que intentan mejorar dicha consulta añadiéndole nuevos términos. Mientras que con las técnicas de *relevance feedback*, como el algoritmo de Rocchio [Rocchio, 1971], el usuario selecciona explícitamente los resultados que considera relevantes y el motor de búsqueda, a partir de dicha información, intenta perfeccionar el conjunto de resultados devuelto, con las técnicas de *query expansion* el usuario no hace dicha selección y son necesarias otras fuentes de información, como *click-through data* [Joachims, 2002].

Cabe destacar que tanto el diseño como la evaluación de todo este tipo de algoritmos resultan ser tareas complejas. Por ejemplo, en *query suggestion*, tradicionalmente se han utilizado estudios con usuarios reales para probar el rendimiento de los métodos propuestos en la literatura. Aunque se ha considerado muy precisa la evaluación con usuarios reales, su principal inconveniente es que los experimentos darán resultados diferentes cada vez, lo cual hace difícil una comparación amplia de tales técnicas.

Diversos autores han optado por afrontar el problema partiendo de que los usuarios no están aislados, sino que pertenecen a distintas comunidades (un mismo sitio web, una misma clase de instituto, etc.), y los miembros de dichas comunidades pueden tener intereses en común. La búsqueda web colaborativa [Smyth, 2007] intenta mejorar los resultados de las búsquedas incorporando información acerca de la relevancia de resultados para una comunidad en concreto, a través de las consultas y resultados previos obtenidos para dicha comunidad. Por ejemplo, se ha propuesto la sugerencia de consultas repetidas que condujeron a resultados relevantes similares [Balfe y Smyth, 2004]. En trabajos como [Smyth, 2007; Smyth et al., 2005] se han considerado las *sesiones exitosas*, aquéllas en las que el usuario hace clic en al menos un resultado, como un método de relevancia. El análisis de clics es un modo de evaluar la relevancia de los resultados obtenidos para una determinada consulta [Joachims et al., 2007]. Estos datos son fácilmente recuperables de un motor de búsqueda. Sin embargo, al ser datos implícitos transparentes para el usuario, presentan ruido.

Capítulo 8

Análisis de un *query log*

En el Capítulo 7 se ha hecho una aproximación a los conceptos generales de los motores de búsqueda. Entre ellos se ha explicado qué es un *query log* y cómo su análisis puede ser de gran ayuda para la mejora de los motores de búsqueda. También se ha comentado que no abundan este tipo análisis en la literatura.

Antes de abordar el empleo de algoritmos de filtrado colaborativo para la recomendación de consultas es necesario comprender el comportamiento de los usuarios en el motor de búsqueda a tratar. Más aún teniendo en cuenta la gran diversidad de usuarios que lo emplean. Comprender cómo el usuario expresa sus necesidades, cómo es su proceso de búsqueda, la longitud de las sesiones, etc., es imprescindible para poder después analizar la aplicación de los distintos algoritmos y sus resultados.

8.1. *Query log*

En este capítulo realizaremos un análisis del *dataset* Microsoft 2006 RFP (MSN) proporcionado en la conferencia Web Search Click Data 2009. Este *dataset* consta de casi 15 millones de consultas pertenecientes a 7,470,916 sesiones físicas diferentes registradas a lo largo de un mes desde el buscador de EE.UU., por lo que la gran mayoría de las consultas están en inglés.

Para cada consulta figuran los siguientes datos: el momento en el que fue lanzada, la cadena de caracteres que la representa, su identificador único, un identificador único de la sesión física a la que pertenece (asociado a una *cookie*) y el número de resultados devueltos por el motor de búsqueda. En la Tabla 8.1 se muestra un fragmento de dicha información. Por su parte, de cada resultado sobre el que se ha hecho clic disponemos del identificador único de consulta al que hace referencia, de la consulta, del momento en que se hizo el clic y de la posición del documento en la lista de resultados (ranking),

tal y como figura en la Tabla 8.2. El número total de clics recogidos supera los 12 millones.

Momento	Consulta	Identificador de consulta	Identificador de sesión	Número de resultados
2006-05-23 13:33:43	bill baxton	03fce322950b4e8c	00000064723a4a55	11
2006-05-23 13:47:55	jeff daniels	4c6560e9c74d47c3	00000064723a4a55	13
2006-05-18 10:56:36	county wide home loans	f6a51307996b449e	000000e99d234b65	18
2006-05-22 12:59:57	wedding flowers	470ebf6342474cec	000000f30441477f	18
2006-05-08 12:31:18	"MIllard Cramp"	9a90d44d6e5144ea	0000012358ee4000	3
2006-05-25 11:43:33	venice beach camera	d729a151423744a1	00000179777b4251	14
2006-05-25 11:47:46	rentals venice ca 90291	ac0c2b3051384eac	00000179777b4251	22
2006-05-25 11:56:39	megans law	91d09b3130824efb	00000179777b4251	18
2006-05-23 15:52:23	yousemite	d12c30fbd89f423d	000001f79a144673	10

Tabla 8.1: Extracto de la información acerca de las consultas.

Identificador de consulta	Consulta	Momento	URL	Posición
0000003a718649f2	schwab	2006-05-11 08:07:35	http://www.schwab.com/	1
0000006d43b549c1	us geography	2006-05-04 14:23:00	http://www.enchantedlearning.com/usa/	3
0000006d43b549c1	us geography	2006-05-04 14:23:03	http://www.sheppardsoftware.com/states_experiment_drag-drop_Intermed_State15s_500.html	4
0000016aa52e4fbc	wwf	2006-05-21 09:25:34	http://www.panda.org/	2
000002aa6e27443f	biggercity	2006-05-07 13:30:45	http://www.biggercity.com/chat/	1
000005aac1f6423f	shawnee studios	2006-05-09 14:21:29	http://www.shawneestudios.com/contact_us.php	1

Tabla 8.2: Extracto de la información acerca de los clics.

8.2. Estudio de sesiones lógicas

En la Sección 7.4 comentamos que para trabajar con los datos de un *query log* suele ser habitual separar las diferentes sesión físicas en sesiones lógicas, es decir, realizar una segmentación. Entre los diferentes métodos de segmentación, el más sencillo, aunque también el menos preciso (si bien es ampliamente usado en la literatura), es el basado en tiempo. En esta sección analizaremos diferentes umbrales de tiempo, el número de sesiones lógicas resultantes, la longitud de cada una, los clics obtenidos, etc. Para ello la segmentación se realizará según el tiempo transcurrido entre dos consultas, empleando distintos umbrales (5, 10, 15, 20, 25, 30 y 60 minutos). También consideramos un caso en el que no existe umbral, es decir, el caso en el que la sesión lógica y la sesión física están formadas por las mismas consultas.

La Tabla 8.3 muestra el número de sesiones lógicas obtenido para cada uno de los umbrales. Lógicamente, cuanto mayor sea dicho umbral, menor será el número de sesiones resultantes. Por tanto, el número de consultas medio en cada sesión ha de aumentar.

Tamaño	Umbral							
	5	10	15	20	25	30	60	-
1	6,346,906	5,376,913	4,928,409	4,692,278	4,670,023	4,661,681	4,650,121	4,637,065
2	1,421,062	1,352,520	1,339,054	1,340,077	1,337,966	1,336,702	1,334,736	1,330,976
3	564,034	590,585	606,842	617,267	617,085	616,983	616,608	615,075
4	269,648	301,208	315,274	323,376	323,924	324,077	324,185	323,450
5	144,438	170,656	181,010	186,768	187,225	187,401	187,611	187,341
6-10	203,350	260,600	283,154	293,804	295,199	295,648	296,449	296,327
11-25	41,497	60,626	69,019	72,635	73,255	73,607	74,018	74,517
26-100	2,787	4,453	5,113	5,366	5,439	5,463	5,534	5,866
> 100	196	238	269	275	276	277	279	305
Total	8,993,918	8,117,799	7,728,144	7,531,846	7,510,392	7,501,839	7,489,541	7,470,922

Tabla 8.3: Número de sesiones lógicas según la longitud y umbral empleados.

Los resultados han mostrado que al menos el 50 % de las sesiones están formadas por una única consulta y que el 75 % de los sesiones, como mínimo, tienen un tamaño menor o igual que 2, sea cual sea el umbral empleado.

Analizando con detenimiento la Tabla 8.3 vemos que el número de sesiones de una sola consulta disminuye al aumentar el umbral, pero no sólo en valor absoluto sino también en su porcentaje (del 71 % al 62 %). Alrededor del umbral de 20 minutos este porcentaje se estabiliza en el 62 %. Por tanto, un porcentaje significativo de usuarios (entre el 29 % y el 38 %) intentan probar otra consulta cuando después de analizar los resultados de la primera ven que no son satisfactorios.

Por otro lado, el número de sesiones con dos consultas disminuye si aumentamos el umbral de 5 a 10 minutos, pero a partir de ahí permanece más o menos constante. Por contra, el número de sesiones de tres o más consultas aumenta al aumentar el umbral. Sin embargo, el único porcentaje que disminuye según se aumenta el valor del umbral es el de las sesiones con una consulta. Todos los demás, o aumentan o permanecen constantes al aumentar el umbral. De hecho, todos se estabilizan a partir del umbral de 20 minutos. Esto implica que la diferencia de tiempo entre dos consultas consecutivas dentro de la misma sesión física no suele ser mayor de 20 minutos; es decir, los usuarios, tras enviar una consulta, suelen tardar 20 minutos o menos en revisar los resultados del buscador antes de intentar una siguiente consulta.

Deteniéndonos en el caso del umbral de 30 minutos, habitualmente usado en la literatura [Radlinski y Joachims, 2005], se puede comprobar en la Figura 8.1 cómo la distribución del número de sesiones lógicas con respecto al tamaño de la mismas sigue una ley de potencia, es decir, $número\,de\,sesiones = c * tamaño^{-\alpha}$. De hecho, los resultados son significativos con un valor de confianza del 95 %, siendo la constante $c = 4,77 * 10^{-6}$ y el exponente $\alpha = 2,77$, con un $R^2 = 0,757$.

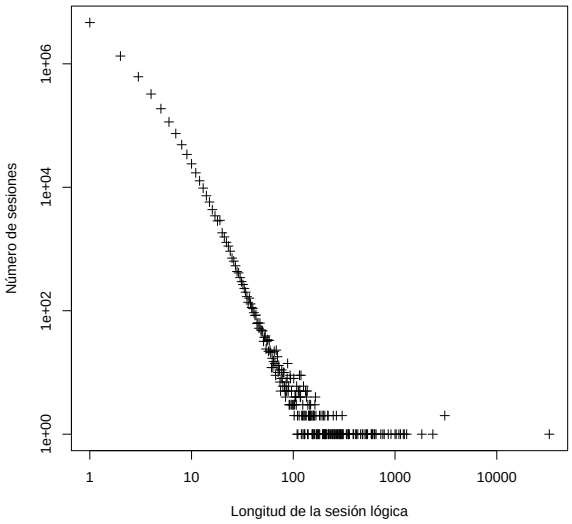


Figura 8.1: Número de sesiones lógicas por tamaño, para un umbral de 30 minutos.

8.3. Clics

8.3.1. Clics según la posición en las sesiones lógicas

Tal y como sucedía en otros trabajos ya comentados en el Capítulo 7, como [Smyth, 2007], interpretaremos como exitosa una sesión lógica si existe algún clic en su última consulta. Un clic puede ser un buen indicador de la relevancia parcial de un resultado, y, por tanto, de la consulta en sí. Es decir, el hecho de que con la consulta se haya encontrado un resultado que puede ser relevante, implica se ha formulado de un modo más o menos correcto. Por su parte, la ausencia de clics consulta indica claramente que no se ha encontrado ningún resultado relevante, y, por tanto, o bien la consulta se debería reformular, o bien los resultados relevantes se encontraban después de los analizados.

Convendría que el número de sesiones lógicas con un clic en su última consulta fuese el más elevado posible, puesto que esto indicaría que la última consulta está bien formulada y, por tanto, que el usuario ha realizado una sesión lógica satisfactoria. La Tabla 8.4 muestra el porcentaje de sesiones lógicas exitosas en función del tamaño de las mismas y del umbral de tiempo utilizado. A partir de ella se pueden realizar ciertas observaciones:

- El porcentaje de éxito se reduce conforme aumenta el tamaño de las sesiones. De

Tamaño	Umbral							
	5	10	15	20	25	30	60	-
1	72.03 %	70.91 %	70.13 %	69.63 %	69.60 %	69.59 %	69.59 %	69.57 %
2	70.81 %	71.20 %	71.51 %	71.74 %	71.77 %	71.77 %	71.78 %	71.79 %
3	67.09 %	68.26 %	68.81 %	69.07 %	69.11 %	69.12 %	69.13 %	69.15 %
4	64.44 %	65.94 %	66.58 %	66.91 %	66.93 %	66.93 %	66.96 %	66.97 %
5	62.31 %	64.03 %	64.69 %	65.01 %	65.05 %	65.05 %	65.06 %	65.08 %
6-10	59.41 %	61.73 %	62.45 %	62.81 %	62.87 %	62.90 %	62.93 %	62.95 %
11-25	50.84 %	54.95 %	56.37 %	56.85 %	56.90 %	56.93 %	56.98 %	57.15 %
> 25	32.69 %	38.86 %	41.30 %	41.85 %	42.15 %	42.25 %	42.35 %	43.74 %

Tabla 8.4: Porcentaje de sesiones lógicas exitosas según tamaño y umbral.

hecho, las sesiones lógicas de elevado tamaño tienen un bajo porcentaje de éxito. Esto puede ser debido a que estamos empleando un método de segmentación sencillo y las sesiones lógicas de gran tamaño encontradas están realmente formadas por un cúmulo de sesiones. De este modo serán exitosas únicamente cuando la última sesión lógica de cada sesión física lo sea. Otra posible razón es que realmente se trate de procesos de búsqueda complejos, que requieren de esfuerzo por parte del usuario y que no siempre conducen a buen término. El único caso en el que aumenta el porcentaje es al pasar de sesiones de una a dos consultas. Es más probable que en estas sesiones figuren consultas erróneas, o que, por otro lado, sean consultas que el algoritmo de segmentación empleado no haya sido capaz de incluir en la sesión adecuada.

- El umbral no altera significativamente el porcentaje de éxito. En general, se aprecia un ligero incremento de dicho porcentaje según aumenta el umbral empleado, estabilizándose el valor a partir de los 20 minutos. El porqué de este aumento es que, cuando usamos un umbral pequeño, podemos estar separando consultas pertenecientes a una misma búsqueda de información. Esto implica que consultas que realmente no son finales se consideren así, por lo que tenderán a ser menos relevantes y recibir menos clics, precisamente porque la información deseada se obtiene a partir de las siguientes consultas. El caso de las sesiones de una consulta difiere de los demás, en el sentido de que el porcentaje decrece según el umbral es mayor. Este decrecimiento viene motivado por la reducción del número de sesiones de este tamaño con dicha evolución, puesto que sus consultas pasan a formar parte de sesiones más largas.

A raíz de los resultados obtenidos, podemos concluir que el número de clics en las

Tamaño	Umbral							
	5	10	15	20	25	30	60	–
1	0.9671	0.9325	0.9145	0.9051	0.9036	0.9029	0.9022	0.9020
2-5	0.7761	0.8221	0.8393	0.8471	0.8478	0.8481	0.8486	0.8487
6-15	0.5866	0.6721	0.6981	0.7083	0.7104	0.7113	0.7124	0.7128
> 15	0.2976	0.4012	0.4360	0.4492	0.4526	0.4543	0.4568	0.4674

Tabla 8.5: Media de clics por consulta, según el umbral y el tamaño de la sesión lógica.

últimas consultas de las sesiones lógicas es reducido, o, lo que es lo mismo, el número sesiones lógicas exitosas es bajo, principalmente cuando son largas. Es interesante saber si este bajo porcentaje de clics sucede sólo en las últimas consultas o no, y si es dependiente del tamaño de las sesiones lógicas y/o de los umbrales de tiempo escogidos para la segmentación. La Tabla 8.5 muestra la media de clics por consulta para diferentes umbrales de tiempo y para distintos tamaños de sesión lógica. En general, el número de clics por consulta es bastante bajo, como ya se había observado en trabajos anteriores, como [Baeza-Yates, 2005]. Aumenta con el umbral empleado, salvo en el caso de sesiones de una sola consulta, por la misma razón explicada comentando la tabla anterior. De esta manera, se reduce el número de sesiones lógicas totales, es decir, las sesiones están formadas por más consultas, y el número de clics aumenta. De hecho, como se puede apreciar, se estabiliza una vez alcanzados los 20 minutos de umbral, para todos los tamaños de sesión lógica, tal y como sucedía en las secciones anteriores con el tamaño de las sesiones lógicas, por ejemplo.

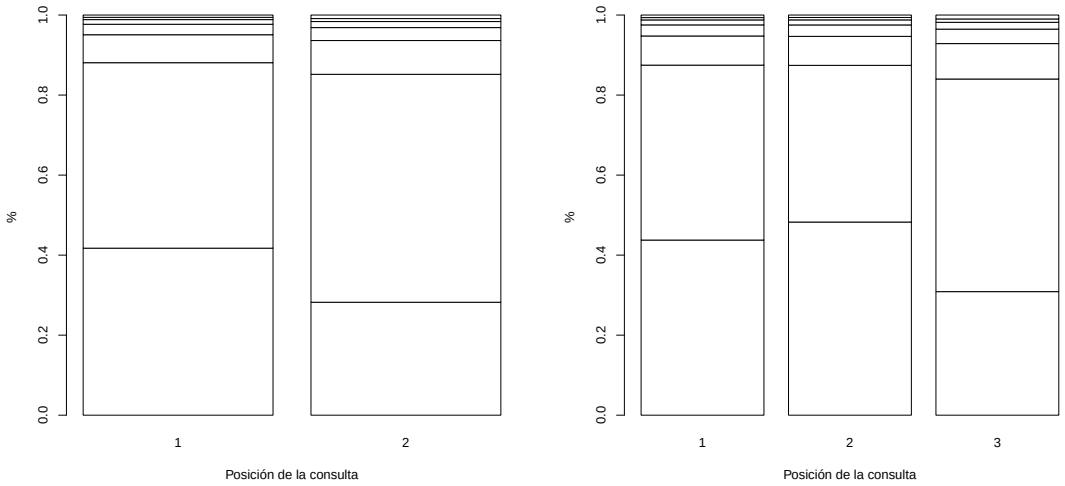
Conforme aumenta el tamaño de las sesiones el número de clics por consulta es menor. Las razones por las que esto sucede las podemos encontrar en el Capítulo 7. En primer lugar, ya habíamos comentado que elaborar una consulta acorde a la necesidad de información no resulta una tarea sencilla. Tampoco interpretar correctamente la relevancia de los resultados obtenidos. A esto se le suma que los usuarios suelen fijarse sobre todo en los primeros resultados [Joachims et al., 2007], por lo que en caso de que no sean satisfactorios una de las opciones es reformular la consulta, sin tener por qué analizar los resultados posteriores en ranking. De hecho, las técnicas de expansión de consultas parten de esta premisa, es decir, suponen que una consulta no es un más que un primer paso para obtener un objetivo final. Por otro lado, podemos también atribuir este bajo valor a la búsqueda exploratoria [Marchionini, 2006], es decir, a tareas que implican varias o incluso múltiples interacciones con el motor de búsqueda y que exigen un esfuerzo al usuario para poder comprender los resultados de las mismas. En

general, una primera búsqueda puede traer consigo nuevas búsquedas [O'Day y Jeffries, 1993] de mayor o menos complejidad. Sin embargo, la razón primordial puede ser que los usuarios, debido a la eficiencia de los motores de búsqueda, pueden comenzar el proceso de búsqueda con consultas sencillas e ir refinándolas en función de los resultados obtenidos (no siempre satisfactorios, claro), hasta obtener la información deseada [Bates, 1990; Hertzum y Frøkjær, 1996; Marchionini, 1995]. Esto último puede provocar que únicamente consultas cuyos primeros resultados parezcan relevantes sean tenidas en cuenta por el usuario, lo cual implica que un número considerable de consultas puede que no reciban clics en sus resultados.

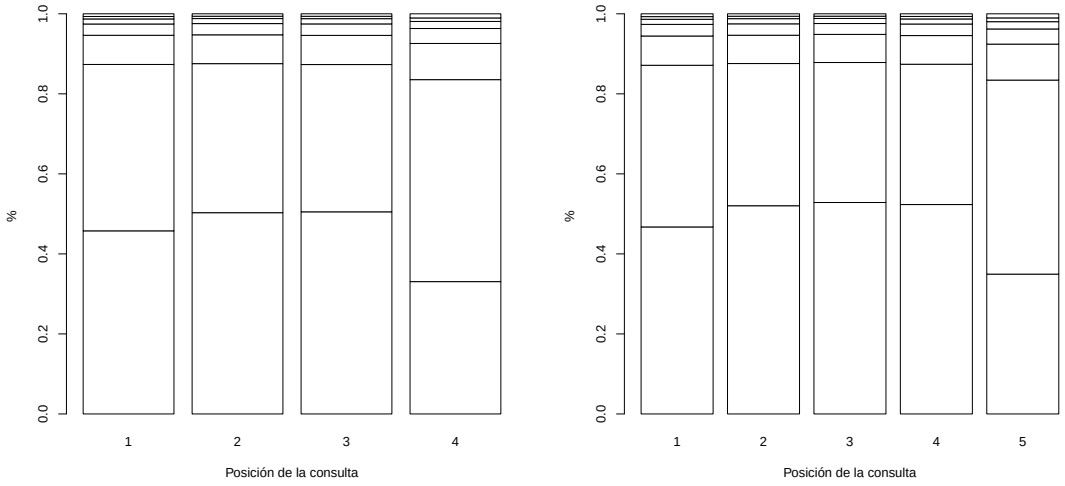
Aunque, como ya hemos comentado, el número de clics en las últimas consultas de las sesiones lógicas es reducido, en la Figura 8.2 se observa que aumenta ligeramente en la última consulta. La justificación es que estas consultas suelen estar mejor formuladas que las anteriores, puesto que el proceso de búsqueda a menudo consiste en una continua reformulación de las consultas. Cuanto mejor formulada esté, será más fácil obtener documentos relevantes y, por tanto, que el usuario haga clic en alguno de los documentos devueltos por el motor. Para todos los ejemplos mostrados también se aprecia que de recibir clic, lo más probable es que sea uno único. Además, otra observación interesante es que, después de la última consulta, la que más clics recibe es la primera, lo cual se debe a que el usuario realiza un primer esfuerzo para encontrar la información deseada en la primera consulta, y formula las siguientes hasta que observa relevancia en los resultados.

Otro aspecto interesante a tener en cuenta es saber qué posiciones ocupan las consultas que reciben algún clic y cuáles son los tamaños de las sesiones en donde eso sucede. La Figura 8.3 muestra unos resultados que, en cierto modo, son intuitivos. La mayoría de los clics en la primera consulta se corresponden con sesiones lógicas cortas (el motor de búsqueda proporciona rápido el resultado deseado). Clics en posiciones tardías se corresponden a sesiones más largas, en donde supone un mayor esfuerzo alcanzar los resultados que se buscan. Como es lógico, en el diagrama los valores de los distintos cuartiles, que muestran la longitud de las sesiones lógicas, aumentan conforme se incrementa la posición de la consulta sobre la que se ha hecho un clic.

La Figura 8.4 muestra que los clics se distribuyen entre todas las consultas de la sesión lógica. Como se puede apreciar en el diagrama, los valores inferiores de los bigotes (que son datos no atípicos) se encuentran en el valor mínimo del eje de ordenadas. Ello implica que por muy grande que sea la sesión también nos podemos encontrar clics en las primeras consultas de la misma.



(a) Longitud = 2 **(b)** Longitud = 3



(c) Longitud = 4 **(d)** Longitud = 5

Figura 8.2: Porcentaje de número de clics (0, 1, 2, 3, 4, 5 y >5) según posición de la consulta en la sesión lógica, para sesiones de longitud 2, 3, 4 y 5 (umbral de 30 minutos).

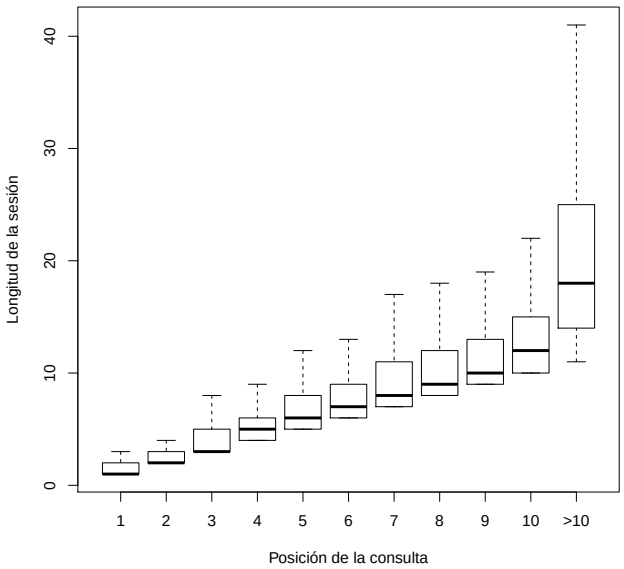


Figura 8.3: Longitud de la sesión lógica según la posición de las consultas que tienen por lo menos un clic, para un umbral de 30 minutos. Los datos atípicos no se muestran.

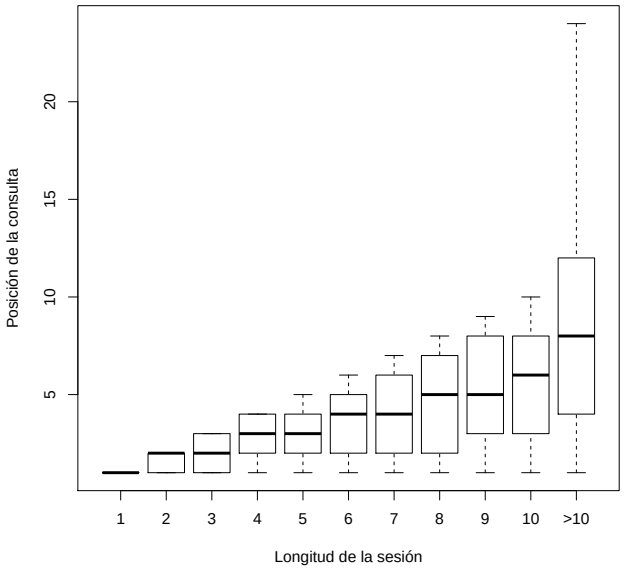


Figura 8.4: Para un umbral de 30 minutos, distribución de la posición de las consultas con clic según el tamaño de la sesión lógica.

8.3.2. Clics según el ranking de los documentos

Hasta ahora nos hemos centrado principalmente en el tamaño de las sesiones lógicas y cómo este tamaño varía según los distintos umbrales de tiempo, o según el éxito o el fracaso de las sesiones lógicas. En esta sección estudiaremos la influencia de la posición del documento a la hora de recibir o no un clic. Ya que hemos visto que el número de sesiones lógicas no varía significativamente con umbrales por encima de 20 minutos, emplearemos el umbral de 30 minutos en nuestros análisis por ser además el más empleado en la literatura.

En la Sección 7.3 se han citado trabajos que estudian cómo los usuarios habitualmente prestan más atención a los primeros resultados mostrados por el motor de búsqueda. La principal razón es que suele ser complicado determinar cuáles son los resultados relevantes para un consulta concreta, y esto provoca que se tienda a confiar en el ranking ofrecido por el propio motor. La Figura 8.5 muestra el porcentaje de clics en función del ranking del documento, teniendo en cuenta sólo aquéllas que han recibido algún clic. El documento que ocupa la primera posición en los resultados proporcionados por el motor de búsqueda es con diferencia el que recibe mayor número de clics. Se puede apreciar cómo a partir de la segunda posición los clics se distribuyen proporcionalmente entre todas las posiciones. Cuanto más arriba en la lista de resultados, más posibilidades tiene el documento de recibir un clic. Las posibilidades disminuyen significativamente por debajo de la décima posición, posición que previsiblemente es la que define el cambio de página de resultados en el motor de búsqueda. Cabe destacar, sin embargo, que un número importante de consultas no tienen ningún clic, como se había comentado anteriormente.

Lo que estudiaremos a continuación es si esto sucede con sesiones lógicas de cualquier tamaño o no. En la Figura 8.6 se observa una variación en el patrón de clics según el tamaño de la sesión lógica. Los clics en el documento en primera posición se producen sobre todo en sesiones cortas, mientras que clics en posiciones tardías (sobre todo a partir de la décima) son más frecuentes en sesiones largas. Esto último puede ser debido a las búsquedas exploratorias [Marchionini, 2006], que suelen pertenecer a sesiones lógicas de gran tamaño y que normalmente requieren de un mayor esfuerzo por parte del usuario. Otra justificación es que la persistencia en la búsqueda [Patterson et al., 2001] o el tiempo dedicado al estudio de los resultados puede significar la diferencia entre una sesión exitosa y una sesión fallida [Tabatabai y Shore, 2005]. Entonces, a mayor número de consultas sobre un mismo objetivo y a mayor cantidad de resultados analizados, más probabilidades hay de encontrar información relevante acerca del tema en cuestión.

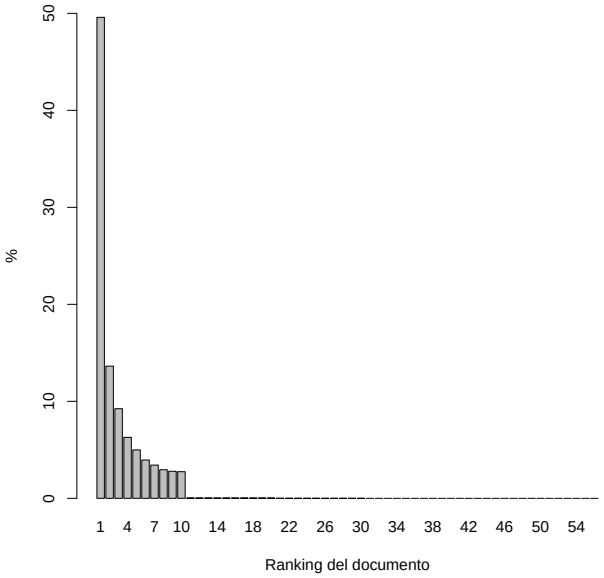


Figura 8.5: Porcentaje de clics en función del ranking del documento teniendo en cuenta sólo las consultas con al menos un clic.

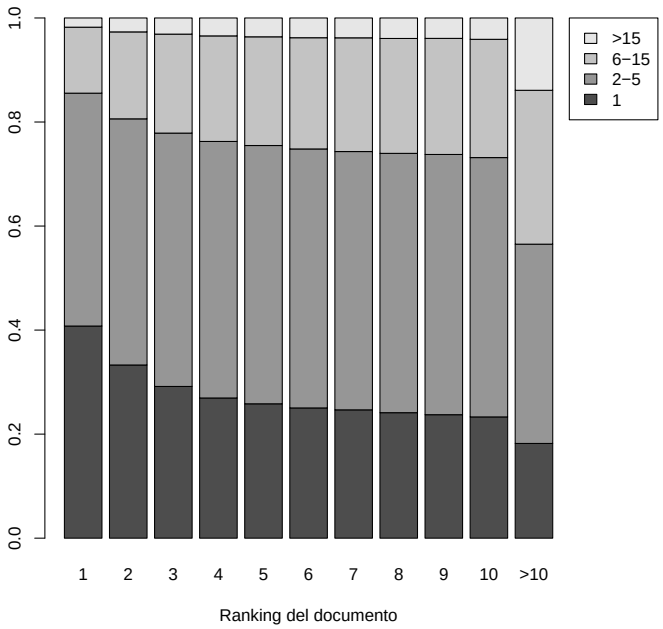


Figura 8.6: Distribución de los clics por tamaño de la sesión lógica, según la posición del documento.

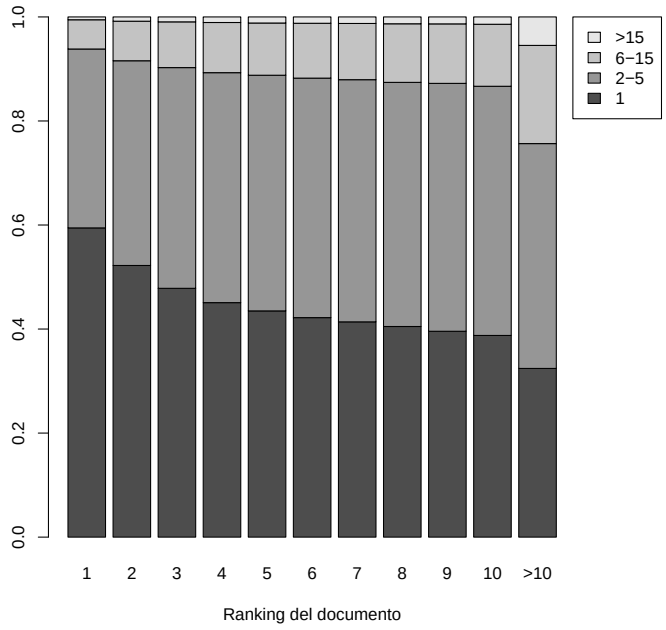


Figura 8.7: Distribución de los clics por posición de la consulta en la sesión lógica, según la posición del documento.

También es interesante analizar en qué documentos es más probable hacer clic en función de la posición de la consultas en la sesión. Como se observa en la Figura 8.7, los clics en el primer resultado son más frecuentes en la primera consulta de la sesión. Según avanzamos en la sesión, se hacen más clics en documentos que están más abajo en la lista de resultados. El motivo por el que esto sucede es que las búsquedas informativas suelen implicar sesiones lógicas cortas y se suelen contestar rápidamente en los primeros resultados proporcionados por el motor de búsqueda sin que el usuario tenga que hacer clic en un gran número de ellos. Sin embargo, búsquedas exploratorias implican generalmente un mayor número de consultas (sesiones lógicas más largas) y esto provoca que el usuario se detenga más en el resto de resultados (no sólo en los primeros) con el fin de consumir el menor tiempo posible para encontrar información relevante.

En la Figura 8.8 se aprecia cómo los clics en el primer documento se corresponden principalmente con sesiones lógicas pequeñas. Son aquéllas en las que el buscador encuentra la respuesta a la consulta fácilmente. Los clics en posiciones tardías se dan en sesiones lógicas más largas, en las que es más complicado encontrar lo que buscamos.

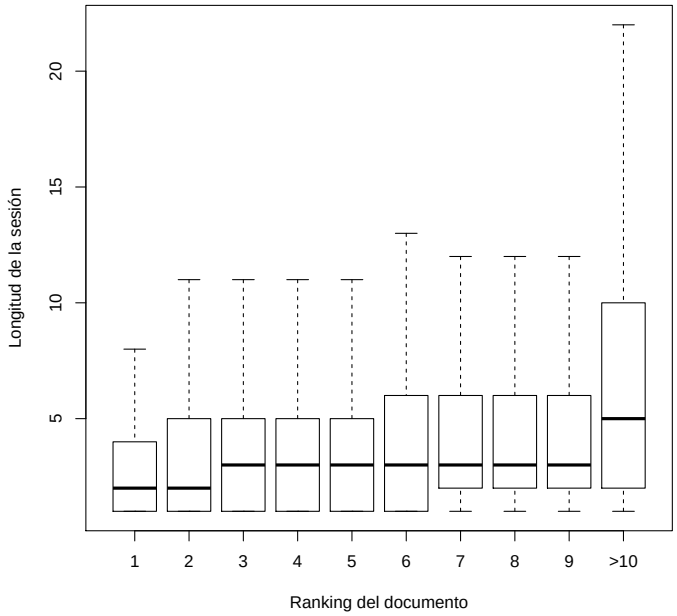


Figura 8.8: Longitud de la sesión lógica según el ranking de los documentos con clic. Los datos atípicos no se muestran.

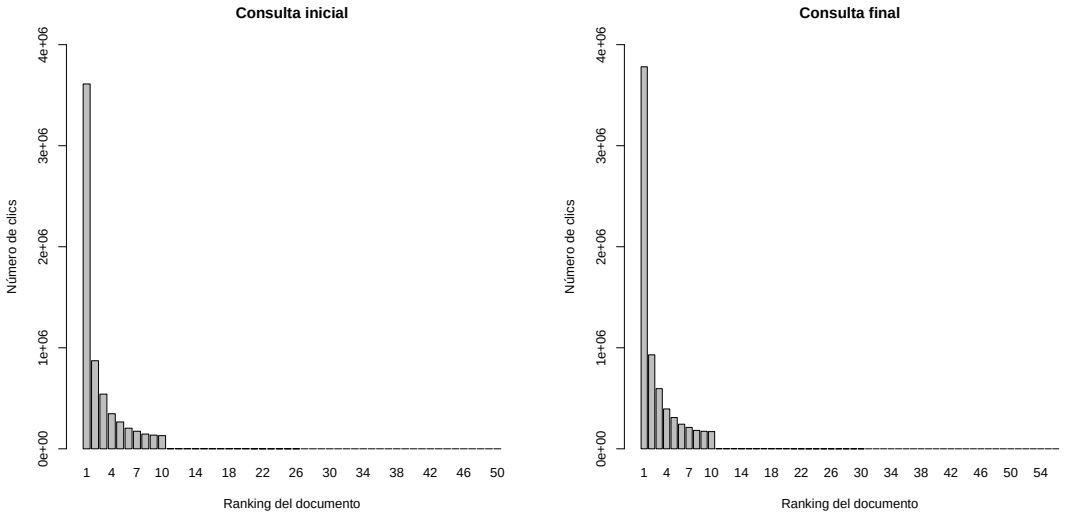


Figura 8.9: Número de clics según el ranking de los documentos, para las consultas iniciales y finales de cada sesión lógica.

De hecho, como se puede comprobar, el límite superior del bigote para los documentos situados más allá de la décima posición alcanza aproximadamente el valor 20 del eje de ordenadas, lo cual implica que en sesiones lógicas largas los clics se distribuyen entre más posiciones (el rango intercuartílico es mayor y también el valor del segundo y tercer cuartiles).

Atendiendo al papel de la consulta en la sesión lógica, podemos deducir a partir de la Figura 8.9 que el papel de la consulta (inicial o final) no influye significativamente en la distribución de los clics según el ranking del documento.

8.3.3. Clics múltiples

Para finalizar el estudio de los clics, atenderemos a los porcentajes condicionados a que un clic previo se haya o no hecho, es decir, analizaremos cuál es el comportamiento observado en el *query log* cuando se hace clic en varios documentos de una lista de resultados (ver Figura 8.10).

A partir de los resultados observados podemos concluir lo siguiente:

- En general, cuando un usuario hace clic en el primer documento no prueba en los siguientes (menos de un 10 % hace clic en el segundo, por ejemplo). Sin embargo, cuando no se hace clic en el primero, más de un 30 % de las veces se prueba en el segundo. Esto indica que en un buen número de consultas la respuesta ya se encuentra en el primer resultado.

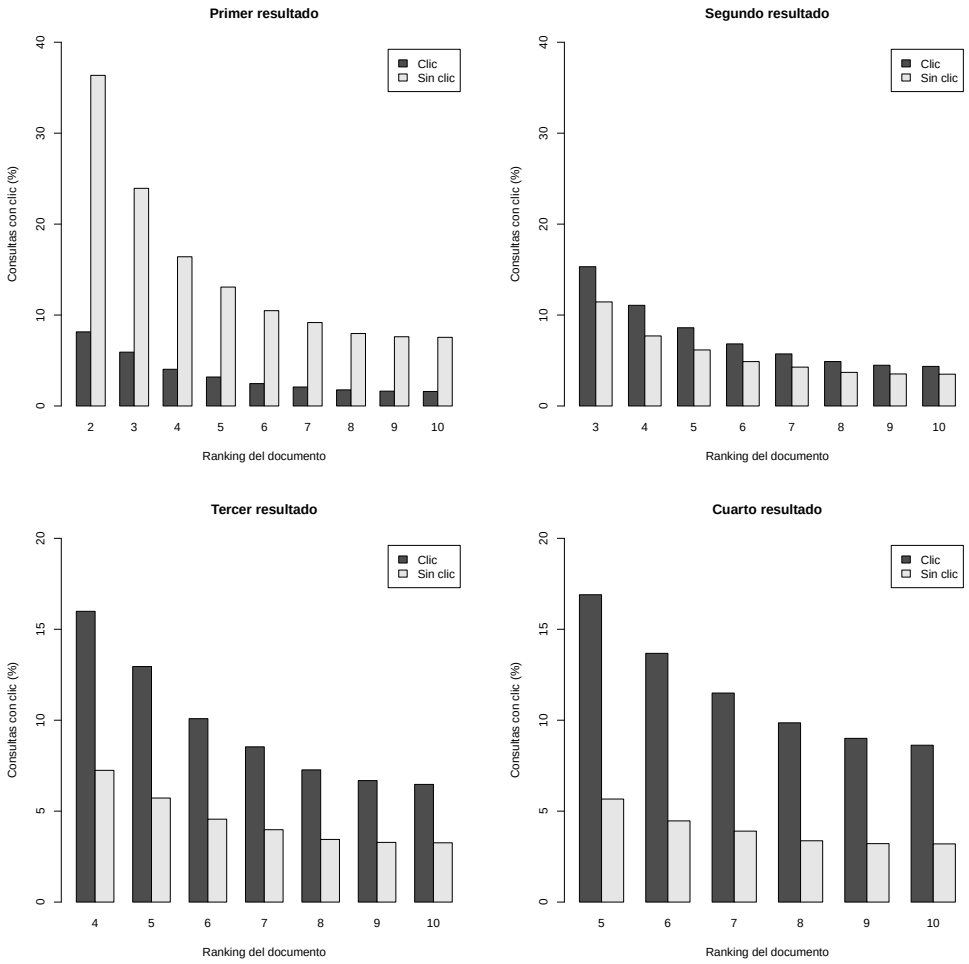


Figura 8.10: Porcentaje de clics (respecto al total de consultas con clic) en los siguientes resultados teniendo en cuenta que el usuario hizo clic en el primer, segundo, tercer o cuarto resultados, según corresponda.

- Cuando el usuario hace clic en los siguientes resultados, la tendencia cambia, sobre todo en el caso de que se haya hecho clic en el cuarto resultado. Por ejemplo, es más probable que se haga clic en el quinto resultado si previamente se ha hecho clic en el cuarto que si no se ha hecho clic. Parece deberse a que clics a partir del segundo resultado se corresponden con consultas en las que el usuario bien no está totalmente satisfecho por el resultado, bien está ante una búsqueda exploratoria compleja y, por tanto, consulta varios documentos.

- Lógicamente, en todos los casos mostrados, dado un clic en un determinado documento, los porcentajes condicionados tanto de hacer clic como de no hacerlo se

ven decrementados conforme el ranking del documento es más bajo.

8.4. Aplicación de técnicas de preprocesado

En la primera parte de esta tesis se ha mostrado el problema de la dispersión de los datos como uno de los problemas más complicados a los que se tienen que enfrentar los algoritmos de filtrado colaborativo. En caso de la búsqueda web este problema a priori se acentuará todavía más, puesto que la información será menos densa todavía que la encontrada en un dominio tradicional (como ya hemos visto, existen muchas consultas que aparecen una única vez, por ejemplo). Además, los *query logs* también contienen muchos más datos que los dominios tradicionales y su crecimiento es más rápido. Es por ello que se han considerado diferentes alternativas para reducir dicha dispersión. Aplicando dichas alternativas se intentará identificar la menor cantidad posible de consultas únicas en el *query log*.

- Una primera aproximación consiste en tratar la cadena de búsqueda (la consulta) como un todo, considerando de este modo que dos consultas son la misma sólo si son idénticas. El problema de esta aproximación es que consultas con exactamente los mismos términos, pero en diferente orden, no serán consideradas iguales, disminuyendo así la densidad de información en el *dataset*.

- La reordenación de los términos que componen una consulta es, una aproximación alternativa que permite considerar iguales consultas que tienen los mismos términos pero en distinto orden.

- *Term Stemming*, técnica usada habitualmente en Recuperación de Información que consiste en reducir distintas palabras a su raíz gramatical común, puede ser usada también en la práctica para aumentar la densidad de información en este dominio.

- *Stopwords Removal*, otra técnica usada con frecuencia en Recuperación de Información que conlleva la eliminación de artículos, nexos de unión (conjunciones, preposiciones...), es otra técnica que probaremos.

- Las técnicas anteriores pueden ser combinadas entre sí con el objetivo de incrementar la densidad de los datos.

El *query log* estudiado consta exactamente de $14,921,285$ consultas. La Tabla 8.6 muestra el número de consultas totales obtenidas tras aplicar cada uno de los métodos

Tratamiento simple de las consultas	Reordenación de términos	Eliminación de *stopwords*	Aplicación de *term stemming*	Eliminación de *stopwords* y aplicación de *term stemming*
6,623,945	6,359,767	6,089,396	6,126,219	5,812,065

Tabla 8.6: Número de consultas según tratamiento aplicado.

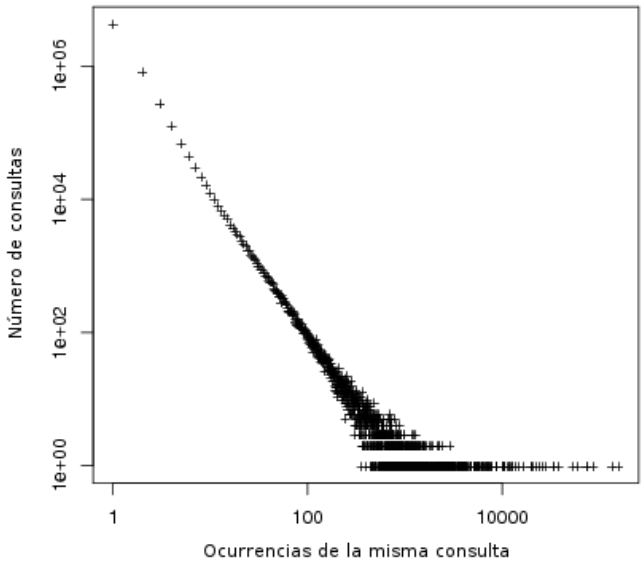


Figura 8.11: Número de consultas que se repiten el mismo número de veces aplicando *stopwords* y *term stemming*.

sugeridos. Lógicamente, con el tratamiento simple de consultas, es decir, identificando como únicas aquellas consultas que sean literalmente idénticas, es como se obtiene el mayor número de consultas. Por otro lado, identificando como únicas aquellas consultas que tienen los mismos términos, en cualquier orden, tras realizar un proceso de *stemming* y de eliminación de *stopwords*, se obtiene el número más pequeño. Por tanto, a priori, la técnica que más nos va a interesar va a ser esta última, puesto que es la que más reduce el número de consultas únicas.

Una de las finalidades de la aplicación de estos métodos es reducir las consultas que aparecen pocas veces en el *query log* y, de esta forma, aumentar la densidad de los datos. Sin embargo, este cambio no tiene lugar con ninguna de las técnicas de preprocesamiento (en la Figura 8.11 se muestra como ejemplo el caso de aplicar *stopwords* y *term stemming*). Hay un gran número de consultas únicas, pero un número mucho más bajo de consultas que se repiten un número elevado de veces. Este comportamiento

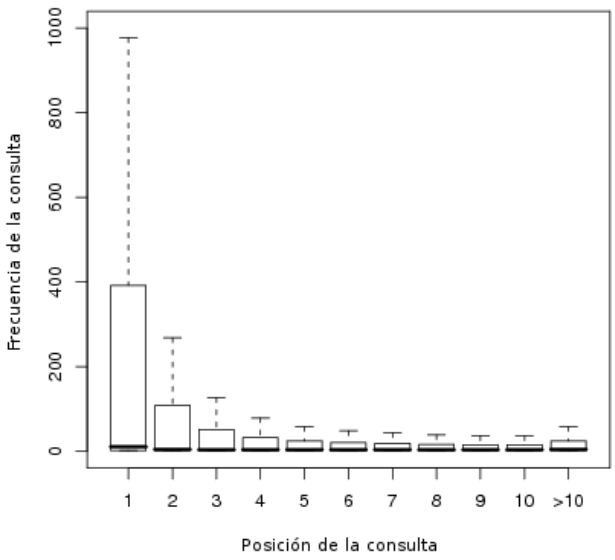


Figura 8.12: Frecuencia de una consulta según su posición en la sesión lógica aplicando *stopwords* y *term stemming*. Los datos atípicos no se muestran.

es consistente con las distintas técnicas de preprocesamiento. Para todos los casos, las consultas siguen una distribución Zipf [Zipf, 1949], como también apuntaron los autores en [Baeza-Yates et al., 2007a], donde estudiaron una muestra grande de consultas recogidas de Noviembre del 2005 a Noviembre del 2006 de Yahoo en el Reino Unido[1]. Esta distribución se debe a que los usuarios combinan consultas populares con consultas únicas y no a que realmente haya dos grupos de personas: aquéllas que usan consultas populares y aquéllas que las usan únicas [Goel et al., 2010].

Una vez mostrado que la distribución no ha cambiado, es interesante averiguar si existe una dependencia importante entre la frecuencia (es decir, el número de veces que repite una consulta) y su posición en la sesión lógica. Esto se confirma en la Figura 8.12, lo cual es lógico, ya que según avanzamos en la sesión lógica nos encontraremos con consultas más específicas. En concreto, la primera consulta de las sesiones es, con diferencia, la más frecuente. Los valores de la mediana, sea cual sea la posición de la consulta, son muy bajos debido a que, a pesar de todo, seguirá habiendo gran cantidad de consultas que únicamente aparezcan una vez, como ya habíamos comentado previamente. Estas observaciones son consistentes para todos los métodos de preprocesado estudiados.

[1] http://www.yahoo.co.uk

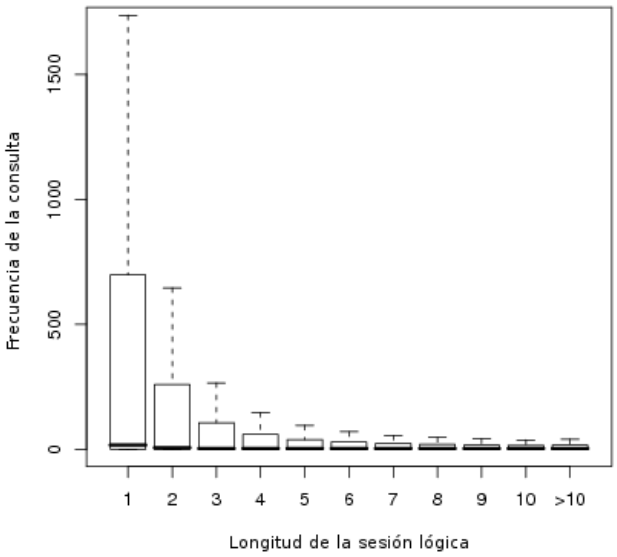


Figura 8.13: Frecuencia de una consulta según la longitud de la sesión lógica aplicando *stopwords* y *term stemming*. Los datos atípicos no se muestran.

Sucede lo mismo según la longitud de la sesión lógica. En la Figura 8.13 se muestra el caso con el método de *stopwords* y *term stemming*, pero estos resultados son extensibles también para el resto de las técnicas de preprocesamiento estudiadas. Como ya se ha comentado en el Capítulo 7, esto es debido en gran parte a que los motores de búsqueda son muy eficientes actualmente, lo cual permite comenzar la búsqueda por consultas genéricas para después ir concretándolas conforme se van obteniendo resultados hasta llegar a obtener la información deseada [Bates, 1990; Hertzum y Frøkjær, 1996; Marchionini, 1995]. Lógicamente, estas consultas genéricas aparecerán más frecuentemente, por término general, en el *query log* que las consultas más refinadas y precisas. Es decir, sesiones cortas suelen tener consultas más genéricas, que aparecen con más frecuencia, mientras que sesiones largas suelen tener consultas más específicas, que, por tanto, aparecen con menos frecuencia.

Aparte de lo que acabamos de comentar, es importante destacar que las consultas iniciales y finales son más frecuentes que las consultas intermedias, tal y como se muestra en la Figura 8.14. El comportamiento es consistente para distintos tamaños de sesiones, y, de nuevo, para los distintos métodos de preprocesamiento. Para justificar este hecho debemos primero pensar en cómo una persona realiza una búsqueda. Está claro que aquéllas que son simples (por ejemplo, el tiempo que va a hacer hoy en nuestra ciudad, la fecha de un determinado evento, etc.) encajan correctamente en consultas iniciales

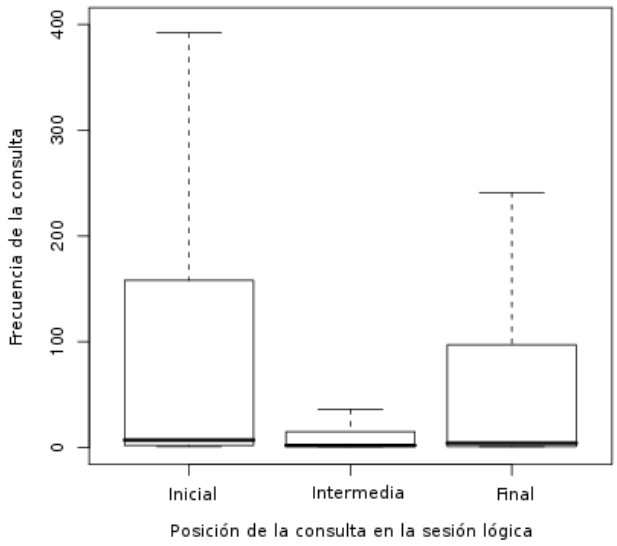


Figura 8.14: Frecuencia de una consulta según su papel en la sesión lógica, para sesiones lógicas de por lo menos dos consultas aplicando *stopwords* y *term stemming*. Los datos atípicos no se muestran.

que pueden llegar a ser relativamente frecuentes, y que, de desearlo, el usuario es capaz de especificar para alcanzar un resultado todavía más concreto. Lógicamente, conforme sea más específica, esta consulta será menos frecuente, como ya se ha comprobado en la Figura 8.12. Por tanto, tiene sentido que las consultas intermedias y la final sean menos frecuentes que la primera. También hemos mostrado que las últimas consultas reciben, por término medio, más clics que las demás (ver Figura 8.2). Esto implica que las consultas situadas en las últimas posiciones de las sesiones lógicas estarán mejor formuladas ya que tienden a ser más relevantes. Esto, unido a que diversos usuarios pueden encontrar la misma consulta relevante y a que los usuarios intentan alcanzar sesiones satisfactorias, hace que sea más probable que estas últimas consultas aparezcan también con cierta frecuencia, lo cual explica la diferencia entre la frecuencia de las consultas intermedias y de las consultas finales.

8.5. Sesiones lógicas satisfactorias

En esta sección estudiaremos el éxito de las sesiones lógicas según su consulta inicial y su consulta final. Para ello emplearemos una segmentación por umbral de tiempo de 30 minutos e identificación simple de consultas. Lo que buscamos es saber qué porcentaje de las sesiones lógicas que comienzan por una consulta dada son exitosas, y

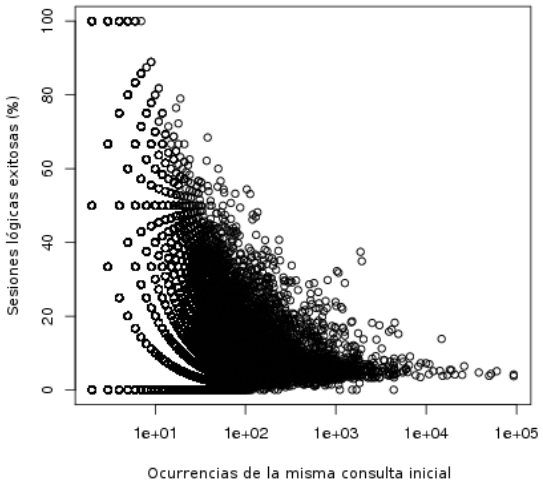


Figura 8.15: Para cada consulta inicial, porcentaje de éxito frente a número de sesiones lógicas que empiezan por esa consulta.

análogamente con la consulta final.

La Figura 8.15 representa el porcentaje de sesiones lógicas exitosas respecto a todas las que comienzan por una misma consulta. En la parte izquierda de la misma aparecen las consultas menos frecuentes y en la derecha las más frecuentes. Así, aquellas consultas que sólo aparecen dos veces en el *query log* podrán únicamente tener tres posibles valores en la gráfica: 0, 50 o 100. El 0 indica que las sesiones lógicas a las que pertenece la consulta en cuestión no fueron exitosas; es decir, no se ha hecho ningún clic en los resultados de las últimas consultas de dichas sesiones. El 50 indica que sólo una de ellas fue exitosa. El 100 que las dos fueron exitosas. De ahí que la gráfica tenga esa forma tan característica. Por otro lado, se observa que las sesiones lógicas que comienzan con consultas muy frecuentes no siempre tienen éxito. Esto se explica porque el usuario ya había escogido mal esa primera consulta o bien porque el usuario no escoge correctamente las siguientes. El primero de los casos parece difícil de resolver, ya que es complicado conocer cuáles son las necesidades reales del usuario si no formula correctamente esa primera consulta. El segundo ofrece más perspectivas de mejora, ya que se trataría de guiar al usuario por el camino correcto a partir de la primera consulta.

La Figura 8.16 aporta información adicional. Como se puede comprobar gracias al valor de las medianas, al menos el 50 % de las sesiones lógicas que comienzan por consultas muy poco frecuentes no son exitosas. De hecho, para el caso de las consultas que aparecen 4 veces en el *query log*, por ejemplo, el valor 75 y el valor 100 aparecen como datos atípicos, ya que, como mínimo, el 75 % de los sesiones lógicas que comienzan

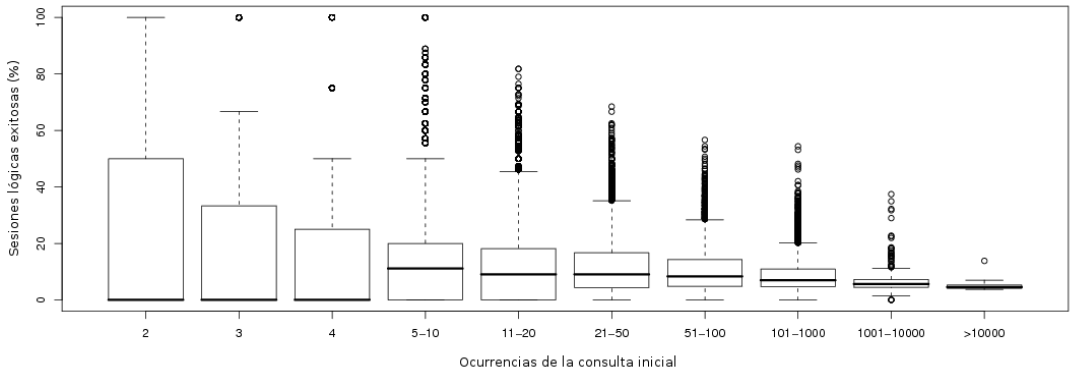


Figura 8.16: Éxito según el número de sesiones lógicas iniciadas por una consulta.

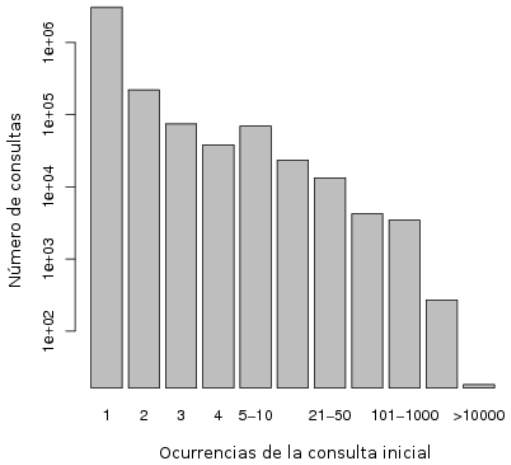


Figura 8.17: Número de consultas iniciales según las veces que se repiten.

por estas consultas tienen un 25 % de porcentaje de éxito, dato que se obtiene a partir del valor del tercer cuartil. Para ningún caso de los estudiados las medianas superan el valor 20 de porcentaje de éxito. A partir de las consultas que se repiten entre 5 y 10 veces, la gran mayoría de las consultas iniciales presentan una tasa de éxito en sus sesiones lógicas por debajo del 20 %, aunque están presentes numerosos datos atípicos.

Los mismos intervalos que aparecen en el eje de abscisas en la Figura 8.16 aparecen también en la Figura 8.17 con el objetivo de conocer el número de consultas iniciales existentes en cada uno de ellos. Como se puede ver en el histograma, a mayor número de veces que aparece una consulta, menor número de consultas existen con ese número de repeticiones en el *query log*.

El comportamiento con respecto a las consultas finales es muy semejante al que se ha mostrado para las consultas iniciales, frente a lo que podríamos suponer a priori, por lo que ya no se muestran las gráficas. Esto implica que existen numerosos procesos de búsqueda diferentes, y que una consulta, ya esté al principio o al final de una sesión lógica, no es indicadora siempre de un resultado satisfactorio, sobre todo cuando se trata de consultas muy frecuentes en el *query log*.

8.6. Conclusiones

En este capítulo hemos analizado el *query log* público de MSN con el fin de comprender cómo se comportan los usuarios a la hora de realizar una búsqueda. Hemos probado la segmentación de dicho *query log* en base a distintos umbrales de tiempo. Esto nos ha permitido comprobar que los usuarios no suelen tardar más de 20 minutos en estudiar los resultados de una consulta antes de realizar la siguiente y que más del 50 % de las sesiones para todos los umbrales estudiados tenían una única consulta.

Hemos visto que el número de clics que recibe cada consulta es bastante bajo, en términos generales, aumentando ligeramente en la última consulta. En muchas de estas consultas el documento situado en la primera posición recibe un clic lo cual demuestra que con frecuencia los usuarios se fían del ranking propuesto por el motor de búsqueda. A menudo la respuesta se encuentra ya en el primer resultado. De no ser así, se suelen realizar clics en más de una posición. Conforme aumenta el tamaño de la sesión estos clics tienen lugar en posiciones más bajas del ranking de documentos y, además, se distribuyen entre más posiciones.

También hemos probado diferentes técnicas de preprocesado para reducir la dispersión de los datos en el *query log*. Sin embargo, ninguno de los métodos estudiados ha supuesto una reducción considerable de las consultas de poca frecuencia. Hemos observado una dependencia entre la frecuencia de las consultas tanto con su posición en la sesión lógica como también con la longitud de la sesión, siendo más frecuentes las consultas en las primeras posiciones y en sesiones cortas.

Finalmente, hemos considerado sesiones exitosas aquellas sesiones con algún clic en la última consulta. Sin embargo, el porcentaje de sesiones exitosas es bajo en general, especialmente en la sesiones de gran tamaño, debido a que en muchas ocasiones suelen requerir mayor esfuerzo por parte del usuario. Esto implica que el usuario no ha escogido correctamente la primera consulta, o que no ha sabido seguir el camino correcto a partir de una consulta bien escogida para alcanzar un final satisfactorio. Supone un desafío que usuarios que hayan seguido un *camino correcto* ayuden a otros usuarios que, habiendo

escogido bien las primeras consultas, no han sido capaces de obtener los resultados esperados.

En el futuro, sería interesante repetir este estudio con otros algoritmos de segmentación de sesiones más complejos, para poder comparar así los resultados y las conclusiones obtenidas.

Capítulo 9

Search Shortcuts: El problema de los atajos de búsqueda

Hasta el momento hemos visto cómo se comportan los algoritmos de filtrado colaborativo en un dominio tan estudiado como el de la recomendación de películas. Hemos también analizado qué es la búsqueda web, qué actores intervienen en ella, como son los procesos de búsqueda, etc., así como los datos contenidos en un *query log* de acceso público. En este capítulo abordamos el modelo de *Search Shortcuts*, consistente en recomendar consultas que fueron útiles a usuarios que habían buscado información semejante anteriormente, un desafío que surgió a partir del análisis presentado en el capítulo anterior. Dicho modelo formaliza el problema de *Query Suggestion*, y permite evaluar los algoritmos empleados a partir de los *query logs* de los buscadores. Por tanto, evaluaremos la aplicación de técnicas de filtrado colaborativo sobre datos procedentes de buscadores reales y estudiaremos diversas alternativas y soluciones a problemas presentes en este nuevo dominio, como la extracción de información de los *query logs*, o la necesidad de nuevas métricas para evaluar la calidad de las recomendaciones.

9.1. Introducción

La cantidad de información disponible a día de hoy es enorme, así como su complejidad. Los grandes motores de búsqueda poseen *logs* de billones de consultas realizadas por millones de visitantes que desean satisfacer sus necesidades de información. En la era de las comunicaciones, los usuarios disponen cada vez de menos tiempo, por lo que esperan encontrar características nuevas y personalizadas para poder acceder a dicha información. La búsqueda se convierte por tanto en una tarea fundamental, que tiene gran impacto en el día a día de los usuarios, por lo que los motores de búsqueda no

sólo han de responder satisfactoriamente las peticiones que reciben, sino que también han de hacerlo rápidamente. En este sentido, cualquier ayuda proporcionada al usuario para reducir el tiempo invertido tiene una gran importancia. De hecho, los principales motores de búsqueda, además de poder responder a las peticiones en unos pocos cientos de milisegundos, normalmente ofrecen al usuario algunas sugerencias, en forma de consultas, que están de algún modo relacionadas con la información que el usuario necesita, como ya hemos comentado en el Capítulo 7.

En este capítulo explicamos el problema de los *Search Shortcuts* como un problema relacionado con la recomendación de consultas en motores de búsqueda y con las posibles reducciones en la longitud de las sesiones de los usuarios. Tal y como indicamos en [Baraglia et al., 2009; Cacheda et al., 2009], la idea es recomendar consultas que ya hayan permitido a algún usuario (en el pasado) encontrar con éxito la información deseada usando un proceso de búsqueda similar.

Para aclarar pongamos un ejemplo. Supongamos que un alto número de usuarios han realizado las consultas q_1, q_2, q_3, y, finalmente, después de enviar la consulta q_4, encontraron la información que necesitaban. Por tanto, podremos considerar que la consulta q_4 es relevante para los usuarios interesados en temas relacionados con q_1, q_2 y q_3. Así, en cuanto otro usuario empiece a buscar temas relacionados con q_1, q_2 o q_3, la consulta q_4 será propuesta como un *shortcut*.

Obviamente, un *shortcut* en la sesión del usuario será más efectivo cuanto antes se sugiera. Además, un *shortcut* para una sesión no tiene por qué anticipar la última consulta, es decir, una consulta propuesta será aceptable siempre que reduzca el tamaño de dicha sesión. Cuanto mayor sea esta reducción, más importante será el *shortcut*. Partiendo de esta idea, definimos una metodología de evaluación para el problema definido basada en *query logs*, que permitirá una comparación sencilla y directa de las técnicas que pueden ser aplicadas al problema.

En particular, hemos estudiado la aplicación de algoritmos de filtrado colaborativo explicados en la primera parte de esta tesis. Su aplicación al problema citado, y, en general, a cualquier sistema basado en datos provenientes de *query logs*, trae consigo una serie de retos que serán comentados a lo largo de este capítulo.

En nuestra aproximación, primero inferimos la relevancia de una consulta basándonos en si termina exitosamente una sesión lógica de búsqueda (es decir, si la consulta es útil para encontrar la información que el usuario está buscando). Esta medida de relevancia, extraída de los datos del *query log*, se puede usar para rellenar algunas celdas de la matriz de puntuaciones que típicamente se usa en los algoritmos de filtrado colaborativo. Por tanto, este tipo de algoritmos se pueden aplicar a la recomendación de consultas.

A continuación describiremos el problema de Search Shortcuts. Después indicaremos qué algoritmos hemos empleado y las diferencias y similitudes entre este dominio y el de la recomendación de películas comentado anteriormente. En la Sección 9.4.1 explicaremos los experimentos realizados y comentaremos los resultados. Finalmente, presentaremos las conclusiones alcanzadas.

9.2. Search Shortcuts

9.2.1. Descripción del problema y motivación

El objetivo de Search Shortcuts es recomendar consultas que en el pasado permitieron a usuarios encontrar exitosamente la información que estaban buscando usando un proceso de búsqueda similar. El problema de Search Shortcuts no se considera realmente distinto a Query Suggestion. De hecho, es una formalización de uno de sus posibles objetivos. La principal ventaja de Search Shortcuts es que permite una evaluación medible de la eficacia de los algoritmos propuestos. Sea Q una sesión lógica de un usuario compuesta por las consultas q_1, q_2, ..., q_n, n$\geq$1. La consulta final de Q, q_n, determina el resultado de la sesión: exitosa, si el usuario hizo un clic en al menos uno de los resultados propuestos por el motor de búsqueda; fallida en otro caso. El conjunto de consultas finales que condujeron a una sesión exitosa del usuario son las candidatas a ser incluidas como atajos de búsqueda que, sugeridas idealmente muy al principio de la sesión, pueden acortar la longitud de la misma.

El análisis de clics es un modo de evaluar la relevancia de los resultados obtenidos para una determinada consulta [Joachims et al., 2007]. Estos datos son fácilmente recuperables de un motor de búsqueda. Sin embargo, al ser datos implícitos transparentes para el usuario, presentan ruido.

En el Capítulo 8 mostramos cómo consultas que en determinadas ocasiones conducen a sesiones exitosas, pueden también dar lugar a sesiones fallidas. Para abordar este hecho, la formulación de nuestro problema define una aproximación colaborativa a la sugerencia de consultas: la idea principal es explotar la experiencia de usuarios previos para conducir en la dirección correcta a otros usuarios con necesidades de información similares.

Hemos validado esta suposición analizando el comportamiento real de usuarios de motores de búsqueda en el *query log* de AOL [Pass et al., 2006], de modo análogo al estudio realizado sobre el *query log* de MSN en la Sección 8.5. Este *query log* está compuesto de aproximadamente 36 millones de registros de consultas, que contienen sobre 20 millones de consultas web realizadas por 650,000 usuarios a lo largo de tres

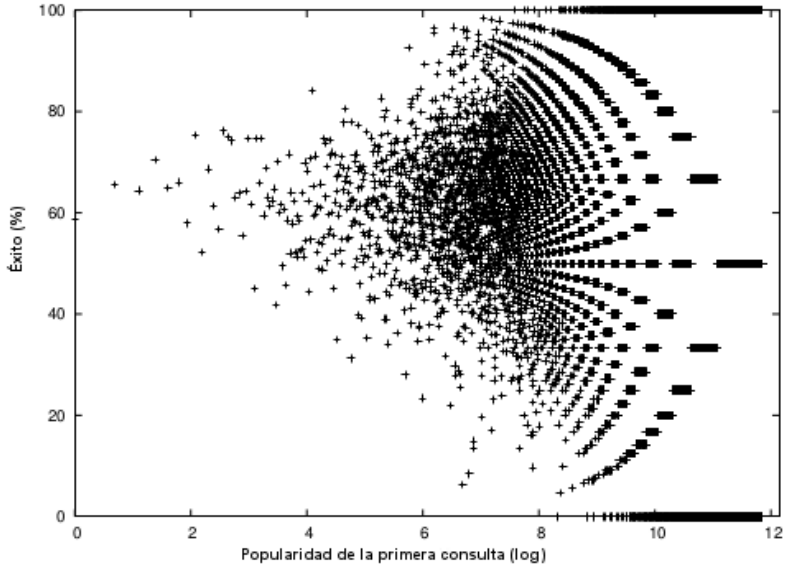


Figura 9.1: Porcentaje de sesiones lógicas satisfactorias en el *query log* de AOL, según la popularidad de la primera consulta.

meses (entre marzo y mayo de 2006). Extrajimos las sesiones de búsqueda del *query log* teniendo en cuenta que una sesión expira si el usuario pasa al menos 30 minutos sin realizar ninguna acción en el motor de búsqueda. Este es límite que se usa comúnmente en análisis de *query logs* [White et al., 2008] y en el que más profundizamos en el Capítulo 8. Como resultado, obtuvimos 10,765,687 sesiones de búsqueda con una longitud media de 2,03 consultas.

Ya que nuestro problema carece de relevancia cuando las sesiones son pequeñas, de las sesiones obtenidas consideramos sólo aquéllas que involucrasen a 2 o más consultas (4,339,683 sesiones con una longitud media de 4.88 consultas/sesión) y agrupamos juntas aquellas sesiones que comenzasen por la misma consulta. Queremos analizar los caminos de búsqueda seguidos por diferentes usuarios que empezaron por la misma consulta, asumiendo que podrían tener la misma necesidad de información. Algunos de los usuarios pueden seguir un camino correcto y terminar después visitando algunos documentos propuestos por el motor de búsqueda, pero otros pueden terminar la sesión sin visitar ningún resultado. Por tanto, para cada una de estas sesiones, examinamos la consulta final y comprobamos si la consulta fue exitosa o no. Las sesiones que terminaron con una consulta final exitosa se consideran caminos de búsqueda satisfactorios.

En resumen, encontramos 140,165 consultas iniciales diferentes (no únicas), que son el punto inicial para varias sesiones (al menos dos). Considerando estos caminos de búsqueda, el 64 % de las sesiones fueron satisfactorias, mientras que el 36 % termina-

ron con una consulta fallida. En la Figura 9.1 mostramos el porcentaje de caminos de búsqueda satisfactorios, ordenados por el logaritmo del ranking de consultas iniciales. En la parte izquierda de la figura, que representa los caminos de búsqueda asociados con las consultas iniciales más frecuentes, podemos ver que la mayoría de las sesiones terminaron de forma exitosa, pero, al mismo tiempo, hay algunas sesiones que terminaron sin ningún clic sobre sus documentos, aunque empezaron en el mismo punto. Esto muestra que la información proporcionada por las sesiones satisfactorias podría conducir a las sesiones fallidas a un final exitoso, lo cual es nuestro principal objetivo.

Para completar, la parte derecha de la imagen contiene las consultas menos frecuentes y el gráfico representa los puntos en las fracciones comunes, lo cual produce su forma particular. Por ejemplo, para todas las consultas iniciales repetidas dos veces, tenemos un punto en 2/2, 1/2 y 0/2; para las consultas iniciales repetidas tres veces, tenemos un punto en 3/3, 2/3, 1/3 y 0/3; y así sucesivamente. En este caso, los beneficios potenciales son menos claros, porque la experiencia que puede ser extraída de caminos de búsqueda similares es más reducida.

9.2.2. Modelo teórico de Search Shortcuts

A continuación se expondrá el modelo teórico definido para poder aplicar las técnicas de filtrado colaborativo en la recomendación de consultas, así como la métrica propuesta para evaluar cómo se ajustan las recomendaciones a las consultas reales (extraídas de un *query log*).

Sea $\mathcal{U} = \{u | u \text{ es un usuario}\}$ el conjunto de todos los usuarios de un motor de búsqueda. Sea $\mathcal{Q} = \{q | q \text{ es una consulta de un usuario}\}$ el conjunto de todas las consultas que han sido enviadas por los usuarios de un motor de búsqueda.

Definición 9.1 *Una sesión para un usuario $u \in \mathcal{U}$ es una secuencia de consultas que el usuario u ha enviado a un servicio de búsqueda con el objetivo de satisfacer una necesidad de información. Formalmente, $\sigma^u =< q_1^u \dots q_n^u >$. Se asume que todas las consultas están relacionadas con la misma tarea de búsqueda.*

Se eliminará el superíndice u en la especificación de σ, e.g. $\sigma =< q_1 \dots q_n >$, cuando esté claro u en el contexto a tratar.

El i-ésimo elemento de una sesión se denota como σ_i.

$\mathcal{S}$ representa el conjunto de todas las sesiones.

Definición 9.2 *Definimos una función $c : \mathcal{S} \times [1..n] \to \{0, 1\}$ como $c(\sigma, i) = 1$ si en σ el usuario ha hecho clic en al menos uno de los documentos mostrados como resultado para σ_i. En cualquier otro caso, $c(\sigma, i) = 0$.*

Definición 9.3 *Una sesión σ será **satisfactoria** si, y sólo si, $c(\sigma, n) = 1$; **insatisfactoria** en otro caso.*

Definición 9.4 *Definimos un **shortcut de grado** k como una función $h : \mathcal{S} \rightarrow 2^{\mathcal{Q}}$ que toma como argumento una sesión y devuelve un conjunto de consultas de cardinalidad menor que k, i.e. $|h(\sigma)| \leq k$.*

$\mathcal{H}$ es el conjunto de todas las posibles funciones shortcut.

Definición 9.5 *La **cabeza** $\sigma_{t|}$ de σ **hasta** $t \leq n$ es la secuencia de las primeras t consultas en σ, i.e. $\sigma_{t|} =< q_1, \ldots, q_t >$*

Definición 9.6 *La **cola** $\sigma_{|t}$ de σ **desde** $t \leq n$ es la secuencia de las últimas $n - t$ consultas en σ, i.e. $\sigma_{|t} =< q_{t+1}, \ldots, q_n >$*

Definición 9.7 *Sea σ una sesión satisfactoria. La similitud de un shortcut de grado k, h, sobre una cabeza $\sigma_{t|}$ y una cola $\sigma_{|t}$ se define como:*

$$s\left(h\left(\sigma_{t|}\right), \sigma_{|t}\right) = \frac{\sum\limits_{q \in h(\sigma_{t|})} \sum\limits_{m=1}^{n-t} \left[q = \left(\sigma_{|t}\right)_m\right] f(m)}{|h(\sigma_{t|})|} \tag{9.1}$$

Donde $f(m)$ es una función monótona creciente. La función $[q = \sigma_m] = 1$ si, y sólo si, la consulta q es igual a la consulta σ_m.

La función de similitud definida en la Ecuación 9.1 se puede usar como la métrica de evaluación objetiva para el problema de Search Shortcuts. Por ejemplo, para evaluar la eficacia de una función *shortcut* h en $\mathcal{S}$, se puede calcular la suma o media de s en todas las sesiones de $\mathcal{S}$. Tiene en cuenta tanto el número de consultas recomendadas que forman parte de la sesión real, como su posición en ésta, siendo mejor valorado un algoritmo que recomienda las últimas consultas en la sesión.

Además, la función de similitud se puede reescribir para incluir la función c y dar importancia sólo a aquellas consultas que tuvieron un resultado con clic. La fórmula se muestra en la Ecuación 9.2.

$$s\left(h\left(\sigma_{t|}\right), \sigma_{|t}\right) = \frac{\sum\limits_{q \in h(\sigma_{t|})} \sum\limits_{m=1}^{n-t} \left[q = \left(\sigma_{|t}\right)_m\right] c\left(\sigma_{|t}, m\right) f(m)}{|h(\sigma_{t|})|} \tag{9.2}$$

Es importante comentar que la principal diferencia entre Search Shortcuts y *Query Suggestion* viene representada por la función $\left[q = \left(\sigma_{|t}\right)_m\right]$ de las Ecuaciones 9.1 y 9.2.

Relajando el requisito de = estricto y reemplazándolo por una relación de similitud, i.e. $\left[q \sim \left(\sigma_{|t}\right)_m\right]$ (es decir, $\left[q \sim \left(\sigma_{|t}\right)_m\right] = 1$ si, y sólo si, la consulta q es parecida a la consulta σ_m) el problema se reduce, básicamente, a *Query Suggestion*. Definiendo una función de similitud apropiada, la Ecuación 9.1 también se puede usar para evaluar la eficacia de *Query Suggestion*.

Por otro lado, la función $f(m)$ es muy importante en la definición del cálculo de las similitudes. De hecho, dependiendo de qué f sea escogida, se pueden probar diferentes características del algoritmo que genera los atajos de búsqueda. Por ejemplo, estableciendo $f(m)$ a una función constante $f(m) = c$, la métrica simplemente mide el número de consultas en común entre el conjunto de atajos de búsqueda y las consultas enviadas por el usuario. Para dar mayor valor a las consultas que un usuario envió al final de su sesión se podrían usar funciones no constantes. Así, la función $f(m) = e^m$ sería la función escogida si se quisiese asignar mucha más puntuación a las consultas sugeridas pronto en la sesión. Se pueden usar otras funciones f más suaves para modular los efectos de la posición de las consultas dentro de la sesión.

9.3. Algoritmos de Search Shortcuts

En esta sección probaremos algunos algoritmos de filtrado colaborativo tradicionales adaptados a este problema. El objetivo de estos algoritmos es predecir la utilidad de un producto dado (tarea de predicción), o bien recomendar una lista de productos (tarea de recomendación). Estos fines parecen ajustarse al problema de Search Shortcuts que hemos introducido. Dado que las técnicas de filtrado colaborativo han sido exitosas en varios dominios, como el comercio eléctronico, parece razonable aplicar estos métodos a nuestro problema.

Recordemos que los algoritmos de filtrado colaborativo calculan las recomendaciones basándose en las puntuaciones que los usuarios dieron a los productos. El sistema puede bien preguntar a los usuarios por tales puntuaciones (puntuaciones explícitas), o bien inferirlas a partir del comportamiento de los usuarios (puntuaciones implícitas). El problema de Search Shortcuts se basará normalmente en puntuaciones implícitas, extraídas de los registros del *query log*. El conjunto de productos valorados por cada usuario, junto con la puntuación dada, conforman el perfil del usuario. Los perfiles de todos los usuarios se almacenan en la matriz denominada matriz de puntuaciones. En dicha matriz, cada fila representa a un usuario y cada columna a un producto. Cuando un usuario valora un producto, se rellena la celda correspondiente de la matriz con la puntuación otorgada por el usuario. Las celdas de productos todavía no valorados

se dejan en blanco. Como es de suponer, la mayoría de las celdas de la matriz no contendrán ningún valor, ya que cada usuario puntuará sólo unos pocos productos, por lo que la matriz es normalmente dispersa, como hemos visto en la primera parte de esta tesis.

Como se muestra, el filtrado colaborativo y Search Shortcuts son problemas similares. Sin embargo, existen también diferencias importantes entre ellos.

Antes de nada, en el problema de Search Shortcuts no existe la matriz de puntuaciones. En su lugar, tenemos un *query log* con los movimientos del usuario en el motor de búsqueda. Crear la matriz de puntuaciones a partir de la información en el *query log* no es una tarea sencilla. Aquí surgen principalmente dos retos. En primer lugar, los conceptos de Search Shortcuts (usuarios, consultas, términos y sesiones) han de tener su correspondencia en el problema de filtrado colaborativo (usuarios y productos). Como la meta en el problema de Seach Shortcuts es recomendar consultas para una sesión dada, parece razonable tratar cada sesión de consultas como un usuario, y cada consulta como un producto. En segundo lugar, las puntuaciones para las consultas deben ser inferidas a partir de la información existente en el *query log*. En la aproximación considerada, puntuamos las consultas en función de la última consulta de la sesión. Las puntuaciones se asignarán del siguiente modo:

- Si la última consulta fue existosa (el usuario hizo clic sobre al menos un resultado), entonces a la consulta se le da una puntuación positiva
- Si la última consulta fue fallida, entonces la consulta recibe una puntuación negativa.
- Todas las restantes consultas se consideran neutrales.

Existen también diferencias significativas entre las características de los datos de un *query log* y los datasets utilizados en los recomendadores tradicionales basados en filtrado colaborativo. Estos datasets son mucho más densos que un *query log*, es decir, tienen muchas más relaciones (puntuaciones) entre los usuarios y los productos. Por ejemplo, en un sitio de comercio electrónico la mayor parte de los productos han recibido alguna puntuación o han sido comprados por varios clientes. Sin embargo, en los *query logs* hay muchas consultas que aparecerán en una única sesión, y, por tanto, no podremos confiar en la utilidad de tales consultas. Esta falta de información trae consigo el consabido problema de la dispersión de los datos que afecta a la mayoría de los algoritmos de filtrado colaborativo. Para poder aplicar un algoritmo al problema de Search Shortcuts ha de tener buen rendimiento en entornos dispersos.

Como ya se hemos visto, los algoritmos a emplear mejoran su comportamiento a medida que el conjunto de datos es menos disperso. Por ello se han considerado las mismas alternativas que hemos visto en la Sección 8.4 para reducir dicha dispersión. Es decir, el tratamiento simple, la reordenación de términos, *Term Stemming* y *Stopwords Removal*. Aplicando dichas alternativas se intentará identificar la menor cantidad posible de consultas únicas en el *query log*.

Finalmente, los *query logs* también contienen muchos más datos que los dominios tradicionales. Mientras que en un sitio de comercio electrónico los datos pueden ser más o menos estáticos, en un *query log* se añaden continuamente nuevas consultas y sesiones. Para aplicar filtrado colaborativo al problema de Search Shortcuts, necesitamos algoritmos computacionalmente eficientes tanto en predicción como en tiempo de entrenamiento, ya que el modelo se actualizará muy frecuentemente.

Teniendo esto en cuenta, se muestran a continuación los algoritmos más relevantes de filtrado colaborativo que han sido probados en el problema de Search Shortcuts:

- Item-Mean. Es un algoritmo extremadamente sencillo, que siempre devuelve como predicción la media del elemento en cuestión.
- Algoritmos basados en usuario [Herlocker et al., 2002; Shardanand y Maes, 1995]. Una de las estrategias de filtrado colaborativo más populares. Predicen la utilidad de un elemento mediante un proceso que incluye tres fases:

  1. Calcular la similitud entre el usuario activo y el resto de usuarios. Existen distintas técnicas para tal fin, entre las que hemos evaluado el coeficiente de correlación de Pearson [Resnick et al., 1994], Pearson forzado [Shardanand y Maes, 1995], Mean Squared Difference [Shardanand y Maes, 1995] y Pearson ponderado [Herlocker et al., 1999].
  2. Seleccionar un subconjunto de los usuarios según su similitud con el usuario activo (es decir, sus vecinos). Hemos evaluado la técnica más popular, el máximo número de vecinos [Resnick et al., 1994], ya que es aquella que ofrece mejores resultados [Herlocker et al., 2002].
  3. Finalmente, calcular la predicción a partir de las valoraciones de los vecinos. Hemos estudiado dos alternativas: normalización centrada en la media [Resnick et al., 1994] y normalización z-score [Herlocker et al., 2002].

- Algoritmos basados en elementos [Sarwar et al., 2001]. Son similares a los basados en usuarios, pero en lugar de buscar los usuarios más parecidos, intentan encontrar elementos similares a aquél para el que queremos realizar la predicción.

- Slope-One [Lemire y Maclachlan, 2005]. Un sencillo modelo para filtrado colaborativo, basado en las diferencias entre valoraciones, que consiste en predictores de la forma $f(x) = x + b$. Hemos evaluado las variantes *simple* y *ponderado*.

- Personality Diagnosis [Pennock et al., 2000]. Está basado en un modelo probabilístico sencillo que aproxima la forma en que los usuarios valoran elementos mediante una distribución normal.

- Algoritmo basado en tendencias [Cacheda et al., 2011a]. Una aproximación basada en las variaciones naturales entre diferentes usuarios y elementos.

9.4. Experimentos

9.4.1. Configuración de los experimentos

Para realizar los experimentos hemos usado datos de *query logs* reales, de los motores de búsqueda AOL [Pass et al., 2006] y MSN (analizado en el Capítulo 8). Hemos dividido los datos en dos subconjuntos, como en la evaluación de los Sistemas Recomendadores tradicionales: subconjunto de entrenamiento y subconjunto de evaluación. En este caso, las consultas son los productos a ser sugeridos, y la evaluación de esta recomendación se extrae implícitamente de las sesiones exitosas pertenecientes al conjunto de entrenamiento.

Para construir los subconjuntos hemos seguido un proceso en dos pasos. Primero, hemos escogido de modo aleatorio un porcentaje de las sesiones para ser usadas como datos para entrenar a los algoritmos. Si pensamos en un sistema real, esta parte del conjunto de entrenamiento sería equivalente a sesiones pasadas ya registradas en el *query log*. Después, alimentamos los algoritmos con las primeras dos consultas de cada una de las sesiones del conjunto de evaluación. Éstas representan la *cabeza de la sesión* que el algoritmo necesita como contexto de predicción. Comparando de nuevo con un sistema real, esto se corresponde a las primeras consultas de la sesión de un usuario que está intentando conseguir un atajo de búsqueda.

Los resultados han sido evaluados usando métricas tradicionales (ampliamente empleadas), y nuevas métricas, una de las cuales ha sido diseñada especialmente para el problema de Search Shortcuts. Tales métricas se pueden clasificar en métricas de precisión en la predicción y métricas de precisión en la clasificación.

Las primeras tienen como objetivo evaluar la calidad del algoritmo al predecir puntuaciones desconocidas para un usuario. Comparan las puntuaciones predichas por el sistema con las reales, hechas por los usuarios, que están en el subconjunto de eva-

luación. Entre este tipo de métricas, hemos usado Mean Absolute Error (MAE), ya que es la más popular usada en la literatura. También hemos usado otras métricas como Good Predicted Items MAE (GPIM) y Good Items MAE (GIM) [Cacheda et al., 2011a]. Éstas se centran en medir los errores de predicción en los productos relevantes, que son aquéllos que realmente importan al usuario, y, por tanto, proporcionan unos resultados más cercanos a su punto de vista.

Por otro lado, las métricas de precisión en la clasificación evalúan la calidad de la lista de recomendación, es decir, la relevancia de los productos que están siendo recomendados. Entre ellas, hemos usado precisión, que mide la proporción de productos relevantes entre los productos recomendados, *recall*, que mide la proporción de productos relevantes recomendados entre todos los productos relevantes del *dataset* y *ROC Curves* [Herlocker et al., 2004]. Éstas evalúan un algoritmo basándose en si los elementos recomendados son realmente buenos. Otra métrica tradicional que también ha sido usada es Half-Life Utility [Breese et al., 1998], interesante porque tiene en cuenta el orden de los productos en la lista. También hemos introducido una métrica especialmente diseñada para la evaluación de Search Shortcuts, basada en la métrica propuesta en la Sección 9.2.2. Para ésta hemos usado diversas variaciones de la función monótona creciente $f(m)$ (ver Ecuación 9.1), para evaluar diferentes aspectos de cada algoritmo:

- Una función constante, $f(m) = 1$
- Una función lineal, $f(m) = m$
- Una función cuadrática, $f(m) = m^2$
- Una función exponencial, $f(m) = e^m$

Cada tipo de métrica comentada requiere una metodología de evaluación diferente. Cuando usamos métricas de precisión en la predicción, cada algoritmo se entrena con el conjunto de entrenamiento, y después debe predecir la utilidad de cada uno de los productos del conjunto de evaluación. Los resultados se comparan con las puntuaciones reales usando las métricas comentadas.

Con respecto a la evaluación de la precisión en la clasificación, el algoritmo genera una lista de productos recomendados. Entonces, esa lista se compara con los productos relevantes encontrados en el conjunto de evaluación. Una pregunta que nos debemos hacer en este momento es cómo seleccionar el subconjunto a partir del cual el algoritmo tomará los productos recomendados. En un sistema real, el algoritmo debería recomendar cualquier producto no valorado por el usuario todavía, es decir, se deberían

recomendar consultas que el usuario no ha enviado aún. Sin embargo, dicha aproximación no es factible en una evaluación basada en un *dataset* offline con un gran volumen de datos como el que estamos usando. La razón es que para evaluar un algoritmo necesitamos comparar sus resultados con los productos presentes en el subconjunto de evaluación. Si permitimos al algoritmo recomendar cualquier producto, haya sido o no valorado por el usuario, podremos obtener una lista de recomendación sin productos que pertenezcan al conjunto de evaluación, y, por tanto, no seremos capaces de realizar ninguna evaluación. Para evitar esto, hemos limitado la recomendación a los productos que forman parte del conjunto de evaluación. Aunque de este modo se puede calcular la precisión de la clasificación, es necesario decir que los resultados pueden diferir de los obtenidos a partir de una evaluación por medio de usuarios reales, donde todos los productos pueden formar parte de la recomendación.

Hemos tenido en cuenta también el *coverage* de cada algoritmo, es decir, el porcentaje de productos para el que el algoritmo es capaz de realizar una predicción. Algoritmos con bajo *coverage* tenderán a recomendar siempre los mismos productos y, por tanto, son menos interesantes desde el punto de vista de Search Shortcuts.

Con respecto a los datos empleados, hemos derivado diez nuevos *datasets* a partir de los dos ya citados, aplicando las técnicas de preprocesado analizadas en el capítulo anterior; es decir:

- Considerar la consulta como una cadena única.
- Tener en cuenta diferentes términos en una consulta.
- Aplicar *stopword removal*.
- Aplicar *term stemming*.
- Aplicar *stopword removal* y *term stemming*.

Además, también hemos estudiado el efecto de considerar sólo sesiones con más de un número determinado de consultas. Ambas técnicas, como vemos en la Figura 9.2, permiten aumentar la densidad de los datos, lo cual se prevé que mejore la precisión de los resultados. En especial, ignorar sesiones con pocas consultas tiene un impacto significativo en la densidad del *dataset*. Cuando se consideran únicamente sesiones de al menos tres consultas, la densidad aumenta significativamente, lo cual es fácilmente explicable por el hecho de que un gran número de sesiones desaparecen del *dataset* final. En realidad, el número total de sesiones se reduce hasta un 80 % en el *dataset* de MSN, y hasta un 70 % en el caso del AOL.

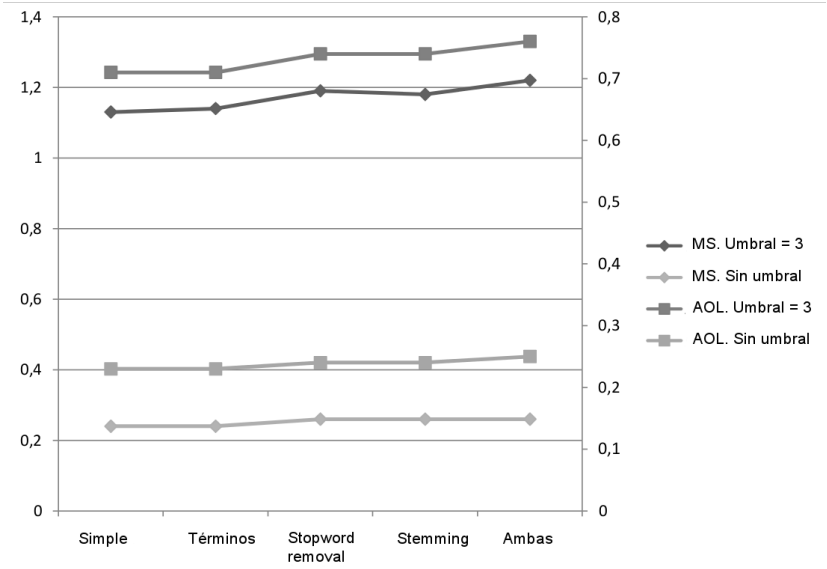


Figura 9.2: Efecto de las diferentes técnicas de preprocesado sobre los *query logs* de AOL -eje izquierdo- y MSN -eje derecho-. Los resultados se muestran en términos de densidad relativa x10^5.

Del mismo modo, aplicar cualquier técnica de preprocesamiento también ayuda a aumentar la densidad de datos, aunque esto tenga menos impacto que descartar sesiones con pocas consultas (ya habíamos visto que existe gran cantidad de sesiones con una sola consulta en el *query log* de MSN). En este caso la densidad se incrementa debido a que estas técnicas reducen el número de consultas únicas. Los mejores resultados se obtienen aplicando *stopword removal* y *term stemming*, especialmente si las sesiones pequeñas también se descartan.

9.4.2. Resultados

En primer lugar, hemos analizado la precisión en la predicción de los diferentes algoritmos, para así evaluar qué tal se comportan las técnicas de filtrado colaborativo en contextos caracterizados por la baja densidad y gran volumen de datos, como el que ahora nos ocupa. Como ya hemos comentado, los dominios en que se han aplicado tradicionalmente estas técnicas son relativamente densos. En la Figura 9.3 podemos observar los resultados, según la métrica MAE. Como vemos, la mayoría de algoritmos presentan unos resultados bastante buenos, especialmente en el *dataset* de AOL, con un error absoluto medio entre el 5 % y el 10 %. Sin embargo, estos buenos resultados deben ser interpretados cuidadosamente.

De hecho, si observamos el *coverage* de cada algoritmo (también mostrado en la

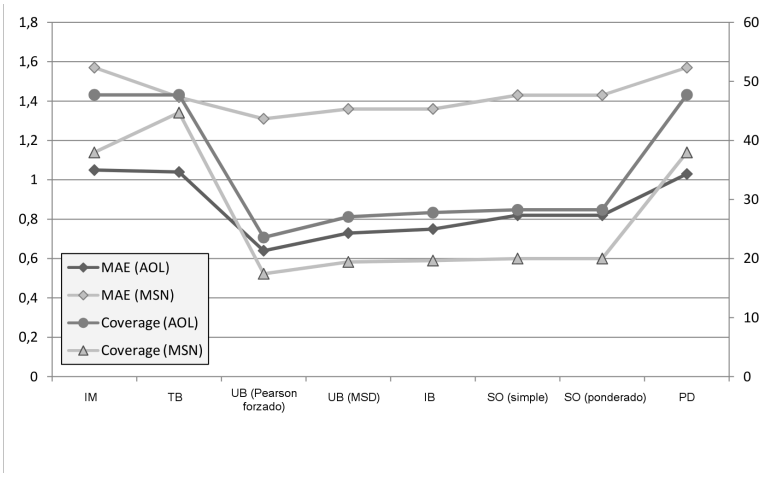


Figura 9.3: Resultados de MAE y *coverage* en los *datasets* de AOL y MSN.
Resultados de MAE -eje de la izquierda- y *coverage* -derecha- en los *datasets* de AOL y MSN.

Figura 9.3), podemos ver que los buenos resultados en MAE (es decir, predicciones bastante exactas) están a menudo relacionados con un bajo *coverage*. La razón es que algunos algoritmos extraen muy poca información de los datos disponibles y, por tanto, sólo pueden predecir la utilidad de aquellos elementos para los cuales existe gran cantidad de información en el *dataset*. Tal caso suele corresponder a elementos populares, generalmente fáciles de predecir, y, por tanto, la precisión es mejor en ellos. Sin embargo, el usuario normalmente ya los conoce, con lo que la recomendación no es muy útil para él [Ziegler et al., 2005]. Esto es lo que esperábamos, ya que los algoritmos de filtrado colaborativo no fueron diseñados para contextos tan dispersos como éste. En dominios como el comercio electrónico, tenemos más información disponible, y, por tanto, los algoritmos pueden basarse sólo en algunas partes de ella y aún así obtener resultados precisos. Este es el caso, por ejemplo, de los algoritmos basados en usuarios y basados en productos, que sólo tienen en consideración las relaciones entre usuarios o productos. Cuando se enfrentan a un contexto tan disperso como los datos de un *query log*, estos algoritmos no son capaces de extraer información suficiente para valorar la mayoría de los productos, y, por tanto, su *coverage* es muy bajo.

Es de destacar que el sencillo algoritmo Item-Mean presenta un coverage de alrededor del 14 %. Dado que este algoritmo simplemente devuelve la puntuación media del producto como predicción, esto significa que más del 86 % de los productos no han sido puntuados todavía. Es decir, el 86 % de las consultas del conjunto de evaluación no aparecen en el conjunto de entrenamiento, y, por tanto, no pueden ser recomendadas

basándose en las sesiones previas. En este *dataset* el mejor *coverage* gira entorno al 14 %, lo cual nos puede dar una idea de la dispersión de los datos y los problemas que tendrán los algoritmos de filtrado colaborativo.

Por otra parte, algunas aproximaciones basadas en modelo, como el algoritmo basado en tendencias o Personality Diagnosis presentan, globalmente, buenos resultados.

Además, la métrica MAE tiene otra limitación importante en este contexto, relacionada con la forma en que asignamos valoraciones de forma implícita a partir del *dataset*. Teniendo en cuenta que hemos asignado una valoración neutral a todas las consultas en una sesión, excepto la última, y además solamente estamos evaluando en sesiones relativamente grandes, la mayor parte de las consultas van a tener una valoración neutral. Por tanto, para un algoritmo va a resultar más sencillo obtener una predicción para dichos elementos. Por ejemplo, un algoritmo que se limite a predecir siempre una valoración neutral, obtendría resultados muy precisos en la mayoría de elementos, y éstos compensarían los malos resultados obtenidos en otros, ya que la métrica MAE tiene en cuenta la media del error cometido en todo el conjunto de evaluación.

Partiendo de esta observación, se puede comprobar que Personality Diagnosis, el algoritmo basado en tendencias e Item Mean son los mejores algoritmos de acuerdo a los resultados globales. Los buenos resultados en Item Mean son un poco sorprendentes, dado que es un algoritmo realmente simple. Además, recomienda los productos basándose en su puntuación media, así que recomendará siempre las mismas consultas para todas las sesiones. Claramente, esto no es una buena recomendación. En este caso, los buenos resultados se deben a las limitaciones de las métricas de precisión en la predicción.

La misma conclusión puede obtenerse a partir de la evaluación basada en métricas de precisión en la clasificación (los resultados para los *datasets* de AOL y MSN pueden verse, respectivamente, en las Figuras 9.4 y 9.5). Aunque ningún algoritmo ofrece unos resultados destacados, debemos tener en cuenta las limitaciones de las métricas de evaluación, especialmente cuando ésta se realiza en un *dataset* offline, con gran volumen de datos pero baja densidad. En estas condiciones, es muy probable que las recomendaciones ofrecidas por el algoritmo no se correspondan realmente con ninguno de los elementos que tenemos en el conjunto de evaluación. Por tanto, la evaluación no podrá llevarse a cabo de forma satisfactoria. Para minimizar este problema, hemos restringido los elementos que el algoritmo podría recomendar a aquéllos presentes en el conjunto de evaluación. Tal hecho debe tenerse en cuenta a la hora de interpretar los resultados obtenidos en estas métricas. Dado que el algoritmo solamente puede recomendar unos pocos elementos, un pequeño error puede llegar a tener un gran impacto en la preci-

sión final. Además, como el porcentaje de productos no relevantes es muy alto, es muy fácil cometer tales errores, lo cual también explica la baja precisión. Existe un gran volumen y dispersión de los datos. El algoritmo no tiene suficiente información para evaluar la mayoría de los productos (bajo *coverage*), por lo que muchos de ellos quedan automáticamente fuera de la lista de recomendación. Al final, el tamaño de la lista es muy pequeño si lo comparamos con todos los datos.

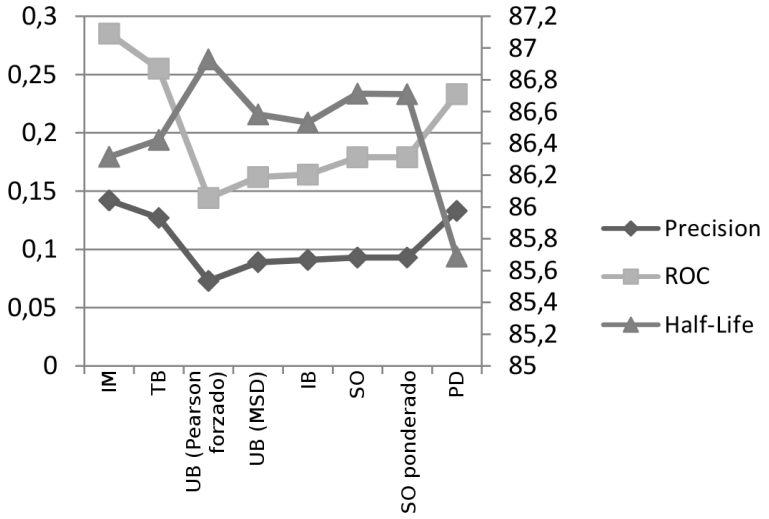


Figura 9.4: Resultados, en el *dataset* de AOL, de las métricas de precisión, ROC -eje de la izquierda- y *half-life utility* -derecha-.

Además, las métricas tradicionales se basan en el concepto de *relevancia* de un elemento, que es en cierto modo ambiguo en el contexto de *Search Shortcuts*. Por supuesto, una consulta debería considerarse relevante cuando conduce al usuario a la información que estaba buscando, pero esto no puede saberse a partir de los datos en el *query log*. Por tanto, los investigadores pueden elegir entre varias formas de interpretar la *relevancia* de una consulta: si tras ella ha finalizado la sesión, el número de resultados que han sido visitados por el usuario, el tiempo que este tarda en consultarlos, etc.

La métrica de *Search Shortcuts* explicada en la Sección 9.2.2, elimina esta ambigüedad, al definir claramente lo que debemos evaluar: el número de consultas que la recomendación ha ahorrado al usuario. Por tanto, esta métrica para minimizar especialmente diseñada para la evaluación de Search Shortcuts intenta paliar las limitaciones que tienen las métricas tradicionales a la hora de evaluar los *datasets offline*. Centrémonos ahora en esta métrica.

En primer lugar, hemos estudiado la relación entre esta nueva métrica y las métricas tradicionales de precisión en la clasificación. En concreto, es especialmente interesante

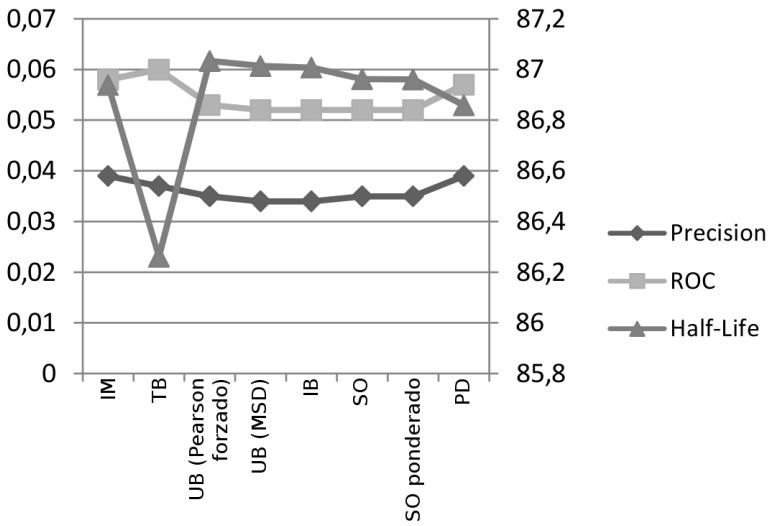


Figura 9.5: Resultados, en el *dataset* de MSN, de las métricas de precisión, ROC -eje de la izquierda- y *half-life utility* -derecha-.

estudiar la correlación entre esta métrica y la métrica tradicional de precisión, ya que las dos pretenden medir características similares. Como podemos ver en la Figura 9.6, la correlación entre ambas métricas depende de la función f elegida. En concreto, si elegimos la función constante, $f(x) = 1$, ambas métricas están fuertemente correlacionadas. Esto no es de extrañar, ya que al elegir una función constante mide el porcentaje de consultas en el subconjunto de evaluación que aparecen en la sesión original, es decir, las consultas *relevantes* desde el punto de vista de *Search Shortcuts*. Esta característica es realmente importante, ya que significa que los resultados de la métrica presentada pueden interpretarse en términos de *precisión*. Pero además, al contrario que la métrica de precisión tradicional, la métrica presentada para *Search Shortcuts* ofrece una interpretación clara de lo que significa relevancia en el contexto tratado, y, además, cómo debe ser calculada. Esto, sin duda, tiene un gran valor a la hora de evaluar y comparar algoritmos entre sí.

Por otra parte, si elegimos una f distinta, podemos evaluar otras características de los algoritmos, que las métricas tradicionales no son capaces de cubrir. Por ejemplo, usando la función exponencial, $f(x) = e^x$, estaremos evaluando no sólo si las consultas recomendadas son *relevantes*, sino hasta qué punto lo son; es decir, lo cerca que están del final de la sesión, y, por tanto, el tiempo que potencialmente podrían ahorrar al usuario.

En las Tablas 9.1 y 9.2 puede consultarse una evaluación de los distintos algoritmos empleando esta métrica. Los resultados en ambos *datasets* muestran claras diferencias

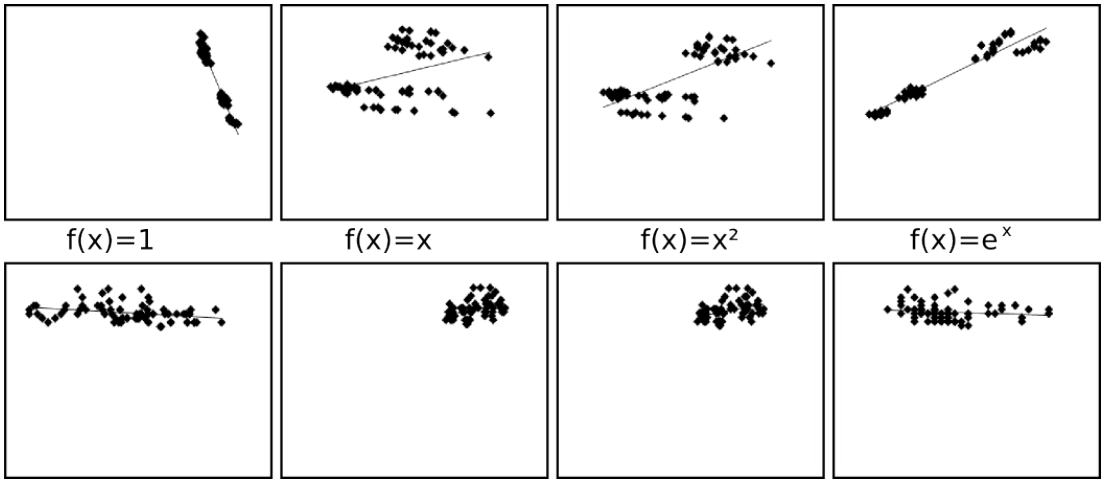


Figura 9.6: Correlación entre la función de similitud de *Search Shortcuts* y la precisión, dependiendo de la función f elegida, tanto en el *dataset* de AOL -arriba- como el de MSN -abajo-.

entre los algoritmos basados en memoria y los basados en modelo. Cuando elegimos una f constante, por tanto considerando de igual forma cualquier *shortcut* relevante, independientemente de su posición en la sesión original, los algoritmos basados en usuario obtienen los mejores resultados. Sin embargo, si reemplazamos la f para evaluar hasta qué punto las recomendaciones son útiles para reducir la sesión, podemos comprobar que tanto el algoritmo basado en tendencias como Personality Diagnosis presentan resultados más precisos. Además, esto se puede observar en ambos *datasets*, y con la mayoría de métricas. Estos dos algoritmos muestran también buenos resultados en términos de exactitud en la predicción, precisión e incluso *coverage*, por lo que parecen buenos candidatos para el problema de *Search Shortcuts*. Sin embargo, siguen siendo sensibles a la baja densidad de información, aunque en menor medida que los algoritmos basados en memoria.

De hecho, el impacto negativo que tiene la baja densidad del *dataset* puede observarse en los resultados de los algoritmos con todas las métricas estudiadas. Este comportamiento es el esperado, ya que los algoritmos de filtrado colaborativo estudiados han sido diseñados para contextos con mucha más información. Por tanto, es de esperar que la aplicación de técnicas para reducir la densidad del *dataset*, como las presentadas en la sección 3.1, tenga un impacto importante en los resultados. Hemos estudiados dos técnicas:

- Ignorar sesiones pequeñas, es decir, sesiones con menos de 3 consultas.

Algoritmos	$f(x) = 1$	$f(x) = x$	$f(x) = x^2$	$f(x) = e^x$
Item Mean	1.956	0.6	0.354	0.267
Basado en tendencias	1.989	**0.612**	**0.361**	0.265
UB (Pearson ponderado)	**2.284**	0.607	0.338	0.246
UB (MSD)	2.181	0.589	0.33	0.252
IB	2.173	0.586	0.329	0.253
Slope One	2.182	0.589	0.331	0.252
Slope One ponderado	2.181	0.589	0.331	0.252
Personality Diagnosis	1.951	0.598	0.351	**0.272**

Tabla 9.1: Resultados de varios algoritmos según la métrica de similitud de *Search Shortcuts*, en el *dataset* de AOL.

Algoritmos	$f(x) = 1$	$f(x) = x$	$f(x) = x^2$	$f(x) = e^x$
Item Mean	1.491	0.301	0.165	0.258
Basado en tendencias	1.416	**0.33**	**0.188**	**0.273**
UB (Pearson ponderado)	**1.537**	0.268	0.142	0.261
UB (MSD)	1.505	0.274	0.146	0.265
IB	1.505	0.272	0.145	0.264
Slope One	1.513	0.27	0.144	0.263
Slope One ponderado	1.513	0.27	0.144	0.262
Personality Diagnosis	1.457	**0.33**	**0.186**	0.263

Tabla 9.2: Resultados de varios algoritmos según la métrica de *Search Shortcuts*, en el *dataset* de MSN.

- Aplicación de técnicas de preprocesado: *term reordering*, *term stemming*, y *stopword removal*.

En la Figura 9.7 podemos ver que, efectivamente, ignorar sesiones con menos de 3 consultas tiene un gran impacto en la precisión de los algoritmos. En ciertos algoritmos basados en modelo, la mejora obtenida es superior al 30 %, y llega hasta cerca del 50 % cuando se combina con técnicas de preprocesado. Este resultado es el esperado, ya que los algoritmos de filtrado colaborativo no se comportan bien con usuarios que han valorado pocos elementos, lo que es equivalente a sesiones pequeñas en el contexto de *Search Shortcuts*. Esto corresponde al problema de *cold-start* [Schein et al., 2002], que, como podemos ver, puede solucionarse fácilmente en este contexto. Por otra parte, la mejora no es tan espectacular en los algoritmos basados en memoria (en el centro de ambas gráficas), lo que puede explicarse por el hecho que que estos algoritmos

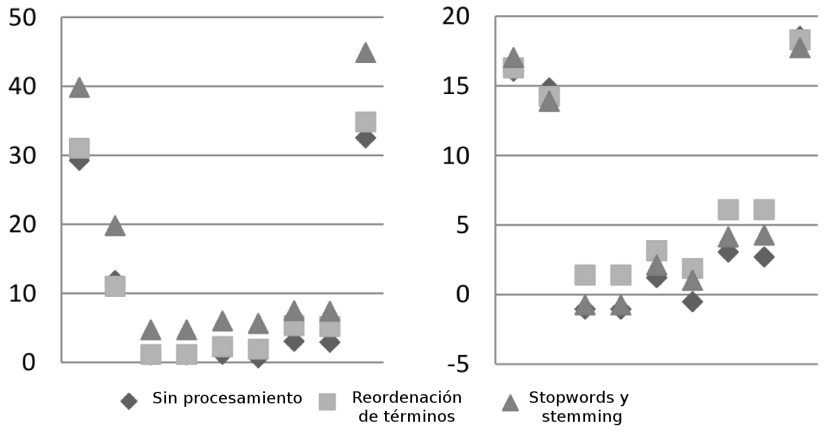


Figura 9.7: Porcentaje de mejora, en precisión del algoritmo, cuando las sesiones con menos de 3 consultas son ignoradas, en los *datasets* de AOL -derecha- y MSN -izquierda-. El orden de los algoritmos es el mismo que en las Figuras 9.3, 9.4 y 9.5.

tienen importantes limitaciones para extraer información de *datasets* poco densos. La contribución de este tipo de técnicas a aumentar la densidad del *dataset* (ver Figura 9.2) no es suficiente para estos algoritmos. De hecho, en ciertos casos puede incluso llegar a empeorar ligeramente los resultados.

Por otra parte, también hemos estudiado el impacto de las técnicas de preprocesado por sí solas, sin eliminar sesiones pequeñas. Los resultados, como se muestra en la Figura 9.8, dependen en gran medida del *dataset* empleado. En el AOL, sólo *term reordering* parece mejorar la precisión de los algoritmos, teniendo las demás técnicas un impacto negativo. Sin embargo, en el *dataset* MSN, tanto *term reordering* por sí solo como combinado con *stopword removal* ofrecen una mejora en la precisión, que llega hasta cerca del 10 %, lo que es un gran resultado. De las técnicas estudiadas, sólo *stopword removal*, cuando no se combina con otras técnicas, ofrece siempre malos resultados. En cualquier caso, parece evidente que las técnicas de preprocesado son mucho más útiles cuando se combinan con la eliminación de sesiones pequeñas.

9.5. Conclusiones

En este capítulo hemos estudiado la aplicación de técnicas de filtrado colaborativo al problema de *Search Shortcuts*. La evaluación ha sido realizada usando datos reales de *query logs* de los *datasets* AOL y MSN. Hemos obtenido resultados bastante precisos, lo que demuestra que este tipo de algoritmos encaja bastante bien en este problema. Sin embargo, también hemos observado el bajo *coverage* de algunos algoritmos, lo que está relacionado con la baja densidad de este tipo de *datasets*. Sobre todo, las técnicas

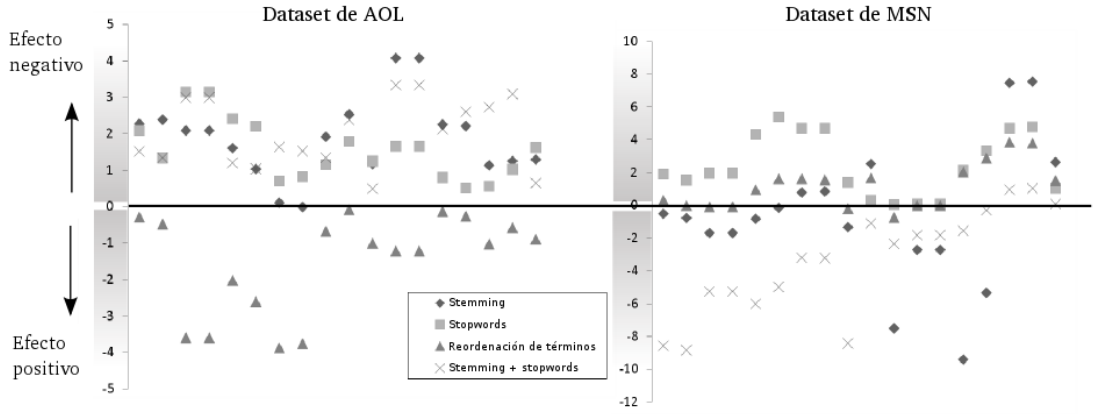


Figura 9.8: Efecto de distintas técnicas de preprocesado en la precisión, en los *datasets* de AOL y MSN. Los resultados están expresados en términos de porcentaje de variación respecto a la precisión sin aplicar ninguna de dichas técnicas.

tradicionales basadas en usuarios o elementos son especialmente sensibles a esta falta de información, debido a que sólo aprovechan una pequeña parte de la información disponible.

También se han estudiado varias técnicas para reducir la influencia de esta baja densidad. Hemos visto como algunos de estos métodos tienen un gran impacto en los resultados, dando lugar a mejoras significativas. En concreto, ignorar las sesiones con pocas consultas ha demostrado ser una técnica muy exitosa, mejorando la precisión de los algoritmos hasta en un 50 %. De la misma forma, técnicas de preprocesado como *term stemming* o *stopword removal* también son útiles, especialmente cuando se combinan con la anterior. Además, las mejoras se producen independientemente del *dataset* empleado.

Por otra parte, cuando se emplean otras métricas en lugar del error en la precisión, los resultados son modestos. Hemos relacionado este hecho tanto con la naturaleza de algunos algoritmos, como con el gran volumen y baja densidad de los datos, así como con las limitaciones de las métricas y metodologías de evaluación tradicionales. Las métricas de precisión en la clasificación obtienen resultados interesantes, pero sólo si se tienen en cuenta los detalles de la metodología de evaluación empleada. En concreto, el uso de *datasets* offline y poco densos impone varias limitaciones, ya que los elementos considerados relevantes por el algoritmo no pueden ser comparados con datos reales en muchos casos.

Para superar estos problemas se ha empleado una métrica especialmente diseñada para la evaluación del problema de *Search Shortcuts*. Su principal ventaja es que los

resultados pueden ser fácilmente interpretados en términos de cuán buenas son las recomendaciones del algoritmo para reducir la longitud de la sesión. Es decir, hasta qué punto ayudan a los usuarios a encontrar la información que necesitan en el menor tiempo posible. Además, permite evaluar diferentes aspectos del algoritmo, incluyendo su precisión según el significado tradicional de dicho término.

Finalmente, nos hemos encontrado con varias limitaciones que deberían ser tratadas en futuros trabajos. En primer lugar, el uso de tres niveles de valoraciones (positiva, negativa, neutral) no funciona tan bien como habíamos esperado, principalmente debido a que la mayor parte de las consultas son, de hecho, neutrales según este esquema. Esto da lugar a recomendaciones incorrectas que, lo que es peor, son consideradas como buenas por las métricas de precisión en la predicción. Además, la idea de interpretar un *clic* como éxito debería ser mejorada teniendo en cuenta más aspectos disponibles en el *query log*, como, por ejemplo, si el usuario ha visitado páginas de resultados adicionales, el tiempo transcurrido hasta que prueba con otra consulta, etc.

De la misma manera, ya que una sesión física puede estar formada por varias sesiones lógicas, deberían estudiarse técnicas más complejas de segmentación de sesiones.

Finalmente, sería interesante comparar los resultados de algoritmos de filtrado colaborativo con técnicas de *query suggestion* tradicionales, así como el desarrollo de nuevos algoritmos especialmente diseñados para este nuevo problema.

Capítulo 10

Conclusiones y trabajos futuros

Esta tesis ha estado enfocada al análisis de los algoritmos de recomendación en entornos dispersos, poniendo especial atención en los sistemas de filtrado colaborativo. A continuación especificaremos cuáles son las principales conclusiones obtenidas, para posteriormente indicar las contribuciones más destacables y los trabajos futuros que se pueden derivar de esta tesis.

10.1. Conclusiones

En el Capítulo 3 abordamos el estudio de la dispersión de los datos en los sistemas recomendadores centrándonos en el comportamiento de los algoritmos de filtrado colaborativo ante distintas condiciones en un dominio de recomendación de películas. Según los resultados alcanzados, el tipo del algoritmo determina en qué grado influye la densidad de los datos en la precisión obtenida. Este comportamiento está muy relacionado con la habilidad que tienen las distintos técnicas de interpretar los datos disponibles y extraer información útil. Mientras que en situaciones de alta densidad de datos la mayoría presentan resultados parecidos, en situaciones con gran dispersión estas diferencias se acentúan. Este hecho está causado, en gran medida, por las limitaciones que tienen a la hora de encontrar las similitudes entre los usuarios y/o productos, debido, por ejemplo, a la gran diversidad existente en gustos y opiniones.

En el Capítulo 4 mostramos cómo un análisis de los datos puede ser de ayuda para desarrollar algoritmos de filtrado colaborativo y para comprender el comportamiento de los usuarios de un dominio. De hecho, muchos de los problemas encontrados en la evaluación de los algoritmos tradicionales basados en usuarios pueden explicarse por medio de un estudio de los datos. Por tanto, este estudio permite también mejorar el diseño de los algoritmos y evitar así los errores comunes introducidos al malinterpretar

las peculiaridades de los datos. En concreto, en dicho capítulo abordamos el desarrollo de algoritmos para la tarea de predicción en contextos con alta dispersión centrándonos en los usuarios en situación de *cold-start*, es decir, usuarios que disponen de pocas puntuaciones y para los que es difícil que el sistema les genere una recomendación precisa. A raíz de las conclusiones obtenidas a partir del análisis realizado, propusimos una medida de similitud basada en el porcentaje de productos valorados en común por cada par de usuarios, que obtuvo ligeras mejoras con respecto a las medidas tradicionales en situaciones de escasa información, en donde éstas presentan ciertas limitaciones por ser demasiado estrictas. Aunque los resultados no proporcionaron un incremento significativo de la precisión, sí que demostraron la importancia del análisis previo.

En la tarea de recomendación la expansión de perfil ha permitido mejorar los resultados de los algoritmos de filtrado colaborativo en situaciones de *cold-start*, en donde la información acerca de los usuarios es escasa. Una de las grandes ventajas de estos algoritmos es que no requieren que el usuario proporcione nueva información. Sólo se basan en las puntuaciones hechas por los usuarios. En el Capítulo 5 logramos prometedores resultados con diferentes modos de combinar las técnicas de expansión. La razón de sus buenos resultados es que algoritmos de distinta naturaleza se pueden complementar al extraer información de diversos modos, lo cual provoca una mejoría de los resultados.

La expansión del perfil puede ser realizada, aparte de por técnicas colaborativas, por otros tipos de algoritmos. Este es el caso de los algoritmos basados en contenido. En el Capítulo 6 propusimos un algoritmo de este tipo. Demostramos que la información de contenido no sólo es útil para calcular la lista de productos, sino que también para la predicción de sus puntuaciones. Además, comprobamos que considerar la popularidad de los productos incrementa la precisión de los resultados notablemente, a costa de sesgar las recomendaciones hacia los productos más populares y, por tanto, perder personalización. Esto no impide que dicha característica pueda llegar a ser de gran utilidad en situaciones en donde la dispersión de los datos sea extrema, por ejemplo.

Para ahondar en cómo afecta la dispersión a los algoritmos de filtrado colaborativo, también hemos estudiado el funcionamiento de estos algoritmos en el dominio de la recomendación de consultas. En primer lugar, analizamos el *query log* público de MSN, para así poder comprender cómo se comportan los usuarios cuando realizan una búsqueda. En dicho análisis aplicamos el método de segmentación por tiempo con diferentes umbrales, por ser un método de segmentación ampliamente usado. De esta manera pudimos comprobar cómo los usuarios no acostumbran a dejar más de 20 minutos entre dos consultas consecutivas y que un 75 % de las sesiones lógicas son cortas, con no más de 2 consultas. Además, el número de clics que se realizan sobre cada consulta

tiende a ser bajo, menor de un clic por consulta en el *query log*. También comprobamos que los usuarios hacen estos clics sobre los primeros documentos de la lista ofrecida por el motor de búsqueda, lo cual deja entrever que el usuario suele confiar en el orden proporcionado por el motor. Otra conclusión de este estudio es que en una sesión lógica las consultas situadas al final reciben una media de clics ligeramente superior a la del resto de consultas. La razón es que es habitual que la última consulta esté más o menos bien formulada, por lo que es probable también que se encuentren documentos relevantes en ella y, por tanto, que estos documentos reciban clics por parte del usuario. A medida que el tamaño de la sesión lógica es más grande, los clics tienden a distribuirse entre más posiciones, debido, en general, a que se trata de procesos de búsqueda más complejos, que requieren un mayor esfuerzo por parte del usuario.

Por otro lado, las técnicas de procesamiento empleadas, como la reordenación de términos, *term stemming* o *stopwords*, a pesar de que no reducen significativamente el número de sesiones lógicas pequeñas, sí que hacen decrecer el número de sesiones totales, decrementando así la dispersión de los datos.

En cuanto a la frecuencia de las consultas, el análisis realizado mostró que ésta aumentaba conforme las sesiones eran más cortas y la consulta estaba más próxima a la primera posición de la sesión lógica. Ambas conclusiones eran previsibles, puesto que, dada la eficiencia de los motores de búsqueda actuales, los usuarios tienden a comenzar las sesiones con consultas sencillas (más genéricas y frecuentes), para después ir complicándolas en función de los resultados que el motor proporcione. Las sesiones pequeñas se corresponden generalmente con necesidades de información fáciles de satisfacer, que requieren menos esfuerzo, por lo que las consultas suelen ser también más simples y populares.

Usando como indicador del éxito o el fracaso de una sesión lógica la presencia o ausencia de clics en la última consulta de dicha sesión, en el *query log* estudiado existen pocas sesiones exitosas, especialmente teniendo en cuenta las sesiones largas. Esto sucede porque bien los usuarios escogen incorrectamente la primera consulta, o bien no son capaces de seguir el camino de búsqueda correcto para satisfacer su necesidad de información. Partiendo de esta idea, una de las conclusiones más importantes que obtuvimos del análisis del *query log* es que usuarios con caminos de búsqueda exitosos podrían ayudar a otros usuarios a lograr sesiones también exitosas. En el Capítulo 9 abordamos este reto, intentando ahorrar tiempo y esfuerzo a los usuarios. Para ello aplicamos algoritmos de filtrado colaborativo al problema de *Search Shortcuts*. Tras haber realizado la evaluación con dos *query logs* públicos, el de AOL y el de MSN, observamos que, a pesar de que los algoritmos empleados encajan bien en el proble-

ma mencionado, los resultados se vieron visiblemente afectados por la gran dispersión de los datos presentes en este dominio. De nuevo, podemos atribuir este hecho a las dificultades que tienen determinados algoritmos de extraer información en situaciones de baja densidad de datos. Aunque en ciertos casos se alcanzan buenas precisiones, estos resultados se producen porque estos algoritmos sólo son capaces de recomendar un porcentaje pequeño de las consultas, es decir, tienen un bajo *coverage*.

Para intentar reducir la influencia de la dispersión de los datos aplicamos las técnicas de preprocesamiento estudiadas en el Capítulo 8. Algunas de estas técnicas obtuvieron una mejora significativa de los resultados. Además, vimos también que ignorar las sesiones pequeñas que no resultaban útiles para el problema a tratar, tuvo un gran impacto positivo. De hecho, los mejores resultados se consiguieron combinando esta última técnica con las técnicas de preprocesado comentadas anteriormente.

Estas conclusiones muestran cómo los resultados obtenidos a partir de las ideas propuestas en esta tesis para abordar la dispersión de los datos en diferentes dominios suponen un punto de partida para nuevos trabajos, que comentaremos en la Sección 10.3.

10.2. Contribuciones

Partiendo de los objetivos presentados en la Sección 1.2 y teniendo en cuenta las conclusiones generales recientemente comentadas, las principales contribuciones de esta tesis son las siguientes:

- Estudio del estado del arte de los algoritmos de filtrado colaborativo ante diferentes condiciones de densidad de datos. Se trata de un estudio con diferentes algoritmos de filtrado colaborativo, con el que se comprueba cómo evoluciona cada uno de ellos conforme el conjunto de datos se hace más disperso. Esta evolución supone la principal diferencia entre las técnicas. De esta manera se puede apreciar la dificultad de extraer información que tienen las técnicas colaborativas en situaciones de poca densidad de datos. Sin embargo, cuando hay suficiente densidad de datos, no existen diferencias significativas entre los algoritmos en cuanto a precisión.

- Propuesta de una medida de similitud para los algoritmos basados en usuario en situaciones de dispersión de datos. El análisis de algunos de los *datasets* más populares para la recomendación de películas mostró las limitaciones en contextos dispersos de las medidas de similitud tradicionales usadas por los algoritmos

colaborativos basados en vecinos en la tarea de predicción. Dado que en dichos contextos estas medidas son demasiado estrictas, propusimos una nueva medida de similitud pensada precisamente para estas situaciones. En particular, abordamos la mejora de los resultados para los usuarios que poseen pocas puntuaciones, amparados en que un porcentaje muy pequeño de los usuarios genera un porcentaje significativo de las puntuaciones, lo cual implica que un número importante de usuarios realizan pocas valoraciones.

- Mejora de los algoritmos basados en usuario en la tarea de recomendación aplicando la técnica de expansión de perfil de orden superior. Los algoritmos de expansión de perfil permiten abordar el problema de *cold-start* y, más concretamente, el problema del nuevo usuario, aumentando el perfil de los usuarios con pocas puntuaciones y, por tanto, disminuyendo la dispersión. Se presentaron dos modos diversos de combinar estas técnicas: en serie y en paralelo. Los resultados obtenidos demostraron que es preferible la combinación de los algoritmos frente al empleo de cada uno de ellos individualmente.

- Mejora de los algoritmos basados en usuario en la tarea de recomendación aplicando la técnica de expansión de perfil basada en contenido. La información de contenido puede ser de ayuda en situaciones en donde la información colaborativa es escasa. En concreto, propusimos la expansión del perfil colaborativo por medio de técnicas basadas en contenido para tratar situaciones de nuevo usuario. A raíz de los resultados, concluimos que la información de contenido es útil no sólo para la selección de los productos a añadir al perfil, sino también para la predicción de las puntuaciones con las que se añaden.

- Análisis de un *query log* público. Para poder trabajar con un *query log* es conveniente previamente haberlo analizado, para así comprender cómo se comportan los usuarios que usan el servicio a la hora de realizar búsquedas. De esta manera se podrán desarrollar técnicas adaptadas a los usuarios en cuestión, que les hagan más sencillo su proceso de búsqueda El análisis realizado lo centramos en varios aspectos: las segmentaciones del *query log* en base a diferentes umbrales de tiempo, el comportamiento de los usuarios a la hora de realizar los clics sobre las consultas, la aplicación de técnicas de preprocesado para reducir el número de consultas únicas y así conseguir un *dataset* menos disperso, y en el estudio del éxito o el fracaso de una determinada búsqueda.

- Aplicación de las técnicas colaborativas en el dominio de la recomendación de

consultas. El estudio de los algoritmos colaborativos en la recomendación de consultas nos permitió analizar la influencia de una acusada dispersión de datos en los resultados de dichos algoritmos. Para ello tuvimos que abordar ciertos retos como la segmentación de las sesiones de usuario en tareas de búsqueda, la extracción de la relevancia de las consultas a partir de los datos del *query log* o el mero hecho de encontrar una correspondencia entre los conceptos básicos de filtrado colaborativo (usuario, producto y puntuaciones) y de la recomendación de consultas (usuario, sesión y consultas).

10.2.1. Publicaciones y estancias de investigación

La elaboración de esta tesis ha traído consigo diversas publicaciones[1]. De entre ellas, las siguientes son publicaciones en revistas de carácter científico:

- F. Cacheda, V. Carneiro, D. Fernández, y V. Formoso. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on Web*, 5:2:1–2:33, Febrero 2011. ISSN: 1559-1131.

- V. Formoso, D. Fernández, F. Cacheda y V. Carneiro. Using profile expansion techniques to alleviate the new user problem. *Information Processing & Management*, 49(3):659–672, Mayo 2013. ISSN: 0306-4573.

- V. Formoso, D. Fernández, F. Cacheda y V. Carneiro. Using rating matrix compression techniques to speed up collaborative recommendations. *Information Retrieval*, 16(6):680–696, 2013. ISSN: 1386-4564.

- D. Fernández, V. Formoso, F. Cacheda y V. Carneiro. A content-based approach to Profile Expansion. Enviado a *Information Processing Letters*. 2014.

- D. Fernández, V. Formoso, F. Cacheda y V. Carneiro. HOPE techniques to tackle the cold-start problem. Enviado a *User Modeling and User-Adapted Interaction*. 2014.

A continuación, citamos las publicaciones tanto en congresos nacionales como internacionales:

[1]Hasta diciembre de 2011 los autores se ordenaron alfabéticamente; a partir de entonces, se empleó el orden de contribución.

- R. Baraglia, F. Cacheda, V. Carneiro, D. Fernández, V. Formoso, R. Perego y F. Silvestri. Search shortcuts: a new approach to the recommendation of queries. En *ACM Conference on Recommender Systems, RecSys'09*, páginas 77–84, New York, NY, USA, 2009. ACM. ISBN: 978-1-60558-435-5.

- F. Cacheda, V. Carneiro, D. Fernández y V. Formoso. Search Shortcuts: recomendación de consultas en buscadores web. En *VIII Jornadas de Ingeniería Telemática, JITEL 2009*, páginas 237-244. Cartagena, 2009. ISBN: 978-84-96997-27-1.

- F. Cacheda, V. Carneiro, D. Fernández y V. Formoso. Eficiencia y precisión de algoritmos de filtrado colaborativo: análisis y comparativa. *I Congreso Español de Recuperación de Información, CERI 2010*, páginas 275-282. Madrid, 2010. ISBN: 978-84-693-2200-0.

- F. Cacheda, V. Carneiro, D. Fernández y V. Formoso. Improving k-nearest neighbors algorithms: practical application of dataset analysis. En *ACM International Conference on Information and Knowledge Management, CIKM'11*, páginas 2253–2256, Glasgow, Scotland, UK, 2011. ACM. ISBN: 978-1-4503-0717-8.

- V. Formoso, D. Fernández, F. Cacheda, y V. Carneiro. Using neighborhood precomputation to increase recommendation efficiency. En *International Conference on Knowledge Discovery and Information Retrieval, KDIR 2012*, páginas 333–335, Barcelona, Spain, 2012. SciTePress. ISBN: 978-989-8565-29-7.

También se ha presentado una solicitud de patente para el método de recomendación explicado en la Sección 3.3.3.5:

- Fidel Cacheda, Víctor Carneiro, Vreixo Formoso, Diego Fernández, Ana Freire. Recommendation Method and System. U.S. Patent Application, 61/492,206. Jun. 1, 2011.

Finalmente, durante el desarrollo de esta tesis se han realizado dos estancias de investigación:

- 1 de junio de 2009 - 30 de junio de 2009. Estancia en el ISTI-CNR (*Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo" – Consiglio Nazionale delle Ricerche*) en Pisa (Italia) bajo la supervisión de Dr. Fabrizio Silvestri.

- 24 de marzo de 2010 - 2 de agosto de 2010. Estancia en el ISTI-CNR (*Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo" – Consiglio Nazionale delle Ricerche*) en Pisa (Italia) bajo la supervisión de Dr. Fabrizio Silvestri.

10.3. Trabajos futuros

En primer lugar hemos hecho un estudio de diferentes técnicas colaborativas y cómo les afecta la dispersión de los datos. Este estudio puede ser ampliado tanto con la inclusión de nuevos algoritmos como con el análisis de nuevos dominios, lo cual resultaría relevante para la comunidad, puesto que no existen abundantes trabajos al respecto. A partir de las conclusiones obtenidas acerca del análisis realizado de los *datasets* de recomendación de películas se podrían elaborar nuevos modelos que se ajusten mejor a los datos, que podríamos introducir también en el estudio.

En cuanto al problema de la dispersión, lo hemos abordado tanto para la tarea de predicción como para el de recomendación. El problema de la predicción en situaciones de baja densidad de datos hemos propuesto medidas de similitud sencillas para los algoritmos basados en vecinos. Como ya comentamos en el Capítulo 4, sería interesante el estudio de una nueva medida en donde se ponderase dinámicamente en función del número de productos que haya valorado cada usuario la contribución del número de productos valorados en común y de la coincidencia en las puntuaciones realizadas. Además, las medidas propuestas pueden ser combinadas con medidas de similitud tradicionales.

Con respecto al estudio de la tarea de recomendación, los trabajos sobre las técnicas de expansión de perfil pueden ser continuados desde muchos frentes. En particular, aparte de los algoritmos basados en vecinos se podrían usar otros algoritmos de filtrado colaborativo. Además, la expansión de orden superior permite múltiples combinaciones. Se puede optar por una solución recursiva, usando diferentes algoritmos de filtrado colaborativo en diferentes iteraciones de la expansión. Aunque hasta el momento sólo hemos propuesto combinaciones dos a dos de algoritmos de expansión, el número de algoritmos combinados puede ser ampliado, sobre todo teniendo en cuenta que también sería interesante el estudio de nuevas fuentes de información como datos demográficos, las relaciones entre usuarios en las redes sociales, etc.

Otro aspecto interesante que abordar en el futuro acerca de la expansión de perfil es el de añadir un número diferente de productos a la expansión en función del tamaño del perfil de cada usuario, puesto que no todos los usuarios del sistema tienen por qué estar en situación de *cold-start*.

Aunque hemos tratado la posibilidad de realizar expansiones del perfil del usuario

activo, existe también la alternativa de expandir la matriz globalmente, es decir, aumentar los perfiles de todos los usuarios o, al menos, de aquéllos que no hayan valorado un número mínimo de productos.

También puede ser estudiado cómo influyen en los resultados finales las medidas de similitud de los algoritmos kNN tanto del algoritmo de recomendación, como de los algoritmos de expansión. El número de vecinos también puede ser modificado según la similitud que tengan con el usuario actual.

Como se puede comprobar, la expansión del perfil supone un tema que abre un gran abanico de futuros trabajos posibles, que podrían ser de gran utilidad para afrontar el problema de la dispersión de los datos.

Por otro lado, el análisis realizado del *query log* de MSN se podría ampliar de formas diversas. Una de ellas, de gran interés, es la aplicación de otros algoritmos de segmentación de sesiones más complejos. Así, se podría comprobar si difieren las conclusiones obtenidas de las presentadas en esta tesis.

En cuanto a la aplicación de algoritmos de filtrado colaborativo al problema de Search Shortcuts, nos hemos encontrado con varias limitaciones que deberían ser tratadas en futuros trabajos. En primer lugar, el uso de tres niveles de valoraciones (positiva, negativa, neutral) no funciona tan bien como habíamos esperado, por lo que el estudio de otros nuevos esquemas puede mejorar los resultados obtenidos. Aunque en la literatura la idea de interpretar un clic en la última consulta de la sesión lógica se ha empleado en repetidas ocasiones, podríamos tener en cuenta más información extraíble del *query log* como el número de resultados sobre los que el usuario ha hecho clic, el tiempo transcurrido entre las distintas consultas, etc. El estudio de otros métodos de segmentación de sesiones podrían ser, de nuevo, de gran interés.

Finalmente, planeamos el empleo de algoritmos de expansión de perfil en la recomendación de consultas, ya que tuvieron un impacto muy positivo en la recomendación de películas.

Bibliografía

Gediminas Adomavicius y Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, 17(6):734–749, 2005. 5, 10

Farooq Ahmad y Grzegorz Kondrak. Learning a spelling error model from search query logs. En *Proceedings of the 2005 Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, páginas 955–962, Vancouver, Canada, Octubre 2005. Association for Computational Linguistic. 133

Hyung Jun Ahn. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Inf. Sci.*, 178:37–51, Enero 2008. ISSN 0020-0255. 63

Xavier Amatriain, Josep M. Pujol, y Nuria Oliver. I like it... i like it not: Evaluating user ratings noise in recommender systems. En *Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization: formerly UM and AH*, UMAP '09, páginas 247–258, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-02246-3. 14, 26

Peter G. Anick. Using terminological feedback for web search refinement: a log-based study. En *SIGIR*, páginas 88–95. ACM, 2003. ISBN 1-58113-646-3. 129

R. Attar y A. S. Fraenkel. Local feedback in full-text retrieval systems. *J. ACM*, 24: 397–417, Julio 1977. ISSN 0004-5411. 88

Esma Aïmeur, Gilles Brassard, José; M. Fernandez, y Flavien Serge Mani Onana. Alambic: a privacy-preserving recommender system for electronic commerce. *Int. J. Inf. Secur.*, 7(5):307–334, Septiembre 2008. ISSN 1615-5262. 21

Ricardo Baeza-Yates. *Web Mining: Applications and Techniques*, chapter Query Usage Mining in Search Engines, páginas 307–321. Idea Group, 2004. 124

Ricardo Baeza-Yates y Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Publishing Company, USA, 2nd edition, 2008. ISBN 0321416910, 9780321416919. 84

Ricardo Baeza-Yates, Carlos Hurtado, y Marcelo Mendoza. Ranking boosting based in query clustering. En *Proceedings of 2004 Atlantic Web Intelligence Conference*, Cancun, Mexico, 2004a. 134

Ricardo Baeza-Yates, Carlos Hurtado, y Marcelo Mendoza. Query clustering for boosting web page ranking. *Advances in Web Intelligence*, páginas 164–175, 2004b. 133

BIBLIOGRAFÍA

Ricardo Baeza-Yates, Aristides Gionis, Flavio Junqueira, Vanessa Murdock, Vassilis Plachouras, y Fabrizio Silvestri. The impact of caching on search engines. En *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, páginas 183–190, New York, NY, USA, 2007a. ACM. ISBN 978-1-59593-597-7. 152

Ricardo A. Baeza-Yates. Applications of web query mining. En David E. Losada y Juan M. Fernández-Luna, editores, *Advances in Information Retrieval, 27th European Conference on IR Research, ECIR 2005, Santiago de Compostela, Spain, March 21-23, 2005, Proceedings*, volumen 3408 de *Lecture Notes in Computer Science*, páginas 7–22. Springer, 2005. ISBN 3-540-25295-9. 124, 133, 140

Ricardo A. Baeza-Yates. Graphs from search engine queries. En Jan van Leeuwen, Giuseppe F. Italiano, Wiebe van der Hoek, Christoph Meinel, Harald Sack, y Frantisek Plasil, editores, *SOFSEM 2007: Theory and Practice of Computer Science, 33rd Conference on Current Trends in Theory and Practice of Computer Science, Harrachov, Czech Republic, January 20-26, 2007, Proceedings*, volumen 4362 de *Lecture Notes in Computer Science*, páginas 1–8. Springer, 2007. ISBN 978-3-540-69506-6. 129

Ricardo A. Baeza-Yates y Berthier A. Ribeiro-Neto. *Modern Information Retrieval - the concepts and technology behind search, Second edition.* Pearson Education Ltd., Harlow, England, 2011. ISBN 978-0-321-41691-9. 126, 132

Ricardo A. Baeza-Yates, Carlos A. Hurtado, y Marcelo Mendoza. Query recommendation using query logs in search engines. En Wolfgang Lindner, Marco Mesiti, Can Türker, Yannis Tzitzikas, y Athena Vakali, editores, *Current trends in database technology - EDBT 2004 workshops*, volumen 3268 de *Lecture Notes in Computer Science*, páginas 588–596. Springer, 2004c. ISBN 3-540-23305-9. 133, 134

Ricardo A. Baeza-Yates, Liliana Calderón-Benavides, y Cristina N. González-Caro. The intention behind web queries. En Fabio Crestani, Paolo Ferragina, y Mark Sanderson, editores, *SPIRE*, volumen 4209 de *Lecture Notes in Computer Science*, páginas 98–109. Springer, 2006. ISBN 3-540-45774-7. 127

Ricardo A. Baeza-Yates, Carlos A. Hurtado, y Marcelo Mendoza. Improving search engines by query clustering. *JASIST*, 58(12):1793–1804, 2007b. 133, 134

Marko Balabanović y Yoav Shoham. Fab: content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, 1997. ISSN 0001-0782. 62

Evelyn Balfe y Barry Smyth. Improving web search through collaborative query recommendation. En Ramon López de Mántaras y Lorenza Saitta, editores, *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004*, páginas 268–272. IOS Press, 2004. ISBN 1-58603-452-9. 134

Ranieri Baraglia, Fidel Cacheda, Victor Carneiro, Diego Fernandez, Vreixo Formoso, Raffaele Perego, y Fabrizio Silvestri. Search shortcuts: a new approach to the recommendation of queries. En *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, páginas 77–84, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-435-5. 160

Deborah Barreau y Bonnie A. Nardi. Finding and reminding: file organization from the desktop. *SIGCHI Bull.*, 27(3):39–43, Julio 1995. ISSN 0736-6906. 126

M. J. Bates. Information search tactics. *Journal of the American Society for Information Science*, 30(4):205–214, 1979. 126

Marcia J. Bates. The design of browsing and berrypicking techniques for the online search interface. *Online review*, 13(5):407–431, 1989. 126

Marcia J. Bates. Where should the person stop and the information search interface start? *Inf. Process. Manage.*, 26(5):575–591, 1990. 126, 141, 153

Doug Beeferman y Adam Berger. Agglomerative clustering of a search engine query log. En *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, páginas 407–416, New York, NY, USA, 2000. ACM. ISBN 1-58113-233-6. 133, 134

Steven M. Beitzel, Eric C. Jensen, Abdur Chowdhury, Ophir Frieder, y David Grossman. Temporal analysis of a very large topically categorized web query log. *J. Am. Soc. Inf. Sci. Technol.*, 58(2):166–178, 2007. ISSN 1532-2882. 125

Robert M. Bell y Yehuda Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. En *ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, páginas 43–52, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-3018-4. 26

James Bennett y Stam Lanning. The netflix prize. En *Proceedings of KDD Cup and Workshop*, KDDCup '07, páginas 3–6, San Jose, California, USA, 2007. ACM. 10, 23, 43, 44, 90, 102, 111

Dania Bilal. Children's use of the yahooligans! web search engine. iii. cognitive and physical behaviors on fully self-generated search tasks. *J. Am. Soc. Inf. Sci. Technol.*, 53(13):1170–1183, Noviembre 2002. ISSN 1532-2882. 126

Daniel Billsus y Michael J. Pazzani. Learning collaborative information filters. En *Proc. 15th International Conf. on Machine Learning*, páginas 46–54. Morgan Kaufmann, San Francisco, CA, 1998. 16

JesúS Bobadilla, Fernando Ortega, Antonio Hernando, y JesúS Bernal. A collaborative filtering approach to mitigate the new user cold start problem. *Know.-Based Syst.*, 26:225–238, Febrero 2012. ISSN 0950-7051. 63

Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, Aristides Gionis, y Sebastiano Vigna. The query-flow graph: model and applications. En *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, páginas 609–618, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-991-3. 132

Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, y Sebastiano Vigna. Query suggestions using query-flow graphs. En *WSCD '09: Proceedings of the 2009 workshop on Web Search Click Data*, páginas 56–63, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-434-8. 132

BIBLIOGRAFÍA

John S. Breese, David Heckerman, y Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. En *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, UAI'98, páginas 43–52, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1-55860-555-X. 25, 31, 34, 42, 46, 78, 90, 111, 169

D. Bridge, M. Goker, L. McGinty, y B. Smyth. Case-based recommender systems. *Knowledge Engineering Review*, 20(3):315–320, 2006. 17

Andrei Z. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002. 127

Andrei Z. Broder, Marcus Fontoura, Evgeniy Gabrilovich, Amruta Joshi, Vanja Josifovski, y Tong Zhang. Robust classification of rare queries using web knowledge. En *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, páginas 231–238, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-597-7. 127

Rebecca Bulander, Michael Decker, Gunther Schiefer, y Bernhard Kolmel. Comparison of different approaches for mobile advertising. En *WMCS '05: Proceedings of the Second IEEE International Workshop on Mobile Commerce and Services*, páginas 174–182, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2391-9. 11

Robin Burke. Knowledge-based recommender systems. *Encyclopedia of Library and Information Science*, 69(32):180–200, 2000. 16, 17

Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, Noviembre 2002. ISSN 0924-1868. 15, 18

Robin Burke. Hybrid web recommender systems. En *The Adaptive Web: Methods and Strategies of Web Personalization*, chapter 12, páginas 377–408. Berlin, 2007. 18

Nikolai (Nick) Buzikashvili y Bernard J. Jansen. Limits of the web log analysis artifacts. En *Workshop on Logging Traces of Web Activity: The Mechanics of Data Collection*, May 2006. 129, 132

Fidel Cacheda, Victor Carneiro, Diego Fernández, y Vreixo Formoso. Search shortcuts: recomendación de consultas en buscadores web. En *Proceedings of the VIII Jornadas de Ingeniería Telemática*, JITEL '09, páginas 237–244, Cartagena, Spain, 2009. ISBN 978-84-96997-27-1. 160

Fidel Cacheda, Victor Carneiro, Diego Fernández, y Vreixo Formoso. Eficiencia y precisión de algoritmos de filtrado colaborativo: análisis y comparativa. En *Proceedings of the I Congreso Español de Recuperación de Información*, CERI '10, páginas 275–282, Madrid, Spain, 2010. ISBN 978-84-693-2200-0. 29

Fidel Cacheda, Víctor Carneiro, Diego Fernández, y Vreixo Formoso. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Trans. Web*, 5:2:1–2:33, Febrero 2011a. ISSN 1559-1131. 23, 29, 32, 40, 60, 71, 168, 169

Fidel Cacheda, Victor Carneiro, Diego Fernández, y Vreixo Formoso. Improving k-nearest neighbors algorithms: practical application of dataset analysis. En *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, páginas 2253–2256, New York, NY, USA, 2011b. ACM. ISBN 978-1-4503-0717-8. 61

John Canny. Collaborative filtering with privacy. En *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, SP '02, páginas 45–57, Washington, DC, USA, 2002a. IEEE Computer Society. ISBN 0-7695-1543-6. 21

John Canny. Collaborative filtering with privacy via factor analysis. En *Proceedings of the 25th annual international ACM SIGIR conference on research and development in information retrieval*, SIGIR '02, páginas 238–245, New York, NY, USA, 2002b. ACM. ISBN 1-58113-561-0. 61

Lara D. Catledge y James E. Pitkow. Characterizing browsing strategies in the world-wide web. *Computer Networks and ISDN Systems*, 27(6):1065–1073, 1995. 129

Mao Chen, Andrea S. LaPaugh, y Jaswinder Pal Singh. Predicting category accesses for a user in a structured information space. En *SIGIR*, páginas 65–72, 2002. 129

Qing Chen, Mu Li, y Ming Zhou. Improving query spelling correction using web search results. En *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, páginas 181–189, Prague, Czech Republic, Junio 2007. Association for Computational Linguistic. 133

Paul-Alexandru Chirita, Wolfgang Nejdl, y Cristian Zamfir. Preventing shilling attacks in online recommender systems. En *Proceedings of the 7th annual ACM international workshop on Web information and data management*, WIDM '05, páginas 67–74, New York, NY, USA, 2005. ACM. ISBN 1-59593-194-5. 20, 60

Cynthia L. Corritore Christine Jenkins y Susan Wiedenbeck. Patterns of information seeking on the web: A qualitative study of domain expertise and web expertise. it & society 1(3): 64. *IT & Society*, 1:64–89, 2003. 127

M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, y M. Sartin. Combining content-based and collaborative filters in an online newspaper. En *Proceedings of the ACM SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation*, Berkeley, California, 1999. ACM. 20, 63

Dan Cosley, Shyong K. Lam, Istvan Albert, Joseph A. Konstan, y John Riedl. Is seeing believing?: how recommender system interfaces affect users' opinions. En *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '03, páginas 585–592, New York, NY, USA, 2003. ACM. ISBN 1-58113-630-7. 14

Paolo Cremonesi, Yehuda Koren, y Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. En *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, páginas 39–46, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-906-0. 43, 68

Paolo Cremonesi, Franca Garzottto, y Roberto Turrin. User effort vs. accuracy in rating-based elicitation. En *Proceedings of the sixth ACM conference on Recommender systems*, RecSys '12, páginas 27–34, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1270-7. 63

Bruce W. Croft, Stephen Cronen-Townsend, y Victor Lavrenko. Relevance feedback and personalization: A language modeling perspective. En *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*, 2001. 128

BIBLIOGRAFÍA

Silviu Cucerzan y Eric Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. En *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, páginas 293–300, Julio 2004. 133

Silviu Cucerzan y Ryen W. White. Query suggestion based on user landing pages. En *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, páginas 875–876, New York, NY, USA, 2007. ACM Press. ISBN 9781595935977. 133

Hang Cui, Ji-Rong Wen, Jian-Yun Nie, y Wei-Ying Ma. Probabilistic query expansion using query logs. En *WWW '02: Proceedings of the 11th international conference on World Wide Web*, páginas 325–332, New York, NY, USA, 2002. ACM. ISBN 1-58113-449-5. 134

Mukund Deshpande y George Karypis. Item-based top-N recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004. ISSN 1046-8188. 34

Christian Desrosiers y George Karypis. A comprehensive survey of neighborhood-based recommendation methods. En Ricci et al. [2011], páginas 107–144. ISBN 978-0-387-85819-7. 60

Doug Downey, Susan T. Dumais, y Eric Horvitz. Models of searching and browsing: Languages, studies, and application. En *IJCAI*, páginas 2740–2747, 2007. 129

Mehdi Elahi, Valdemaras Repsys, y Francesco Ricci. Rating Elicitation Strategies for Collaborative Filtering E-Commerce and Web Technologies. volumen 85 de *Lecture Notes in Business Information Processing*, chapter 14, páginas 160–171. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-23013-4. 63

Jordan Ellenberg. This Psychologist Might Outsmart the Math Brains Competing for the Netflix Prize. *Wired Magazine*, 16(3), 2008. 15

Martin Ester, Hans-Peter Kriegel, Jörg Sander, y Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. En Evangelos Simoudis, Jiawei Han, y Usama M. Fayyad, editores, *KDD*, páginas 226–231. AAAI Press, 1996. ISBN 1-57735-004-9. 39

A. Felfernig y R. Burke. Constraint-based recommender systems: technologies and research issues. En *Proceedings of the 10th international conference on Electronic commerce*, ICEC '08, páginas 3:1–3:10, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-075-3. 17

Alexander Felfernig, Gerhard Friedrich, Dietmar Jannach, y Markus Zanker. An integrated environment for the development of knowledge-based recommender applications. *Int. J. Electron. Commerce*, 11(2):11–34, Diciembre 2006. ISSN 1086-4415. 17

Larry Fitzpatrick y Mei Dent. Automatic feedback using past queries: social searching? En *SIGIR '97: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, páginas 306–313, New York, NY, USA, 1997. ACM. ISBN 0-89791-836-3. 133

Bruno M. Fonseca, Paulo B. Golgher, Edleno S. de Moura, y Nivio Ziviani. Using association rules to discover search engines related queries. En *LA-WEB '03: Proceedings of the First Conference on Latin American Web Congress*, página 66, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-2058-8. 133, 134

Bruno M. Fonseca, Paulo Golgher, Bruno Pôssas, Berthier Ribeiro-Neto, y Nivio Ziviani. Concept-based interactive query expansion. En *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, páginas 696–703, New York, NY, USA, 2005. ACM. ISBN 1-59593-140-6. 133

Vreixo Formoso, Diego Fernández, Fidel Cacheda, y Victor Carneiro. Using neighborhood pre-computation to increase recommendation efficiency. En Ana L. N. Fred, Joaquim Filipe, Ana L. N. Fred, y Joaquim Filipe, editores, *KDIR*, páginas 333–335. SciTePress, 2012. ISBN 978-989-8565-29-7. 26

Vreixo Formoso, Diego Fernández, Fidel Cacheda, y Victor Carneiro. Using profile expansion techniques to alleviate the new user problem. *Inf. Process. Manage.*, 49(3):659–672, Mayo 2013a. ISSN 0306-4573. 82, 84, 86, 87, 91, 112

Vreixo Formoso, Diego Fernández, Fidel Cacheda, y Victor Carneiro. Using rating matrix compression techniques to speed up collaborative recommendations. *Information Retrieval*, 16(6):680–696, 2013b. ISSN 1386-4564. 26

Nir Friedman, Dan Geiger, y Moises Goldszmidt. Bayesian network classifiers. *Mach. Learn.*, 29(2-3):131–163, Noviembre 1997. ISSN 0885-6125. 42

Simon Funk. Netflix Update: Try This at Home. `http://sifter.org/~simon/journal/20061211.html`, 2006. 32, 38

Evgeniy Gabrilovich y Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. En *Proceedings of the 20th international joint conference on Artifical intelligence*, IJCAI'07, páginas 1606–1611, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc. 132

Daniel Gayo-Avello. A survey on session detection methods in query logs and a proposal for future evaluation. *Inf. Sci.*, 179(12):1822–1843, 2009. 128, 131

Thomas George y Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. En *Proceedings of the Fifth IEEE International Conference on Data Mining*, ICDM '05, páginas 625–628, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2278-5. 40

Sharad Goel, Andrei Broder, Evgeniy Gabrilovich, y Bo Pang. Anatomy of the long tail: ordinary people with extraordinary tastes. En *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, páginas 201–210, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-889-6. 152

Nadav Golbandi, Yehuda Koren, y Ronny Lempel. Adaptive bootstrapping of recommender systems using decision trees. En *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, páginas 595–604, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0493-1. 63

Jennifer Golbeck. Generating predictive movie recommendations from trust in social networks. En *Proceedings of the 4th international conference on Trust Management*, iTrust'06, páginas 93–104, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-34295-8, 978-3-540-34295-3. 18

BIBLIOGRAFÍA

David Goldberg, David Nichols, Brian M. Oki, y Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992. 5, 10, 29, 81

Nathaniel Good, J. Ben Schafer, Joseph A. Konstan, Al Borchers, Badrul Sarwar, Jon Herlocker, y John Riedl. Combining collaborative filtering with personal agents for better recommendations. En *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, AAAI '99/IAAI '99, páginas 439–446, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence. ISBN 0-262-51106-1. 26, 62

Laura A. Granka, Thorsten Joachims, y Geri Gay. Eye-tracking analysis of user behavior in www search. En *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, páginas 478–479, New York, NY, USA, 2004. ACM. ISBN 1-58113-881-4. 128

Asela Gunawardana y Christopher Meek. Tied boltzmann machines for cold start recommendations. En *Proceedings of the 2008 ACM conference on Recommender systems*, RecSys '08, páginas 19–26, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-093-7. 62

Ayse Göker y Daqing He. Analysing web search logs to determine session boundaries for user-oriented learning. En Peter Brusilovsky, Oliviero Stock, y Carlo Strapparava, editores, *AH*, volumen 1892 de *Lecture Notes in Computer Science*, páginas 319–322. Springer, 2000. ISBN 3-540-67910-3. 129

Matthias Hagen, Jakob Gomoll, Anna Beyer, y Benno Stein. From Search Session Detection to Search Mission Detection. En *10th International Conference Open Research Areas in Information Retrieval (OAIR 13)*, páginas 85–92. ACM, Mayo 2013. 128, 132

Michael Hahsler. Developing and testing top-n recommendation algorithms for 0-1 data using recommenderlab, 2010. 61

Mark H. Hansen y Elizabeth Shriver. Using navigation data to improve ir functions in the context of web search. En *Proceedings of the tenth international conference on Information and knowledge management*, CIKM '01, páginas 135–142, New York, NY, USA, 2001. ACM. ISBN 1-58113-436-3. 128

Daqing He y Ayse Göker. Detecting session boundaries from web user logs. En *Proceedings of of the BCS-IRSG 22nd Annual Colloquium on Information Retrieval Research*, páginas 57–66, 2000. 128, 129, 130

Daqing He, Ayse Göker, y David J. Harper. Combining evidence for automatic web session identification. *Inf. Process. Manage.*, 38(5):727–742, 2002. ISSN 0306-4573. 128, 129, 130

Jon Herlocker, Joseph A. Konstan, y John Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Inf. Retr.*, 5(4):287–310, 2002. ISSN 1386-4564. 32, 35, 53, 54, 55, 167

Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, y John Riedl. An algorithmic framework for performing collaborative filtering. En *Proceedings of the 22nd annual*

international ACM SIGIR conference on Research and development in information retrieval, SIGIR '99, páginas 230–237, New York, NY, USA, 1999. ACM. ISBN 1-58113-096-1. 14, 26, 33, 43, 109, 167

Jonathan L. Herlocker, Joseph A. Konstan, y John Riedl. Explaining collaborative filtering recommendations. En *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, CSCW '00, páginas 241–250, New York, NY, USA, 2000. ACM. ISBN 1-58113-222-0. 22, 26

Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, y John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004. ISSN 1046-8188. 12, 13, 21, 22, 23, 25, 26, 32, 34, 46, 101, 169

Morten Hertzum y Erik Frøkjær. Browsing and querying in online documentation: A study of user interfaces and the interaction process. *ACM Trans. Comput.-Hum. Interact.*, 3(2):136–161, 1996. 126, 141, 153

Will Hill, Larry Stead, Mark Rosenstein, y George Furnas. Recommending and evaluating choices in a virtual community of use. En *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '95, páginas 194–201, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-84705-1. 10, 14

T. Hofmann y J. Puzicha. Latent class models for collaborative filtering. En *Proc. of the Sixteenth International Joint Conference on Artificial Intelligence*, páginas 688–693, Stockholm, 1999. 42

Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Mach. Learn.*, 42(1-2):177–196, Enero 2001. ISSN 0885-6125. 42

Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004. ISSN 1046-8188. 42

Chien-Kang Huang, Lee-Feng Chien, y Yen-Jen Oyang. Relevant term suggestion in interactive web search based on contextual information in query session logs. *J. Am. Soc. Inf. Sci. Technol.*, 54(7):638–649, 2003. ISSN 1532-2882. 133

Jeff Huang y Efthimis N. Efthimiadis. Analyzing and evaluating query reformulation strategies in web search logs. En *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, páginas 77–86, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-512-3. 130

Xiangji Huang, Fuchun Peng, Aijun An, y Dale Schuurmans. Dynamic web log session identification with statistical language models. *Journal of the American Society for Information Science and Technology*, 55:1290–1303, 2004a. 129

Zan Huang, Hsinchun Chen, y Daniel Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1): 116–142, 2004b. ISSN 1046-8188. 19, 60

Christoph Hölscher y Gerhard Strube. Web search behavior of internet experts and newbies. *Computer Networks*, 33(1-6):337–346, 2000. 127

BIBLIOGRAFÍA

Anil K. Jain y Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988. ISBN 0-13-022278-X. 129

Bernard J. Jansen y Amanda Spink. An analysis of web documents retrieved and viewed. En *International Conference on Internet Computing*, páginas 65–69, 2003. 128

Bernard J. Jansen y Amanda Spink. An analysis of web searching by european alltheweb.com users. *Inf. Process. Manage.*, 41(2):361–381, 2005. 125

Bernard J. Jansen y Amanda Spink. How are we searching the world wide web? a comparison of nine search engine transaction logs. *Inf. Process. Manage.*, 42(1):248–263, 2006. 125

Bernard J. Jansen, Amanda Spink, Judy Bateman, y Tefko Saracevic. Real life information retrieval: a study of user queries on the web. *SIGIR Forum*, 32(1):5–17, 1998. ISSN 0163-5840. 128

Bernard J. Jansen, Amanda Spink, y Tefko Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. *Inf. Process. Manage.*, 36(2):207–227, Enero 2000. ISSN 0306-4573. 128

Bernard J. Jansen, Amanda Spink, y Jan O. Pedersen. A temporal comparison of altavista web searching. *JASIST*, 56(6):559–570, 2005. 126

Bernard J. Jansen, Danielle L. Booth, y Amanda Spink. Determining the user intent of web search engine queries. En *WWW '07: Proceedings of the 16th international conference on World Wide Web*, páginas 1149–1150, New York, NY, USA, 2007a. ACM. ISBN 978-1-59593-654-7. 127

Bernard J. Jansen, Amanda Spink, Chris Blakely, y Sherry Koshman. Defining a session on web search engines: Research articles. *Journal of the American Society for Information Science and Technology*, 58(6):862–871, 2007b. ISSN 1532-2882. 130, 131

Bernard J. Jansen, Amanda Spink, y Sherry Koshman. Web searcher interaction with the dogpile.com metasearch engine. *JASIST*, 58(5):744–755, 2007c. 125, 126

Bernard J. Jansen, Amanda Spink, y Bhuva Narayan. Query modifications patterns during web searching. *2013 10th International Conference on Information Technology: New Generations*, 0:439–444, 2007d. 130

Thorsten Joachims. Optimizing search engines using clickthrough data. En *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, páginas 133–142, New York, NY, USA, 2002. ACM Press. ISBN 158113567X. 124, 134

Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, y Geri Gay. Accurately interpreting clickthrough data as implicit feedback. En *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, páginas 154–161, New York, NY, USA, 2005. ACM. ISBN 1-59593-034-5. 128

Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, Filip Radlinski, y Geri Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Trans. Inf. Syst.*, 25(2):7, 2007. ISSN 1046-8188. 128, 134, 140, 161

Rosie Jones y Kristina Lisa Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. En *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, páginas 699–708, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-991-3. 129, 131, 132

Rosie Jones, Benjamin Rey, Omid Madani, y Wiley Greiner. Generating query substitutions. En *WWW '06: proceedings of the 15th international conference on world wide web*, páginas 387–396, New York, NY, USA, 2006. ACM Press. ISBN 1595933239. 134

William Jones, Harry Bruce, y Susan Dumais. Keeping found things found on the web. En *Proceedings of the tenth international conference on Information and knowledge management*, CIKM '01, páginas 119–126, New York, NY, USA, 2001. ACM. ISBN 1-58113-436-3. 127

William Jones, Susan Dumais, y Harry Bruce. Once found, what then? a study of " keeping" behaviors in the personal use of web information. *Proc. Am. Soc. Info. Sci. Tech.*, 39(1):391–402, Enero 2002. ISSN 1550-8390. 126, 127

In-Ho Kang y GilChang Kim. Query type classification for web document retrieval. En *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, páginas 64–71, New York, NY, USA, 2003. ACM. ISBN 1-58113-646-3. 127

George Karypis. Evaluation of item-based top-N recommendation algorithms. En *Proceedings of the tenth international conference on Information and knowledge management*, CIKM '01, páginas 247–254, New York, NY, USA, 2001. ACM. ISBN 1-58113-436-3. 34

R. Kohavi, B. Becker, y D. Sommerfield. Improving simple Bayes, 1997. 16

J. A. Konstan, J. Riedl, A. Borchers, y J. L. Herlocker. Recommender systems: A grouplens perspective. En *Recommender Systems. Papers from 1998 Workshop. Technical Report WS-98-08*, páginas 60–64. AAAI Press, 1998. 19

Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. En *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, páginas 426–434, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-193-4. 32, 39, 42

Yehuda Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Trans. Knowl. Discov. Data*, 4(1):1:1–1:24, Enero 2010a. ISSN 1556-4681. 26

Yehuda Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, 53:89–97, Abril 2010b. ISSN 0001-0782. 61, 105

Alexander Kotov, Paul N. Bennett, Ryen W. White, Susan T. Dumais, y Jaime Teevan. Modeling and analysis of cross-session search tasks. En *SIGIR*, páginas 5–14, 2011. 132

BIBLIOGRAFÍA

D. Kristol y L. Montulli. HTTP State Management Mechanism. RFC 2109 (Historic), Febrero 1997. Obsoleted by RFC 2965. 124

Karen Kukich. Techniques for automatically correcting words in text. *ACM Comput. Surv.*, 24(4):377–439, Diciembre 1992. ISSN 0360-0300. 133

Shyong K. Lam y John Riedl. Shilling recommender systems for fun and profit. En *Proceedings of the 13th international conference on World Wide Web*, WWW '04, páginas 393–402, New York, NY, USA, 2004. ACM. ISBN 1-58113-844-X. 20, 60

Shyong K. "Tony" Lam, Dan Frankowski, y John Riedl. Do you trust your recommendations? an exploration of security and privacy issues in recommender systems. En *Proceedings of the 2006 International Conference on Emerging Trends in Information and Communication Security*, ETRICS'06, páginas 14–29, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-34640-6, 978-3-540-34640-1. 21

Xuan Nhat Lam, Thuc Vu, Trong Duc Le, y Anh Duc Duong. Addressing cold-start problem in recommendation systems. En *Proceedings of the 2nd international conference on Ubiquitous information management and communication*, ICUIMC '08, páginas 208–211, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-993-7. 63

Neal Lathia, Stephen Hailes, Licia Capra, y Xavier Amatriain. Temporal diversity in recommender systems. En *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, páginas 210–217, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0153-4. 61, 105

Tessa Lau y Eric Horvitz. Patterns of search: analyzing and modeling web query refinement. En *UM '99: Proceedings of the seventh international conference on User modeling*, páginas 119–128, Secaucus, NJ, USA, 1999. Springer-Verlag New York, Inc. ISBN 3-211-83151-7. 130

Ard W. Lazonder, Harm J. A. Biemans, y Iwan G. J. H. Wopereis. Differences between novice and experienced users in searching information on the world wide web. *J. Am. Soc. Inf. Sci.*, 51(6):576–581, Abril 2000. ISSN 0002-8231. 127

Uichin Lee, Zhenyu Liu, y Junghoo Cho. Automatic identification of user goals in web search. En *WWW '05: Proceedings of the 14th international conference on World Wide Web*, páginas 391–400, New York, NY, USA, 2005. ACM. ISBN 1-59593-046-9. 125, 127

Daniel Lemire. Scale and translation invariant collaborative filtering systems. *Information Retrieval*, 8:1–22, 2003. 71

Daniel Lemire y Anna Maclachlan. Slope one predictors for online rating-based collaborative filtering. En *Proceedings of SIAM Data Mining*, SDM'05, 2005. 32, 37, 168

Cane Wing-ki Leung, Stephen Chi-fai Chan, y Fu-lai Chung. Applying cross-level association rule mining to cold-start recommendations. En *Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops*, WI-IATW '07, páginas 133–136, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-3028-1. 63

Mu Li, Yang Zhang, Muhua Zhu, y Ming Zhou. Exploring distributional similarity based models for query spelling correction. En *ACL-44: Proceedings of the 21st International*

Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, páginas 1025–1032, Morristown, NJ, USA, 2006. Association for Computational Linguistics. 133

Greg Linden, Brent Smith, y Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, Enero 2003. ISSN 1089-7801. 10, 31

Nathan N. Liu, Min Zhao, Evan Xiang, y Qiang Yang. Online evolutionary collaborative filtering. En *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, páginas 95–102, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-906-0. 61

Fabiana Lorenzi y Francesco Ricci. Case-based recommender systems: a unifying view. En *Proceedings of the 2003 international conference on Intelligent Techniques for Web Personalization*, ITWP'03, páginas 89–113, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3-540-29846-0, 978-3-540-29846-5. 17

Claudio Lucchese, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, y Gabriele Tolomei. Identifying task-based sessions in search engine query logs. En *WSDM*, páginas 277–286, 2011. 131, 132

Claudio Lucchese, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, y Gabriele Tolomei. Discovering tasks from search engine query logs. *ACM Trans. Inf. Syst.*, 31(3):14:1–14:43, Agosto 2013. ISSN 1046-8188. 132

J. B. MacQueen. Some methods for classification and analysis of multivariate observations. En L. M. Le Cam y J. Neyman, editores, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volumen 1, páginas 281–297. University of California Press, 1967. 39

Christopher D. Manning, Prabhakar Raghavan, y Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715. 24

Gary Marchionini. *Information seeking in electronic environments*. Cambridge University Press, New York, NY, USA, 1995. ISBN 0-521-44372-5. 126, 141, 153

Gary Marchionini. Exploratory search: from finding to understanding. *Commun. ACM*, 49 (4):41–46, 2006. 125, 140, 144

Evangelos Markatos. On caching search engine query results. *Computer Communications*, 24:137–143(7), 2000. 124

B. Marlin. Collaborative filtering: A machine learning perspective. Master's thesis, University of Toronto, 2004a. 42

Benjamin Marlin. Modeling user rating profiles for collaborative filtering. En Sebastian Thrun, Lawrence Saul, y Bernhard Schölkopf, editores, *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2004b. MIT Press. 42

Matthew R. McLaughlin y Jonathan L. Herlocker. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. En *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information*

retrieval, SIGIR '04, páginas 329–336, New York, NY, USA, 2004. ACM. ISBN 1-58113-881-4. 43

Sean M. McNee, John Riedl, y Joseph A. Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. En *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, páginas 1097–1101, New York, NY, USA, 2006. ACM. ISBN 1-59593-298-4. 22

Prem Melville, Raymod J. Mooney, y Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. En *Eighteenth national conference on Artificial intelligence*, páginas 187–192, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence. ISBN 0-262-51129-0. 62

Natasa Milic-Frayling, Rachel Jones, Kerry Rodden, Gavin Smyth, Alan F. Blackwell, y Ralph Sommerer. Smartback: supporting users in back navigation. En *WWW*, páginas 63–71, 2004. 126

Koji Miyahara y Michael J. Pazzani. Collaborative filtering with the simple bayesian classifier. En *Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence*, PRICAI'00, páginas 679–689, Berlin, Heidelberg, 2000. Springer-Verlag. ISBN 3-540-67925-1. 42

Bamshad Mobasher, Robin Burke, Runa Bhaumik, y Chad Williams. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Trans. Interet Technol.*, 7(4):23, 2007. ISSN 1533-5399. 20

Craig G. Murray, Jimmy Lin, y Abdur Chowdhury. Identification of user sessions with hierarchical agglomerative clustering. *Proceedings of the American Society for Information Science and Technology*, 43(1):1–5, 2006. 129

An-Te Nguyen, Nathalie Denos, y Catherine Berrut. Improving new user recommendations with rule-based induction on cold user data. En *Proceedings of the 2007 ACM conference on Recommender systems*, RecSys '07, páginas 121–128, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-730–8. 63

V. L. O'Day y R. Jeffries. Orienteering in an information landscape: How information seekers get from here to there. En *Proc. of INTERCHI-93*, páginas 438–445, Amsterdam, The Netherlands, 1993. 125, 126, 141

H. Cenk Ozmutlu y Fatih Çavdur. Application of automatic topic identification on excite web search engine data logs. *Inf. Process. Manage.*, 41(5):1243–1262, 2005. ISSN 0306-4573. 130, 131

Huseyin Cenk Ozmutlu, Fatih Çavdur, y Seda Ozmutlu. Automatic new topic identification in search engine transaction logs. *Internet Research*, 16(3):323–338, 2006. 130, 131

Manos Papagelis, Dimitris Plexousakis, y Themistoklis Kutsuras. Alleviating the sparsity problem of collaborative filtering using trust inferences. *Trust Management*, 3477: 224–239, 2005a. 63

Manos Papagelis, Ioannis Rousidis, Dimitris Plexousakis, y Elias Theoharopoulos. Incremental collaborative filtering for highly-scalable recommendation algorithms. En *Proceedings of the 15th international conference on Foundations of Intelligent Systems*, ISMIS'05, páginas 553–561, Berlin, Heidelberg, 2005b. Springer-Verlag. ISBN 3-540-25878-7, 978-3-540-25878-0. 21

Seung-Taek Park y Wei Chu. Pairwise preference regression for cold-start recommendation. En *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, páginas 21–28, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-435-5. 63

Seung-Taek Park, David Pennock, Omid Madani, Nathan Good, y Dennis DeCoste. Naïve filterbots for robust cold-start recommendations. En *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, páginas 699–705, New York, NY, USA, 2006. ACM. ISBN 1-59593-339-5. 62

Greg Pass, Abdur Chowdhury, y Cayley Torgeson. A picture of search. En *InfoScale '06: Proceedings of the 1st international conference on Scalable information systems*, página 1, New York, NY, USA, 2006. ACM. ISBN 1-59593-428-6. 161, 168

Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. En *Proceedings of KDD Cup and Workshop*, KDDCup '07, páginas 39–42, San Jose, California, USA, 2007. ACM. 32, 39

Emily S. Patterson, Emilie M. Roth, y David D. Woods. Predicting vulnerabilities in computer-supported inferential analysis under data overload. *Cognition, Technology & Work*, 3(4):224–237, 2001. 127, 144

Michael J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artif. Intell. Rev.*, 13(5-6):393–408, Diciembre 1999. ISSN 0269-2821. 62

David Pennock, Eric Horvitz, Steve Lawrence, y C. Lee Giles. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. En *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, UAI 2000*, páginas 473–480, Stanford, CA, 2000. 32, 42, 168

B. Piccart, J. Struyf, y H. Blockeel. Alleviating the sparsity problem in collaborative filtering by using an adapted distance and a graph-based method. En *SIAM International Conference on Data Mining*, SDM 2010, páginas 189–198, 2010. 63

Peter L. T. Pirolli. *Information Foraging Theory: Adaptive Interaction with Information.* Oxford University Press, Inc., New York, NY, USA, 1 edition, 2007. ISBN 0195173325, 9780195173321. 126

Annabel Pollock y Andrew Hockley. What's wrong with internet searching. *D-Lib Magazine*, 3(3), 1997. 126

Mari Carmen Puerta Melguizo, Lou Boves, Anita Deshpande, y Olga Muñoz Ramos. A proactive recommendation system for writing: helping without disrupting. En Willem-Paul Brinkman, Dong-Han Ham, y B. L. William Wong, editores, *ECCE*, volumen 250 de *ACM International Conference Proceeding Series*, páginas 89–95. ACM, 2007. ISBN 978-1-84799-849-1. 11

BIBLIOGRAFÍA

Diego Puppin, Fabrizio Silvestri, y Domenico Laforenza. Query-driven document partitioning and collection selection. En *InfoScale '06: Proceedings of the 1st international conference on Scalable information systems*, página 34, New York, NY, USA, 2006. ACM. ISBN 1-59593-428-6. 124

Filip Radlinski y Thorsten Joachims. Query chains: learning to rank from implicit feedback. En *Kdd '05: proceedings of the eleventh acm sigkdd international conference on knowledge discovery in data mining*, páginas 239–248, New York, NY, USA, 2005. ACM Press. ISBN 159593135X. 129, 131, 137

Vijay V. Raghavan y Hayri Sever. On the reuse of past optimal queries. En *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, páginas 344–350, New York, NY, USA, 1995. ACM. ISBN 0-89791-714-6. 133

Naren Ramakrishnan, Benjamin J. Keller, Batul J. Mirza, Ananth Grama, y George Karypis. When being weak is brave: Privacy in recommender systems. *CoRR*, cs.CG/0105028, 2001. 21

Al Mamunur Rashid, George Karypis, y John Riedl. Learning preferences of new users in recommender systems: an information theoretic approach. *SIGKDD Explor. Newsl.*, 10: 90–100, Diciembre 2008. ISSN 1931-0145. 63

Paul Resnick y Hal R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, Marzo 1997. ISSN 0001-0782. 10

Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, y John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. En *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, CSCW '94, páginas 175–186, New York, NY, USA, 1994. ACM. ISBN 0-89791-689-1. 10, 14, 15, 31, 32, 33, 35, 60, 78, 167

Francesco Ricci, Lior Rokach, Bracha Shapira, y Paul B. Kantor, editores. *Recommender Systems Handbook*. Springer, 2011. ISBN 978-0-387-85819-7. 15, 196

E. Rich. User modeling via stereotypes. En *Cognitive Science: A Multidisciplinary Journal*, volumen Vol. 3, No. 4, páginas 329–354, 1979. 16

J. Rocchio. Relevance feedback in information retrieval. En Gerard Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, páginas 313–323. Prentice-Hall, 1971. 134

L. Rokach y S. Kisilevich. Initial profile generation in recommender systems using pairwise comparison. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 42(6):1854–1859, nov. 2012. 63

Daniel E. Rose y Danny Levinson. Understanding user goals in web search. En *Proceedings of the 13th international conference on World Wide Web*, WWW '04, páginas 13–19, New York, NY, USA, 2004. ACM. ISBN 1-58113-844-X. 127

Daniel M. Russell, Mark J. Stefik, Peter Pirolli, y Stuart K. Card. The cost structure of sensemaking. En Bert Arnold, Gerrit van der Veer, y Ted White, editores, *CHI '93: Proceedings of the SIGCHI conference on Human factors in computing systems*, páginas 269–276, New York, NY, USA, 1993. ACM Press. 126

Ian Ruthven y Mounia Lalmas. A survey on the use of relevance feedback for information access systems. *Knowl. Eng. Rev.*, 18(2):95–145, Junio 2003. ISSN 0269-8889. 128

Mehran Sahami y Timothy D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. En *Proceedings of the 15th international conference on World Wide Web*, WWW '06, páginas 377–386, New York, NY, USA, 2006. ACM. ISBN 1-59593-323-9. 133

J. J. Sandvig, Bamshad Mobasher, y Robin Burke. Robustness of collaborative recommendation based on association rule mining. En *Proceedings of the 2007 ACM conference on Recommender systems*, RecSys '07, páginas 105–112, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-730–8. 20

Badrul Sarwar, George Karypis, Joseph Konstan, y John Riedl. Analysis of recommendation algorithms for e-commerce. En *Proceedings of the 2nd ACM conference on Electronic commerce*, EC '00, páginas 158–167, New York, NY, USA, 2000a. ACM. ISBN 1-58113-272-7. 42

Badrul Sarwar, George Karypis, Joseph Konstan, y John Riedl. Item-based collaborative filtering recommendation algorithms. En *Proceedings of the 10th international conference on World Wide Web*, WWW '01, páginas 285–295, New York, NY, USA, 2001. ACM. ISBN 1-58113-348-0. 26, 32, 34, 60, 85, 167

Badrul Sarwar, George Karypis, Joseph Konstan, y John Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. En *Fifth International Conference on Computer and Information Science*, páginas 27–28, 2002. 21

Badrul M. Sarwar, Joseph A. Konstan, Al Borchers, Jon Herlocker, Brad Miller, y John Riedl. Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. En *Computer Supported Cooperative Work*, páginas 345–354, 1998. 26

Badrul M. Sarwar, George Karypis, Joseph A. Konstan, y John T. Riedl. Application of dimensionality reduction in recommender systems – a case study. En *ACM WebKDD Workshop*, 2000b. 38

Badrul M. Sarwar, Joseph A. Konstan, y John Riedl. Distributed recommender systems for internet commerce. En *Encyclopedia of Information Science and Technology (II)*, páginas 907–911. 2005. 21

A. Schein, A. Popescul, L. Ungar, y D. Pennock. Generative models for cold-start recommendations. En *Proceedings of the 2001 SIGIR Workshop on Recommender Systems*, 2001. 26

Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, y David M. Pennock. Methods and metrics for cold-start recommendations. En *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, páginas 253–260, New York, NY, USA, 2002. ACM. ISBN 1-58113-561-0. 19, 60, 62, 82, 177

Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, 2002. ISSN 0360-0300. 16

BIBLIOGRAFÍA

Mahdi Shafiei y Evangelos Milios. Model-based Overlapping Co-Clustering. En *Proceedings of the Fourth Workshop on Text Mining, Sixth SIAM International Conference on Data Mining*, Bethesda, Maryland, Abril 2006. 40

Upendra Shardanand. Social information filtering for music recommendation. Master's thesis, Massachussets Institute of Technology, Septiembre 1994. 10, 32, 34, 60, 61

Upendra Shardanand y Pattie Maes. Social information filtering: algorithms for automating "word of mouth". En *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '95, páginas 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-84705-1. 33, 35, 167

Xuehua Shen, Bin Tan, y ChengXiang Zhai. Implicit user modeling for personalized search. En *Proceedings of the 14th ACM international conference on Information and knowledge management*, CIKM '05, páginas 824–831, New York, NY, USA, 2005. ACM. ISBN 1-59593-140-6. 131

Xiaodong Shi y Christopher C. Yang. Mining related queries from search engine query logs. En *WWW '06: Proceedings of the 15th international conference on World Wide Web*, páginas 943–944, New York, NY, USA, 2006. ACM. ISBN 1-59593-323-9. 131

Milad Shokouhi, Justin Zobel, Saied Tahaghoghi, y Falk Scholer. Using query logs to establish vocabularies in distributed information retrieval. *Inf. Process. Manage.*, 43(1): 169–180, 2007. ISSN 0306-4573. 124

Reza Shokri, Pedram Pedarsani, George Theodorakopoulos, y Jean-Pierre Hubaux. Preserving privacy in collaborative filtering through distributed aggregation of offline profiles. En *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, páginas 157–164, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-435-5. 21

Craig Silverstein, Hannes Marais, Monika Henzinger, y Michael Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999. ISSN 0163-5840. 125, 128, 129

Fabrizio Silvestri. Mining query logs: Turning search usage data into knowledge. *Found. Trends Inf. Retr.*, 4(1—2):1–174, Enero 2010. ISSN 1554-0669. 124

Rashmi Sinha, , Rashmi Sinha, y Kirsten Swearingen. Comparing recommendations made by online systems and friends. En *In Proceedings of the DELOS-NSF Workshop on Personalization and Recommender Systems in Digital Libraries*, 2001. 17

Gleb Skobeltsyn, Flavio Junqueira, Vassilis Plachouras, y Ricardo A. Baeza-Yates. Resin: a combination of results caching and index pruning for high-performance web search engines. En *SIGIR*, páginas 131–138, 2008. 125

B. Smyth y P. McClave. Similarity vs. diversity. En D. W. Aha y I. Watson, editores, *Proceedings of the 4th International Conference on Case-Based Reasoning*, páginas 347–361, Vancouver, 2001. Springer-Verlag. 21

Barry Smyth. A community-based approach to personalizing web search. *Computer*, 40(8): 42–50, 2007. ISSN 0018-9162. 134, 138

Barry Smyth, Evelyn Balfe, Oisin Boydell, Keith Bradley, Peter Briggs, Maurice Coyle, y Jill Freyne. A live-user evaluation of collaborative web search. En *In IJCAI*, páginas 1419–1424, 2005. 134

E. Isaac Sparling y Shilad Sen. Rating: how difficult is it? En *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, páginas 149–156, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0683-6. 14

Sarah Spiekermann y Corina Paraschiv. Motivating human-agent interaction: Transferring insights from behavioral marketing to interface design. *Electronic Commerce Research*, 2(3):255–285, 2002. 17

Amanda Spink, Howard Greisdorf, y Judy Bateman. From highly relevant to not relevant: Examining different regions of relevance. *Inf. Process. Manage.*, 34(5):599–621, 1998a. 128

Amanda Spink, Thomas D. Wilson, David Ellis, y Nigel Ford. Modeling users' successive searches in digital environments: A national science foundation/british library funded study. *D-Lib Magazine*, 4(4), 1998b. 128

Amanda Spink, Bernard J. Jansen, y Cenk H. Ozmutlu. Use of query reformulation and relevance feedback by excite users. *Internet Research: Electronic Networking Applications and Policy*, 10(4):317–328, 2000. 130

Amanda Spink, Dietmar Wolfram, Major B. J. Jansen, y Tefko Saracevic. Searching the web: the public and their queries. *J. Am. Soc. Inf. Sci. Technol.*, 52(3):226–234, Febrero 2001. ISSN 1532-2882. 125

Amanda Spink, Bernard J. Jansen, Dietmar Wolfram, y Tefko Saracevic. From e-sex to e-commerce: Web search changes. *Computer*, 35(3):107–109, 2002. ISSN 0018-9162. 125

Xiaoyuan Su y Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:2–2, 2009. ISSN 1687-7470. 32

Alistair G. Sutcliffe y Mark Ennis. Towards a cognitive theory of information retrieval. *Interacting with Computers*, 10(3):321–351, 1998. 126, 128

Diana Tabatabai y Bruce M. Shore. How experts and novices search the Web. *Library & Information Science Research*, 27:222–248, 2005. 127, 128, 144

Jaime Teevan, Christine Alvarado, Mark S. Ackerman, y David R. Karger. The perfect search engine is not enough: a study of orienteering behavior in directed search. En *CHI '04: Proc. of the SIGCHI conf. on Human factors in computing systems*, páginas 415–422. ACM Press, 2004. 126, 127

Pertti Vakkari. Relevance and contributing information types of searched documents in task performance. En *SIGIR*, páginas 2–9, 2000. 128

C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 2nd edition, 1979. 24

Slobodan Vucetic y Zoran Obradovic. A regression-based approach for scaling-up personalized recommender systems in e-commerce. En *ACM WebKDD Workshop*, 2000. 37

Jun Wang. *Relevance Models for Collaborative Filtering.* PhD thesis, Delft University of Technology, Delft, The Netherlands, Abril 2008. 61

Jun Wang, Arjen P. de Vries, y Marcel J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. En *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, páginas 501–508, New York, NY, USA, 2006. ACM. ISBN 1-59593-369-7. 36

Yuanyuan Wang, Stephen Chi-Fai Chan, y Grace Ngai. Applicability of demographic recommender system to tourist attractions: A case study on trip advisor. En *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology - Volume 03*, WI-IAT '12, páginas 97–101, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-0-7695-4880-7. 16

Ji-Rong Wen y HongJiang Zhang. Query clustering in the web context. En Weili Wu, Hui Xiong, y Shashi Shekhar, editores, *Clustering and Information Retrieval*, páginas 195–226. Kluwer, 2003. ISBN 1-4020-7682-7. 128

Ji-Rong Wen, Jian-Yun Nie, y Hong-Jiang Zhang. Clustering user queries of a search engine. En *WWW '01: Proceedings of the 10th international conference on World Wide Web*, páginas 162–168, New York, NY, USA, 2001. ACM Press. ISBN 1581133480. 134

Ryen W. White y Gary Marchionini. Examining the effectiveness of real-time query expansion. *Inf. Process. Manage.*, 43(3):685–704, 2007. 132

Ryen W. White, Ian Ruthven, y Joemon M. Jose. A study of factors affecting the utility of implicit relevance feedback. En *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, páginas 35–42, New York, NY, USA, 2005. ACM Press. ISBN 1595930345. 127, 128

Ryen W. White, Mikhail Bilenko, y Silviu Cucerzan. Leveraging popular destinations to enhance web search interaction. *ACM Trans. Web*, 2(3):1–30, 2008. ISSN 1559-1131. 162

Dietmar Wolfram. A query-level examination of end user searching behaviour on the excite search engine. En *Proceedings of the 28th Anual Conference Canadian Association for Information Science*, 2000. 125, 128

Liang Xiang y Qing Yang. Time-dependent models in collaborative filtering based recommender system. En *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '09, páginas 450–457, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3801-3. 15

Li Xiong y Eugene Agichtein. Towards privacy-preserving query log publishing. En Einat Amitay, Craig G. Murray, y Jaime Teevan, editores, *Query Log Analysis: Social And Technological Challenges. A workshop at the 16th International World Wide Web Conference (WWW 2007)*, Mayo 2007. 128

Gui-Rong Xue, Chenxi Lin, Qiang Yang, WenSi Xi, Hua-Jun Zeng, Yong Yu, y Zheng Chen. Scalable collaborative filtering using cluster-based smoothing. En *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, páginas 114–121, New York, NY, USA, 2005. ACM. ISBN 1-59593-034-5. 40

Kai Yu, Anton Schwaighofer, Volker Tresp, Xiaowei Xu, y Hans-Peter Kriegel. Probabilistic memory-based collaborative filtering. *Trans. on Knowledge and Data Engineering*, 16(1):56–69, 2004. ISSN 1041-4347. 42

Osmar R. Zaïane y Alexander Strilets. Finding similar queries to satisfy searches based on query traces. En *OOIS '02: Proceedings of the Workshops on Advances in Object-Oriented Information Systems*, páginas 207–216, London, UK, 2002. Springer-Verlag. ISBN 3-540-44088-7. 134

Mi Zhang. Enhancing diversity in top-n recommendation. En Lawrence D. Bergman, Alexander Tuzhilin, Robin D. Burke, Alexander Felfernig, y Lars Schmidt-Thieme, editores, *RecSys*, páginas 397–400. ACM, 2009. ISBN 978-1-60558-435-5. 21

Tian Zhang, Raghu Ramakrishnan, y Miron Livny. BIRCH: an efficient data clustering method for very large databases. En *SIGMOD*, páginas 103–114, 1996. 39

Zhiyong Zhang y Olfa Nasraoui. Mining search engine query logs for query recommendation. En *WWW '06: Proceedings of the 15th international conference on World Wide Web*, páginas 1039–1040, New York, NY, USA, 2006. ACM. ISBN 1-59593-323-9. 133

Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, y Georg Lausen. Improving recommendation lists through topic diversification. En *Proceedings of the 14th international conference on World Wide Web*, WWW '05, páginas 22–32, New York, NY, USA, 2005. ACM. ISBN 1-59593-046-9. 172

G. K. Zipf. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Addison-Wesley, 1949. 152