



Facultad de Ciencias Exactas
Universidad Nacional del Centro de la
Provincia de Buenos Aires (UNICEN)

“Clasificación de Documentos Web utilizando Marcadores Sociales”

Tesis de Grado - Ingeniería de Sistemas

Nicolás Andrés Tourné

Directora:

Dra. Daniela Godoy

Tandil, Buenos Aires (Argentina)
Junio, 2011

Prólogo

En los últimos 15 años hemos tenido el privilegio de asistir a un fenómeno excepcional, el nacimiento de un nuevo medio de comunicación: Internet. Con tan sólo pensar un momento, se puede apreciar que tan importante e inusual ha sido este acontecimiento. Y es que la historia de la comunicación muestra que han de pasar décadas, o hasta siglos enteros, para presentarse tal suceso. Los diarios nacieron a principios del siglo XVIII, los primeros experimentos de emisiones radiofónicas datan de finales del XIX, la televisión surgió no hace más de 80 años. Nuestra generación ha podido ver germinar un nuevo medio y modo de comunicación social que, al día hoy, no quedan dudas que está llamado a convertirse en hegemónico en este siglo XXI.

Todo empezó cuando la Unión Soviética lanzó el *Sputnik*¹, el primer satélite artificial. En respuesta a este hecho, Estados Unidos creó el *ARPA* (Organismo de Proyectos de Investigación Avanzada) dentro del Ministerio de Defensa a fin de establecer su liderazgo en las áreas de ciencia y tecnología aplicadas a las fuerzas armadas. Este fue el verdadero comienzo de lo que hoy conocemos como Internet.

A partir de la última década del siglo pasado el crecimiento ha sido exponencial en cuanto a páginas webs. Desde tan solo un puñado de páginas activas a fines de 1990, hasta alcanzar las más de 11.500 millones de páginas disponibles en la red... y contando. El aumento de nodos, ordenadores, dominios y webs es imparable e impactante, pero aquí es dónde surge el interrogante: ¿qué tan simple es acceder a esta información? ¿es posible darle un orden para lograr ser rastreada en forma eficiente? ¿es factible la creación de un índice como si de una gran enciclopedia se tratara? Una de las mejores respuestas a estas cuestiones ha sido la construcción de directorios de páginas webs.

Actualmente, el directorio más conocido y confiable de la red² es administrado en forma manual, lo que supone una importante sobrecarga no sólo por el abundante volumen de información que maneja, sino también por el dinamismo y periodicidad en que es actualizada. Se ha experimentado con varias técnicas para automatizar esta labor, logrando distintos tipos de resultados. El actual trabajo presenta una nueva metodología para clasificar estas páginas webs mediante la utilización de un novedoso recurso: los marcadores sociales.

¹ El 4 de octubre de 1957.

² Se refiere al Open Directory Project, también conocido como DMoz (por Directory Mozilla).

Agradecimientos

Mucha gente ha estado presente en estos años de estudio, brindándome su apoyo de manera incondicional durante toda la carrera. De cada uno de ellos me guardo un recuerdo, el que más valoro, el que me sirve para dar cuenta de lo afortunado que he sido por habérmelos cruzado en el camino.

Simplemente me gustaría agradecer a mi familia, que ha tolerado mis días y noches de “ausencia” ante cada examen, disfrutando tanto de mis alegrías y logros como así también bancando mis frustraciones, pero siempre ofreciéndome su confianza a cada momento. Un GRACIAS de corazón a mis viejos por ilusionarse junto a mí en este desafío; a Mariano, por ser mi compañero de la vida; y a Leandrito, porque sé que desde algún lugar me iluminas día a día.

Contenido

Capítulo 1: Introducción	1
1.1. Marco teórico.....	1
1.2. Contexto.....	4
1.3. Propuesta	6
1.4. Alcance	6
1.5. Organización de la tesis	7
Capítulo 2: Marcadores sociales en la Web	8
2.1. Las etiquetas (o tags)	8
2.2. Tagging colaborativo	9
2.3. ¿Qué son los “marcadores sociales”?	13
2.4. Clasificación de páginas web empleando tags.....	17
Capítulo 3: Recursos utilizados.....	21
3.1. Colección de datos CABS120k08.....	21
3.2. Weka	32
3.3. Parser (CABS120k08 a formato ARFF).....	43
Capítulo 4: Desarrollo de la investigación	49
4.1. Resumen	49
4.2. Generación de los datasets.....	49
4.3. Pre-procesamiento de cada dataset en Weka	50
4.4. Clasificación y Análisis de resultados.....	53
Capítulo 5: Optimizaciones.....	62
5.1. Objetivo.....	62
5.2. Sin aplicar stemming	62
5.3. Aplicación de sinónimos (WordNet)	65
5.4. Aplicando spell check.....	68
5.5. Aplicando spell check mejorado	76
Conclusiones y discusión	81
Bibliografía	85
Links de interés.....	91
Anexo A: Glosario	93
Traducciones	93
Anexo B: Morfología lingüística en IR	98
Anexo C: Indicadores de evaluación de clasificadores.....	101
Anexo D: WordNet	105

Indice de Figuras

Figura 1.1. Etapa de entrenamiento de un clasificador.	3
Figura 1.2. Etapa de clasificación.	3
Figura 2.1. Representación gráfica de una folcsonomía.	10
Figura 2.2. Representación gráfica de una folcsonomía broad.....	11
Figura 2.3. Representación gráfica de una folcsonomía narrow.....	12
Figura 2.4. Captura de pantalla para agregar una nueva página en Delicious.....	14
Figura 2.5. Navegación de los documentos marcados en Delicious partir de los tags.	15
Figura 2.6. Búsqueda de páginas web almacenadas en Delicious basándose en tags.....	15
Figura 3.1. Triunvirato de metadatos.....	21
Figura 3.2. Longitud promedio de los marcadores sociales, search queries y anchor texts.....	25
Figura 3.3. Porcentajes de nuevos datos provistos por los tags de los documentos, palabras de los anchor text y términos de búsqueda.	27
Figura 3.4. Ejemplo de colección de datos CABS120k08.....	31
Figura 3.5. Captura de pantalla de la interface Explorer de Weka.	34
Figura 3.6. Estructura de un archivo ARFF.	36
Figura 3.7. Red bayesiana del ejemplo de la planta nuclear.....	38
Figura 3.8. Conjunto de muestras separadas en distintos grupos o categorías (H_1 , H_2 y H_3) a partir de aplicar SVM.	39
Figura 3.9. Resultado entregado por Weka luego de finalizada una clasificación.....	41
Figura 3.10. Vista de módulos del Parser.....	43
Figura 3.11. Encabezado de un dataset que contiene los campos query, anchortext y tags.	44
Figura 3.12. Datos de cada documento en un dataset con los campos query, anchortext y tags.....	45
Figura 3.13. Arquitectura del módulo Write ARFF Data.	45
Figura 3.14. Filtrado aplicado al contenido de un documento.	47
Figura 4.1. Dataset ARFF original, antes de realizar cualquier procesamiento en Weka.	50
Figura 4.2. Dataset ARFF obtenido luego de realizar un procesamiento en su contenido a partir de los filtros que proveee Weka.....	52
Figura 5.1. Filtrado aplicado al contenido de un documento, sin incluir Stemming.....	63
Figura 5.2. Filtrado aplicado al contenido de un documento, incluyendo la generación de sinónimos.	66
Figura 5.3. Filtrado aplicado al contenido de un documento, incluyendo un spell checker.....	69
Figura 5.4. Filtrado aplicado al contenido de un documento, incluyendo un spell checker mejorado... ..	77
Figura B.1. Algoritmo de Porter.....	99
Figura C.1. Representación gráfica de Precision y Recall.....	102

Indice de Tablas

Tabla 3.1. Estadísticas generales de CABS120k08.....	24
Tabla 3.2. Probabilidades estimadas.....	24
Tabla 3.3. Similitud de pares de tags (T), anchor texts (T), search queries (S) y categorías (S).	28
Tabla 3.4. Formato de datos de los documentos en CABS120k08.....	29
Tabla 3.5. Formato de datos de las categorías en CABS120k08.	29
Tabla 3.6. Formato de datos de las <i>search queries</i> en CABS120k08.....	29
Tabla 3.7. Formato de datos de los <i>incoming anchor texts</i> en CABS120k08.	30
Tabla 3.8. Formato de datos de los top tags de Delicious en CABS120k08.	30
Tabla 3.9. Formato de datos de los usuarios de Delicious en CABS120k08.....	30
Tabla 4.1. Resultados de aplicar el clasificador NaiveBayes con los valores por defecto de percentage splits y cross-validation.	53
Tabla 4.2. Indicadores obtenidos a partir del clasificador Naive Bayes con valores por defecto de percentage splits y cross-validation.	54
Tabla 4.3. Resultados de aplicar el clasificador SMO (PolyKernel) con los valores por defecto de percentage splits y cross-validation.	56
Tabla 4.4. Indicadores obtenidos a partir del clasificador SMO (PolyKernel) con valores por defecto de percentage splits y cross-validation.	57
Tabla 4.5. Resultados de aplicar el clasificador SMO (RBFKernel) con los valores por defecto de percentage splits y cross-validation.	57
Tabla 4.6. Indicadores obtenidos a partir del clasificador SMO (RBFKernel) con valores por defecto de percentage splits y cross-validation.	58
Tabla 4.7. Resultados de aplicar el clasificador SMO a distintos porcentajes de documentos de entrenamiento para Percentage splits.....	59
Tabla 5.1. Resultados de aplicar el clasificador SMO a distintos tamaños del conjunto de entrenamiento para el dataset anchortext+tags.	62
Tabla 5.2. Resultados de aplicar SMO a distintos tamaños del conjunto de entrenamiento para el dataset anchortext+tags sin aplicar stemming.	63
Tabla 5.3. Comparación de distintos términos antes y después de aplicar stemming.	64
Tabla 5.4. Indicadores obtenidos a partir del clasificador SMO con valores por defecto de percentage splits para los dataset baseline y no stemming.	65
Tabla 5.5. Resultados de aplicar SMO a distintos tamaños del conjunto de entrenamiento para el dataset anchortext+tags con sinónimos.	67
Tabla 5.6. Indicadores obtenidos a partir del clasificador SMO con valores por defecto de percentage splits para los dataset baseline y sinónimos.	68
Tabla 5.7. Resultados de aplicar el clasificador SMO a distintos tamaños del conjunto de entrenamiento para el dataset anchortext+tags aplicando spell check de Tumba!.....	70
Tabla 5.8. Indicadores obtenidos a partir del clasificador SMO con valores por defecto de percentage splits para los dataset baseline y spell check Tumba.	71
Tabla 5.9. Resultados de aplicar el clasificador SMO a distintos tamaños del conjunto de entrenamiento para el dataset anchortext+tags aplicando spell check de JaSpell.	71
Tabla 5.10. Indicadores obtenidos a partir del clasificador SMO con valores por defecto de percentage splits para los dataset baseline y spell check JaSpell.	72
Tabla 5.11. Resultados de aplicar el clasificador SMO a distintos tamaños del conjunto de entrenamiento para el dataset anchortext+tags aplicando spell check de Hunspell.....	73
Tabla 5.12. Indicadores obtenidos a partir del clasificador SMO con valores por defecto de percentage splits para los dataset baseline y spell check Hunspell.	74
Tabla 5.13. Comparación de los resultados obtenidos al aplicar el clasificador SMO a distintos tamaños del conjunto de entrenamiento para los tres spell-checkers.	75
Tabla 5.14. Resultados de aplicar el clasificador SMO a distintos tamaños del conjunto de entrenamiento para el dataset anchortext+tags aplicando spell check mejorado.	78
Tabla 5.15. Indicadores obtenidos a partir del clasificador SMO con valores por defecto de percentage splits para los dataset baseline, spell check JaSpell y spell check mejorado.	79

Indice de Gráficos

Gráfico 4.1. Representación de los resultados de clasificación utilizando NaiveBayes para 66% de instancias de entrenamiento y 10 folds, respectivamente.	54
Gráfico 4.2. Representación de los resultados de clasificación utilizando SMO (PolyKernel) para 66% de instancias de entrenamiento y 10 folds, respectivamente.	56
Gráfico 4.3. Representación de los resultados de clasificación utilizando SMO (RBFKernel) para 66% de instancias de entrenamiento y 10 folds, respectivamente.	58
Gráfico 4.4. Representación de los resultados de clasificación utilizando SMO y distintos % de instancias de entrenamiento.	60
Gráfico 5.1. Representación gráfica de los resultados del clasificador SMO con distintos tamaños del conjunto de entrenamiento para anchortext+tags sin aplicar stemming.	64
Gráfico 5.2. Representación gráfica de los resultados del clasificador SMO con distintas combinaciones de Cross-validation para anchortext+tags con sinónimos.	67
Gráfico 5.3. Representación gráfica de los resultados del clasificador SMO con distintas combinaciones de Cross-validation para anchortext+tags aplicando el spell check de Tumba!	70
Gráfico 5.4. Representación gráfica de los resultados del clasificador SMO con distintas combinaciones de Cross-validation para anchortext+tags aplicando el spell check de JaSpell.....	72
Gráfico 5.5. Representación gráfica de los resultados del clasificador SMO con distintas combinaciones de Cross-validation para anchortext+tags aplicando el spell check de Hunspell.	73
Gráfico 5.6. Representación gráfica de los resultados del clasificador SMO con distintas combinaciones de Cross-validation para anchortext+tags para los tres spell-checkers.....	75
Gráfico 5.7. Representación gráfica de los resultados del clasificador SMO con distintas combinaciones de Cross-validation para anchortext+tags aplicando el spell check JaSpell y el mejorado.	79

Capítulo 1: Introducción

1.1. Marco teórico

Recientemente, la minería de datos (*data mining* de ahora en más) ha sido objeto de numerosos artículos en revistas de negocios y software. Sin embargo, hace solo unos años, pocas personas habían oído hablar del término. Aunque el *data mining* es la evolución de un campo con una larga historia, el término en sí sólo se introdujo hace relativamente poco tiempo, en el decenio de 1990.

El *data mining* remonta sus raíces a lo largo de una familia con tres líneas. La más larga de estas líneas son las *estadísticas clásicas*. Sin estadísticas, no habría *data mining*, las estadísticas son la base de la mayoría de las tecnologías en las que la *data mining* es construida. Las estadísticas clásicas abarcan conceptos tales como análisis de regresión, distribución estándar, desviación estándar, varianza estándar, análisis discriminante, análisis de clusters, y los intervalos de confianza, los cuales se utilizan para estudiar los datos y sus relaciones. Estos son los estudios más importantes y avanzados que se basan en análisis estadísticos. Ciertamente, las estadísticas clásicas juegan un rol significativo al ubicarse en el centro de las herramientas y técnicas de *data mining*.

La segunda familia más extensa de *data mining* es la *inteligencia artificial* (del inglés *artificial intelligence*, *AI*). Esta disciplina, que se basa en heurística en contraposición a las estadísticas, intenta aplicar el razonamiento humano como procesamiento a los problemas estadísticos. Debido a que este enfoque requiere gran potencia de procesamiento por parte de las computadoras, no fue práctico hasta principios de los 80, cuando las computadoras comenzaron a ofrecer un procesamiento poderoso y útil a precios razonables. *AI* encontró un lugar dentro de un puñado de aplicaciones de alto nivel en los mercados científicos/gubernamentales, pero los altos precios de las supercomputadores de la época establecían una barrera importante para aplicarse en cualquier otro ámbito. A excepción de esto, se ha utilizado en algunos productos comerciales de alta gama, tales como la optimización de los módulos de consultas (*query*) para los sistemas de gestión de bases de datos relacionales (RDBMS).

La tercer línea de la familia de *data mining* fue *machine learning*, que se describe con mayor precisión como la unión de las estadísticas y la *AI*. Aunque la *AI* no fue un éxito comercial, sus técnicas eran en gran parte usadas por *machine learning*, que fue capaz de tomar ventaja de la mejora en la relación precio/rendimiento que ofrecían las computadoras de los años 80 y 90, logrando crear una mayor cantidad de aplicaciones ya que el precio era inferior que con *AI*. *Machine learning* puede considerarse como la evolución de la *AI*, ya que mezcla las heurísticas de *AI* con análisis estadísticos avanzados. Además, intenta permitirle a los programas que aprendan sobre los datos que ellos estudian, ya sea tomando decisiones basadas en la calidad de los mismos, usando estadísticas para fundamentar conceptos, o agregando heurísticas y algoritmos de *AI* más avanzados para conseguir sus objetivos.

Data mining, en muchos sentidos, es fundamentalmente la adaptación de las técnicas de *machine learning* a las aplicaciones comerciales. *Data mining* es mejor descrita como la unión de desarrollos estadísticos históricos y recientes, *AI* y *machine learning*. Estas estadísticas son luego utilizadas conjuntamente para estudiar los datos y localizar tendencias o patrones ocultos en ellos. También, ha encontrado una gran aceptación en las áreas de ciencias y negocios, donde existe la necesidad de analizar grandes volúmenes de datos para descubrir tendencias que de otra manera serían imposibles de hallar.

Con el surgimiento y rotundo crecimiento de la web, el *data mining* encontró un campo muy interesante donde actuar. Se empezó entonces a hablar de *web mining*, comúnmente usada para el estudio de varios aspectos esenciales de un sitio, ayudando a descubrir tendencias y relaciones en el comportamiento de los usuarios que sirven como pistas para, por ejemplo, mejorar la usabilidad del mismo.

Cuando un sitio es navegado por los usuarios, los logs de los servidores que lo alojan almacenan información acerca de cada visita. Luego, estos logs pueden ser procesados por programas de estadísticas como Awstats³, Webtrends⁴ o Google Analytics⁵ (para más información ver *Links de interés*), que devuelven información estructurada y significativa como patrones de navegación, comportamiento de los usuarios ante cierta indexación de contenidos o estructuras de texto, preferencias del usuario, inconsistencias, etc. El *web mining*, para mejorar su efectividad, se subdivide en áreas que abarcan el contenido del sitio, la estructura de navegación y el comportamiento de los usuarios ante los dos primeros: *Web Content Mining*⁶, *Web Structure Mining*⁷ y *Web Usage Mining*⁸.

Las técnicas de *data mining* más utilizadas en *web mining* son la clasificación y el clustering. La primera consiste de un procedimiento en el cual un conjunto de ítems es ubicado en grupos o clases – previamente identificados– tomando como referencia información cuantitativa proveniente de una o más características de los mismos –conocidas como atributos– y basados en un conjunto de entrenamiento de ítems anteriormente clasificados. Mientras que la segunda se basa en un procedimiento de agrupación de una serie de vectores de acuerdo con un criterio de cercanía.

Por lo tanto, un clasificador utiliza los conceptos aprendidos en la *etapa de entrenamiento* para clasificar nuevos ejemplos [1]. La figura 1.1 muestra cómo se construye un clasificador.

³ Herramienta open source de informes de análisis web. <http://awstats.sourceforge.net>

⁴ Herramienta de análisis web, orientado a social media. <http://www.webtrends.com>

⁵ Servicio gratuito de estadísticas de sitios web, propiedad de Google. <http://www.google.com/analytics>

⁶ Minería de contenido web. Se centra en el contenido.

⁷ Minería de estructura web. Analiza la estructura del sitio y si los usuarios encuentran la información que buscan.

⁸ Minería de uso de web. Se refiere a patrones de navegación que se descubren de los usuarios.

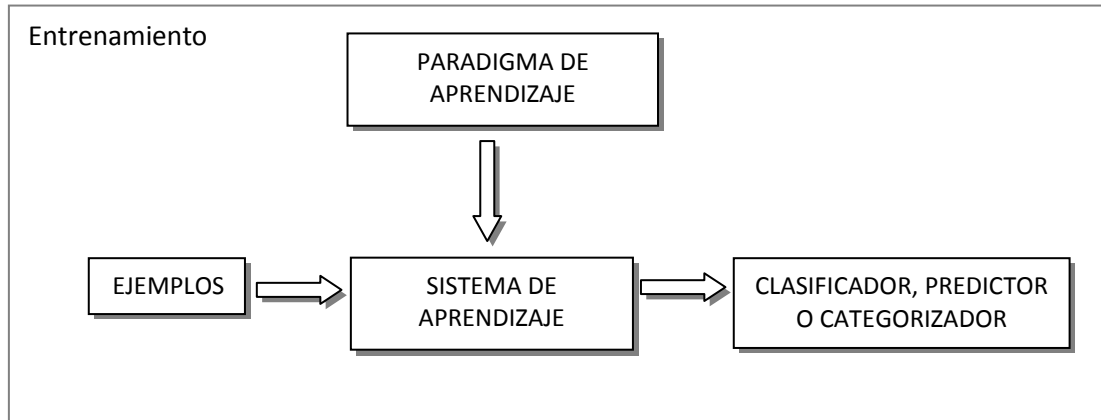


Figura 1.1. Etapa de entrenamiento de un clasificador.

Un paradigma de aprendizaje inductivo intenta aprender conceptos a través de instancias o ejemplos de estos conceptos. La figura 1.2 grafica la etapa de clasificación.

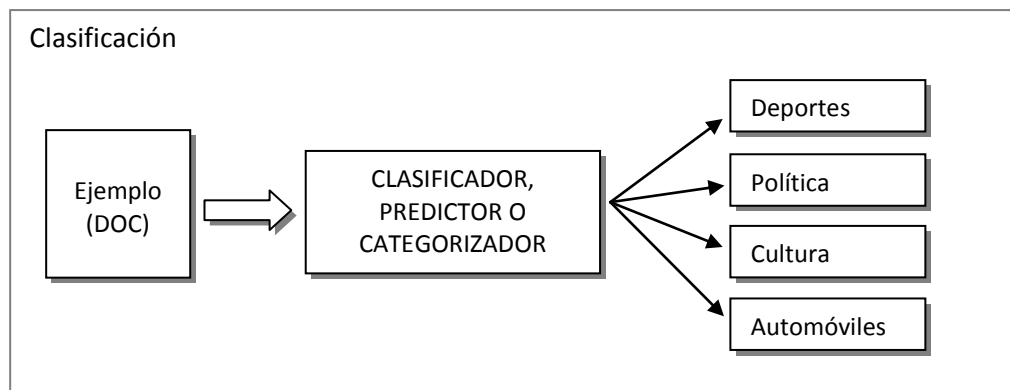


Figura 1.2. Etapa de clasificación.

Clasificación de documentos en la Web

Existen un abundante número de investigaciones realizadas en torno a la clasificación de páginas web. En lo que refiere a la construcción de directorios web, Huang [2, 3] propone una técnica automática para la creación de clasificadores a partir del contenido de los documentos, basándose en categorías definidas previamente por el usuario. Por otro lado, existen estudios que intentan mejorar la calidad de los resultados en los motores de búsqueda tradicionales. Por ejemplo, Chekuri [4] realizó un estudio sobre cómo incrementar la precisión de los resultados a partir de la categorización automática de páginas web. Chen y Dumais [5] presentan una propuesta para mostrar los resultados agrupados por categorías o clusters, en lugar del listado tradicional.

Si bien existen investigaciones sobre clasificación web basándose en el contenido textual, algunas de ellas carecen de un análisis profundo de las características propias de la web. Getoor y Diehl [6] examinaron las técnicas de data mining que permiten contemplar los vínculos entre los objetos.

Fürnkranz [7] investigó varios aspectos del web mining, incluyendo una breve discusión sobre el uso de la estructura de enlaces para mejorar la clasificación web. Muy relacionado con este último artículo se presenta el trabajo de Choi y Yao [8] que describen distintas técnicas para construir sistemas de clasificación automático de páginas web.

También existen estudios que se basan en analizar el contenido y estructura de las páginas. Por ejemplo, Lee Kwon [9] propuso clasificar páginas web usando un algoritmo k -NN⁹ modificado, donde diferentes tags reciben distintos pesos. Shen [10] propone un método para clasificar páginas web a través de un resumen de las mismas. Kan y Thi [11] demostraron que una página puede ser clasificada basándose solamente en su URL. Attardi [12] creó un algoritmo para categorizar automáticamente una página web a partir del análisis de sus links y contenido.

1.2. Contexto

Se llama *web directory* al sitio que contiene un directorio organizado de enlaces a otros sitios, con una estructura de categorías y subcategorías. Yahoo!¹⁰ fue el primero en ofrecer este servicio en 1994. Los webmasters informaban sobre el deseo de que su sitio sea incluido y luego, los editores autorizados de Yahoo!, comprobaban que el enlace se adecuara a los requisitos de aceptación antes de agregarlo finalmente en el directorio.

A medida que Internet fue creciendo, también lo hacían las críticas hacia los directorios web. El principal motivo era que demoraban demasiado tiempo en aprobar un enlace sugerido, sumado a las estructuras organizacionales rígidas y disputas entre los voluntarios editores. Es aquí donde se comienza a pensar en la “categorización automática de sitios web” [13]. Un ejemplo de ello es el caso de Ansearch¹¹, empresa australiana que presta servicios de Internet. Su principal producto es un motor de búsqueda que fue lanzado oficialmente a principios de diciembre de 2004. Actualmente, el directorio de Ansearch se compone de un 90% de enlaces categorizados automáticamente, mientras que el 10% restante –correspondiente a casos especiales– es manejado manualmente. Además, su motor de búsqueda provee rankings basados en la popularidad de los sitios, como así también en el género y la edad.

La categorización automática ayuda a evitar el problema en la demora de la aprobación de un enlace, como se mencionó en el párrafo anterior. Sin embargo, es un gran desafío reemplazar el proceso de clasificación humano debido al alto grado de fiabilidad que posee. En resumen, el reto principal que tendría la clasificación automática es el de *asignar la categoría correcta a cada sitio web*. Existen algoritmos de clasificación que han estado empleándose durante años en otros dominios, donde a partir del contenido –comúnmente el texto– de los documentos, puede determinarse a qué

⁹ Siglas de *k-Nearest Neighbors* (los K vecinos más cercanos).

¹⁰ <http://www.yahoo.com> (sitio principal) y <http://dir.yahoo.com> (directorio).

¹¹ <http://www.ansearch.com/browse>

categoría corresponden. Teniendo en cuenta que las categorías de los sitios web están predefinidas, la propuesta de usar estos algoritmos parece correcta.

Web 2.0 y marcadores sociales

El término **Web 2.0** fue acuñado por Tim O'Reilly en 2004 [14] para referirse a una segunda generación en la historia del desarrollo de tecnología Web basada en comunidades de usuarios y una gama especial de servicios, como las redes sociales, los blogs, los wikis o los marcadores sociales, que fomentan la colaboración y el intercambio ágil de información entre los usuarios de una comunidad o red social. La web 1.0 principalmente trata lo que es el estado estático, es decir, los datos que se encuentran en esta no pueden cambiar, se encuentran fijos, no varían, no se actualizan. La Web 2.0 es también llamada “web social”, por el enfoque colaborativo y de construcción social de esta herramienta. [15]

En ocasiones se ha relacionado el término *Web 2.0* con el de *Web semántica* [16]. Sin embargo ambos conceptos corresponden más bien a estados evolutivos de la web, y la *Web semántica* correspondería en realidad a una evolución posterior, o sea, a la Web 3.0 o web inteligente. La combinación de sistemas de redes sociales con el desarrollo de *tags* o etiquetas, que en su uso social derivan en folcsonomías, así como el plasmado de todas estas tendencias a través de blogs y wikis, confieren a la Web 2.0 un aire semántico sin serlo realmente.

Centrándonos nuevamente en la categorización de documentos, está comprobado que cuanto mayor información pueda ser procesada por un algoritmo de clasificación, se espera que mejores sean las predicciones obtenidas sobre un conjunto de ítems a clasificar. Como se dijo anteriormente, la Web 2.0 brinda nuevas fuentes de información, entre ellas los denominados *marcadores sociales* (o anotaciones sociales). En esta clase de sistemas, los usuarios guardan una lista de recursos que consideran útiles, de la misma forma que lo hacen cuando agregan una página web a favoritos en su navegador. Entre los servicios más populares de marcadores sociales se encuentran Delicious¹² y Google Bookmarks¹³ para la administración de páginas webs, Bloglines¹⁴ y Technorati¹⁵ hacen lo mismo con blogs y videos, Digg¹⁶ y Menéame¹⁷ orientados a las noticias, y muchos otros. Las listas pueden ser accedidas públicamente, esto es importante para aquellas personas con intereses comunes. Los recursos son categorizados mediante el uso de etiquetas o *tags*, que no son más que palabras relacionadas con el artículo y asignados por los mismos usuarios. Su popularidad va creciendo a diario, por lo que no sería absurdo pensar que en un futuro puedan convertirse en un medio importante a la hora de buscar información.

¹² <http://www.delicious.com>

¹³ <http://www.google.com/bookmarks>

¹⁴ <http://www.bloglines.com>

¹⁵ <http://technorati.com>

¹⁶ <http://www.digg.com>

¹⁷ <http://www.meneame.net>

1.3. Propuesta

El objetivo del presente trabajo consiste en estudiar la importancia de los marcadores sociales en la clasificación de documentos web [17]. En teoría, muchos investigadores afirman que podrían ser valiosos ya que aportan información sobre el contenido del recurso –sea página web, video, noticia, etc.– desde un punto de vista humano, que en la mayoría de los casos resulta ser más preciso que el obtenido por parte de los algoritmos de extracción de datos; conjuntamente al hecho que brindan información adicional que no está presente en los documentos mismos.

En la práctica, los *tags* son escogidos libremente en lugar de ser tomados de un vocabulario propuesto, sumándose además la presencia de sinónimos, o casos especiales donde un mismo *tag* tiene varios significados [18]. Esto tendería a disminuir la eficiencia del algoritmo de clasificación. Para ello, existen técnicas diversas como *stemming* –normalizar una palabra– y eliminación de *stop-words* –palabras que no aportan información a la clasificación, como las preposiciones– que se utilizarán en el presente trabajo.

Por lo tanto, si a los algoritmos de clasificación se los puede alimentar con la información de los marcadores sociales creados por los propios usuarios, en lugar de o sumado a los datos recolectados por los algoritmos de extracción (como el texto u otras características), se estima que las predicciones deberían verse mejoradas. Esto es lo que se intentará dilucidar en la corriente investigación.

Para ello, se tomará como punto de partida una colección de documentos web [19] compuesta por 120.000 URLs con metadatos, como consultas de los motores de búsquedas (o *queries*) que llevaron a esa URL, categorías asignadas, marcadores sociales (o *tags* que los usuarios han colocado en Delicious a cada URL) y *anchor text* (es el texto visible y cliqueable en un hipervínculo de una página web) de los enlaces. Se crearán distintas instancias de esta colección (*datasets*), previamente convertidas –utilizando un *parser*¹⁸– al formato requerido por la herramienta de clasificación Weka [20], que las procesará entregando los resultados de cada una. Las instancias tendrán distintas combinaciones de metadatos que serán analizados en cuanto a su grado de clasificación. Posteriormente, se describirán y estudiarán los resultados obtenidos, llegando finalmente a ensamblar una conclusión sobre el tema.

1.4. Alcance

El alcance de este trabajo consistirá en evaluar si efectivamente los marcadores sociales son realmente útiles para ser utilizados en la clasificación automática de documentos web. Se generarán diferentes colecciones de datos, donde cada una estará compuesta por distintas combinaciones de las fuentes de información antes mencionadas (*tags*, *queries* y *anchortext*). Luego, estas colecciones serán sometidas a variados procesamiento de texto en su contenido, como ser *stemming*, eliminación de *stop-words*,

¹⁸ Ver 4.2. Construcción del Parser (CABS120k08 a formato ARFF).

aplicación de sinónimos, etc. Todo esto sumado a diversos clasificadores con distintas configuraciones de los mismos.

Como conclusión final, se determinará cuál es la combinación ideal para ser utilizada en un algoritmo de clasificación, comparando los distintos resultados y deduciendo el por qué de dicha decisión.

1.5. Organización de la tesis

El documento se organiza de la siguiente manera: En el capítulo 1 se realiza una introducción al tema, brindando un marco teórico necesario para comprender el resto de la investigación. Se explica el concepto y la aplicación de lo que se denomina *data mining*, como así también nos ubica en contexto en lo que hace a la web 2.0 y los directorios de páginas web; culminando luego con la presentación de la propuesta de trabajo dilucidando cuál es la problemática que se intenta resolver. El segundo capítulo denominado “Marcadores sociales en la Web” comienza describiendo la composición de los sistemas de *tagging colaborativos*. Luego, se concentra en explicar en detalle qué son los marcadores sociales, para finalmente enumerar los distintos trabajos de investigación realizados en el área de la clasificación de páginas web empleando tags.

A lo largo del capítulo 3 se describen los recursos empleados en la presente investigación, comenzando con la colección de datos CABS120k08, y las distintas funcionalidades y características de la herramienta Weka. Finalizando luego con el parser que se ha construido para convertir los datos iniciales en el formato necesario para su posterior clasificación. El capítulo 4 es el más extenso e importante del informe, dónde se describe el desarrollo de la investigación llevada a cabo. Comienza explicando la generación y clasificación de los distintos datasets. Luego, realiza un análisis de los resultados obtenidos y elabora una serie de intentos de optimización.

Por último, se presenta una conclusión donde se observan cuáles han sido los logros alcanzados y qué futuras posibles extensiones pueden realizarse a esta investigación.

Capítulo 2: Marcadores sociales en la Web

2.1. Las etiquetas (o tags)

¿Qué son exactamente los tags? Según [21] una simple definición podría ser que los tags no son más que keywords, nombres de categorías o metadatos. Básicamente, un tag es un conjunto de palabras claves sobre un texto escogidas libremente. Sin embargo, como los tags no son creados por especialistas de la información, no siguen ninguna regla formal de escritura. Esto significa que los ítems pueden ser categorizados con cualquier palabra que defina una relación entre un recurso online y un concepto según el pensamiento del usuario. Cualquier número de palabras pueden ser elegidas, algunas de ellas son representaciones obvias, otras no tanto.

Los dos servicios más populares basados en folcsonomías, Delicious y Flickr¹⁹, presentan un sistema de navegación y búsqueda mediante etiquetas. A fin de comprender cómo esto puede verse mejorado, es importante primero entender a los usuarios. Actualmente, se conoce muy poco acerca del proceso de decisión que existe en la selección de un tag. Un enfoque útil podría ser examinar los motivos de los usuarios cuando agregan tags, por qué se deciden a utilizar determinadas palabras, observar cuántas etiquetas agregan y comparar cómo los mismos ítems son clasificados por diferentes usuarios.

Existe un estudio [22] que plantea cuestiones muy interesantes, aunque no encuentra el motivo de por qué ciertas decisiones son llevadas a cabo. Una de las razones que se extraen de este estudio es que, aunque los tags comúnmente utilizados tienen un significado “oculto” que sólo conoce su creador, está claro que hay ciertas etiquetas que tienen un significado social para compartir junto con el significado personal. Son estas etiquetas las que brindan el mayor beneficio, por lo tanto existen métodos para fomentar su creación y uso.

Muchos sitios ofrecen visualizaciones sobre los tags más populares elegidos en común, entre ellos: tag.alicio.us²⁰, extisp.icio.us²¹ y facetious²². Tag.alicio.us es un diseño experimental creado por Olivier Richard que opera como un filtro de tags, extrayendo links desde Delicious de acuerdo a determinados criterios (por ejemplo, las etiquetas de la hora actual, del día o de la semana). Extisp.icio.us muestra una dispersión aleatoria de las etiquetas de un determinado usuario, con distintos tamaños de acuerdo al número de veces que el usuario ha reutilizado cada etiqueta. Y facetious es un *reworking* de la base de datos de Delicious. Que hace uso de la clasificación por facetas, agrupando tags bajo distintas

¹⁹ Flickr es un servicio web que permite almacenar, ordenar, buscar y compartir fotografías y videos online. Es propiedad de Yahoo! <http://www.flickr.com>

²⁰ <http://frenchfragfactory.net/ozh/archives/2004/10/05/tagalicious-a-way-to-integrate-delicious/>

²¹ <http://kevan.org/extispicious>

²² <http://www.siderean.com/delicious/facetious.jsp>

cuestiones como ser “por lugar” (Iraq, USA, Australia), “por tecnología” (blog, wiki, website) y “por atributo” (rojo, gracioso, retro).

En [23], se sugiere que la distribución de los tags persigue un escenario de *power law*. Los tags más usados son altamente visibles así que pueden ser utilizados por otros usuarios (pocas etiquetas utilizadas por muchos). Luego, existe un gran número de tags que son usados sólo por un pequeño número de usuarios (muchos tags usados por pocos). Y finalmente, existirá un gran número de tags que serán usados por uno o dos usuarios solamente.

2.2. Tagging colaborativo

Los sistemas de *tagging colaborativo*, también conocidos como *folcsonomías* (del inglés *folksonomy*) o *social tagging*, se basan en una indexación social, es decir, clasificación colaborativa por medio de etiquetas (tags) simples en un espacio de nombres sin jerarquías ni relaciones de parentesco predeterminadas [24, 25]. Se trata de una práctica que se produce en entornos de software social, como Delicious y Flickr, por nombrar los más populares. Si se compara con otros sistemas de categorización, como el de Gmail [26] que también se vale de etiquetas, se distingue en que en este último los usuarios no comparten sus categorizaciones (sencillamente porque sus mails son recursos privados).

Los sistemas de *tagging colaborativo* surgen cuando varios usuarios participan en la descripción de un mismo material informativo. Por ejemplo, en Delicious muchas personas han guardado la página de Wikipedia marcándola con diferentes *tags*, pero coincidiendo la mayoría en *referente*, *wiki* y *encyclopedia*. Jon Udell [18] sugiere que “el abandono de las taxonomías en favor de las listas de palabras claves no es novedad, y que su diferencia fundamental es el intercambio de opiniones (el *feedback*) que se da en la folcsonomía y no en la taxonomía.”

Más en detalle, una folcsonomía se compone de un conjunto de anotaciones, cada una vinculada con tres entidades (usuarios, tags y recursos) que se relacionan entre sí de diferentes maneras. Formalmente:

Como se define en [27], una folcsonomía F es una tupla tal que $F = \{ U, T, R, A \}$ donde U : conjunto de usuarios, T : conjunto de tags, R : conjunto de recursos, A : conjunto de anotaciones tal que $A \subseteq U \times T \times R$

El siguiente diagrama ilustra de manera simple y comprensiva las distintas relaciones entre los usuarios, tags y recursos.

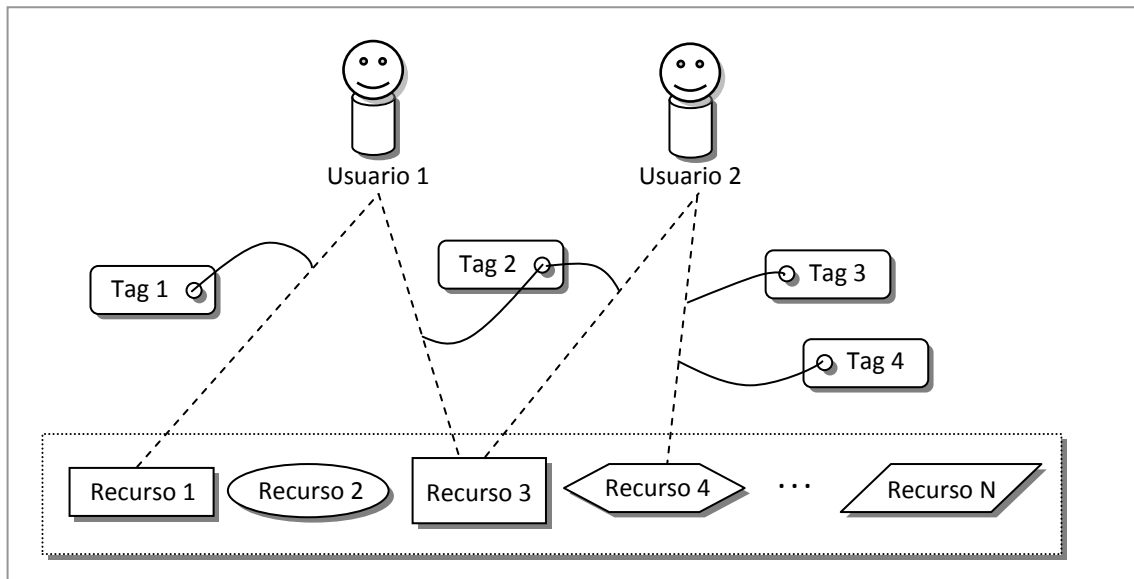


Figura 2.1. Representación gráfica de una folcsonomía.

Como se aprecia en la Figura 2.1, la combinación de usuarios, tags y recursos da lugar a la conformación de un conjunto de anotaciones ($U \times T \times R$). Entre ellas se puede señalar, por ejemplo, que el Usuario 1 ha etiquetado el Recurso 1 usando el Tag 1 ($U_1 \times T_1 \times R_1$). Mientras que el Usuario 2 hace lo mismo con el Recurso 4, pero colocando dos tags en lugar de uno ($U_2 \times T_3 \times R_4$ y $U_2 \times T_4 \times R_4$). Por otro lado, ambos usuarios etiquetan con el mismo tag al Recurso 3 ($U_1 \times T_2 \times R_3$ y $U_2 \times T_2 \times R_3$).

¿De dónde proviene el término “folksonomy”? Derivado de *taxonomía*, la palabra *folksonomy* ha sido atribuida a Thomas Vander Wal [28]. Taxonomía procede del griego “*taxis*” y “*nomos*”: *taxis* significa clasificación y *nomos* (o *nomia*), ordenar, gestionar; por su parte, “*folc*” proviene del alemán “pueblo” (Volk). En consecuencia, de acuerdo con su formación etimológica, folcsonomía (folc+taxo+nomía) significa literalmente “clasificación gestionada por el pueblo (o democrática)”.

Es importante destacar que las folcsonomías funcionan mejor cuando los tags usados para describir un recurso forman parte de un mismo vocabulario en común, y no a la terminología propia de la persona que generó el tag – ya que esta herramienta no sólo trabaja como un sistema de administración personal de la información en la web, sino que también es utilizada para compartir información interesante con el resto del mundo.

Folcsonomías amplias y estrechas

Flickr, Delicious y otras herramientas sociales que aprovechan el poder de las folcsonomías, apuntan a diferentes perfiles de usuarios y muestran distintas tendencias de uso. En otras palabras, la naturaleza de las folcsonomías que producen es muy diferente.

Como se explicó en [29], se pueden distinguir dos topologías de folcsonomía, cada una asociada a propiedades específicas y sugerencias de uso: folcsonomías amplias y folcsonomías estrechas.

Una **folcsonomía amplia** (broad)²³ (como la utilizada en Delicious) es el resultado de mucha gente *taggeando*²⁴ un mismo item. Cada usuario puede taggear cada objeto de una forma diferente siguiendo su propio modelo mental, vocabulario y lenguaje. Este enfoque tiende a mostrar una **power law curve**²⁵ y un **efecto long tail**²⁶.

En una folcsonomía broad, la *power law* revela que mucha gente coincide en utilizar unos pocos tags populares; y por otro lado, algunos grupos más pequeños a menudo prefieren emplear términos menos conocidos para describir sus artículos de interés. Por lo tanto, una folcsonomía broad provee una importante herramienta para investigar las tendencias en grandes grupos de personas que describen el corpus de artículos y puede ser utilizado para seleccionar cuáles son los términos preferidos para luego extraer un vocabulario controlado.

El verdadero poder de las folcsonomías broad está en la riqueza de las masas, en las personas que de forma explícita exponen su manera de definir y describir las cosas.

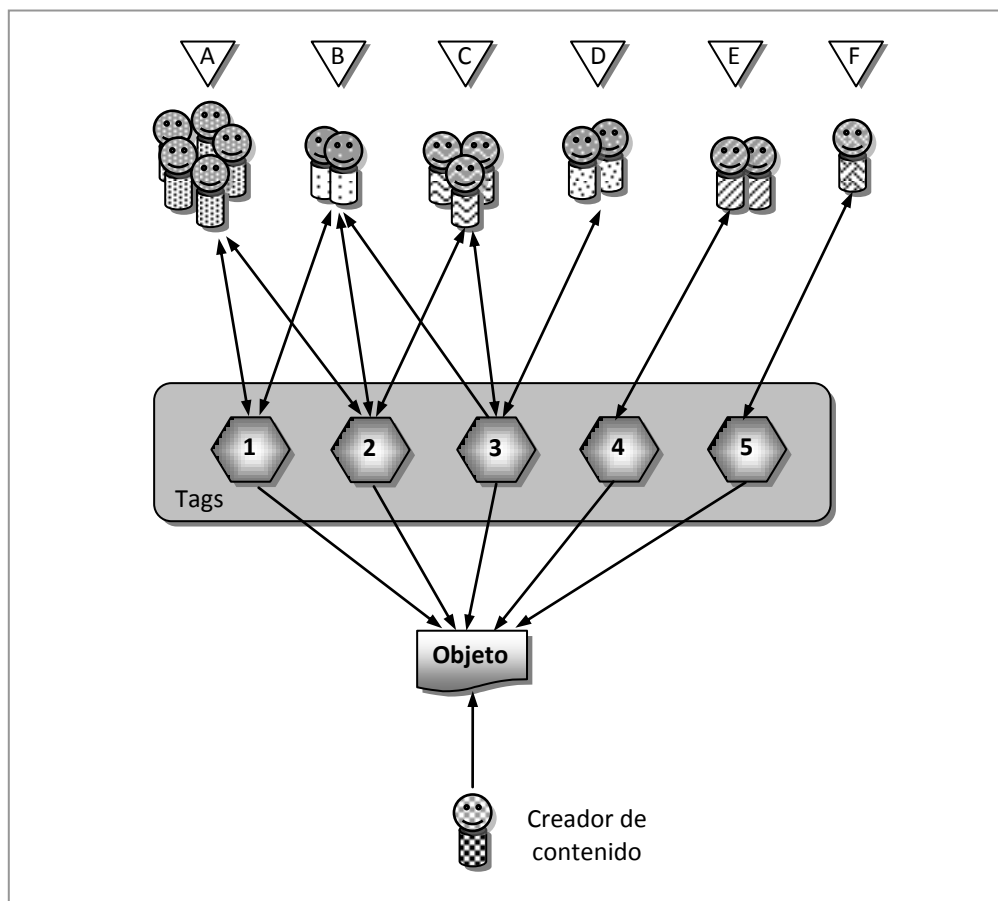


Figura 2.2. Representación gráfica de una folcsonomía broad.

²³ Del inglés *general* o *ancho*.

²⁴ Del verbo *colocar un tag*.

²⁵ Del inglés *Curva de ley de potencia*. Ver Glosario.

²⁶ Del inglés *Efecto de cola larga*. Ver Glosario.

En la figura 2.2, se puede ver cómo distintos grupos de usuarios han taggeado el mismo objeto empleando diferentes tags. Por ejemplo, los 5 usuarios del grupo “A” han taggeado el objeto utilizado los tags “1” y “2”, usando también esos términos para encontrar el objeto nuevamente. Por otro lado, los 2 usuarios del grupo “B” hicieron lo mismo, con la diferencia que utilizan el tag “3” para encontrar información.

Una **folcsonomía estrecha** (narrow)²⁷ (como el empleado en Flickr), por el otro lado, es el resultado de un pequeño número de individuos taggeando ítems (usando uno o más tags) para recuperarlos más tarde o bien para su propia conveniencia.

La folcsonomía narrow pierde la riqueza de las masas, pero provee beneficios en taggear objetos que no son fácilmente encontrados con las herramientas tradicionales (ej. full-text search²⁸) o que no puede ser simplemente descrita con el software actual basado en texto que tiene la web.

Una folcsonomía narrow está orientada a distintas audiencias (tal vez con un vocabulario bastante específico), permitiendo agregar tags en su propio idioma. Esta propiedad hace más fácil y eficiente la recuperación de información (ej. obtener las fotos almacenadas en Flickr basándose en algún criterio como “New York” o “apple”).

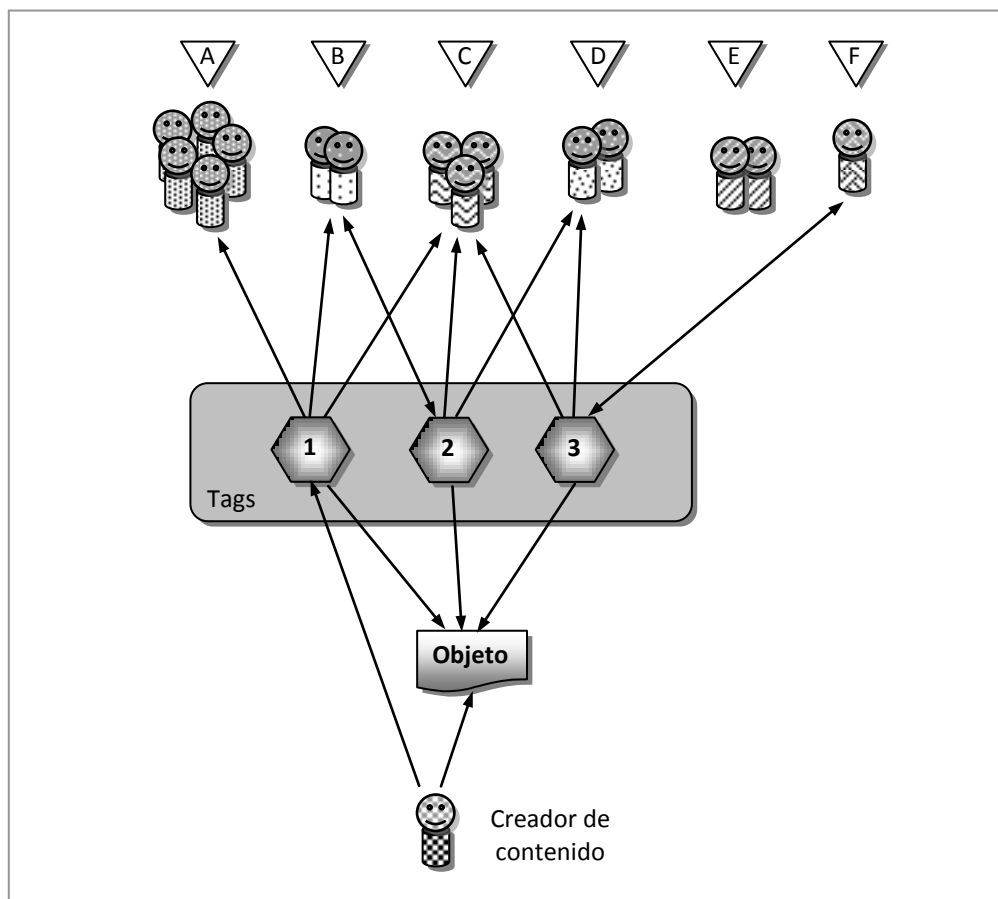


Figura 2.3. Representación gráfica de una folcsonomía narrow.

²⁷ Del inglés *estrecho* o *angosto*.

²⁸ Técnica para buscar información textual almacenada en un documento o base de datos.

En la figura 2.3, se aprecia la interacción de las personas, los objetos y los tags en una folcsonomía narrow. Por ejemplo, el grupo “A” utiliza el tag “1” para encontrar el objeto en cuestión, aunque el creador de ese contenido, que fue quién colocó el tag, seguramente no tenía ese propósito. Por otro lado, se ve que el grupo “B” también utilizó el tag “1” para encontrar el objeto, pero además colaboró agregando el tag “2” al recurso. El grupo “F” no encontró el objeto utilizando los tags existentes, sino que debe haber llegado al contenido a partir de otros medios, por ejemplo con un link enviado por email.

Las folcsonomías son criticadas debido a que su falta de control terminológico tiende a causar resultados inconsistentes y poco confiables. Si las etiquetas son escogidas libremente (en lugar de ser tomadas de un vocabulario propuesto), sumado a que *sinónimos* (múltiples etiquetas del mismo concepto), *homonimia* (misma etiqueta con diferente significado), y *polisemia* (misma etiqueta con múltiples significados relacionados) tienden a aparecer, disminuye notablemente la eficiencia de la búsqueda del contenido indexado. Otra razón para lo que se conoce como *meta noise*²⁹ son la falta de *stemming* (normalización de palabra) y la heterogeneidad de usuarios y contextos.

2.3. ¿Qué son los “marcadores sociales”?

Los marcadores sociales, también conocidos como “anotaciones sociales” o *social bookmarks*, son una forma sencilla y popular de almacenar, clasificar y compartir enlaces en internet o en una intranet. Es una de las distintas implementaciones que se generan a partir del concepto de *tagging colaborativo* o *folcsonomía*.

En un sistema de marcadores sociales, los usuarios guardan una lista de recursos –páginas web, videos, noticias– de Internet que consideran útiles. Las listas pueden ser accesibles públicamente o de forma privada. Otras personas con intereses similares pueden ver los enlaces agrupados por categorías, etiquetas o al azar.

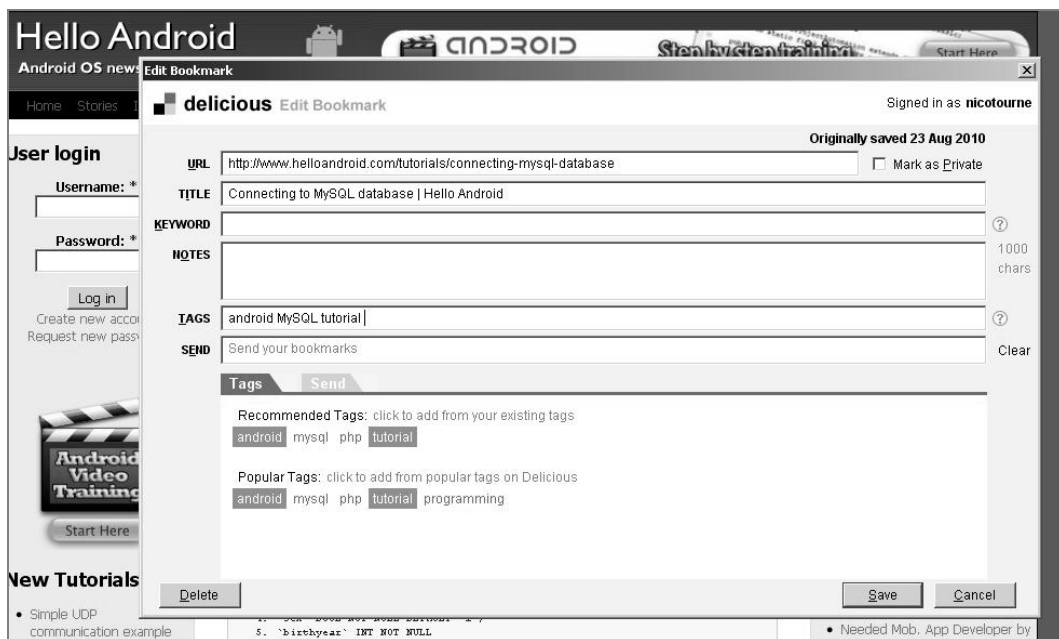
Los recursos son categorizados mediante el uso de etiquetas (también llamadas *tags*), que no son más que términos relacionados con el mismo y asignadas por los propios usuarios. La mayoría de los servicios de marcadores sociales permiten que los usuarios busquen recursos asociados con ciertos *tags* y ordenen esos recursos según la cantidad de veces que han sido *taggeados*.

Hoy en día existe un gran número de sitios web que brindan el servicio de gestión de marcadores sociales, entre ellos el más popular es **Delicious**³⁰ (antes conocido como “del.icio.us”). Este servicio ofrece la posibilidad de instalar un plugin en el navegador web para acceder de manera simple a su

²⁹ Se refiere a metadatos incorrectos o irrelevantes. Es frecuente en sistemas que cuentan con un esquema no basado en un vocabulario controlado, como ciertas folcsonomías.

³⁰ <http://www.delicious.com>

funcionalidad. El mecanismo es muy sencillo: cuando un usuario encuentra alguna página interesante, no tiene más que guardar el link en su cuenta “marcándola” con distintos tags que considere adecuados, los cuales le servirán para luego poder recuperarla fácilmente. El sistema también sugiere algunos términos con los cuales taggear al recurso, basándose en los tags colocados por los demás usuarios sobre la misma página. Este proceso se grafica en la Figura 2.4. Cabe aclarar que las páginas pueden ser almacenadas públicamente (donde es posible acceder al perfil de cualquier usuario y conocer los recursos que ha guardado, en una forma de compartir conocimiento) o bien de manera privada (sólo el propio usuario tiene acceso).



The screenshot shows the 'delicious Edit Bookmark' interface. The user is signed in as 'nicotourne'. The form contains the following fields:

- URL:** ☐ Mark as Private
- TITLE:**
- KEYWORD:**
- NOTES:**
- TAGS:**
- SEND:**

Below the form, there are two sections for tag suggestions:

- Recommended Tags:** click to add from your existing tags. Tags:
- Popular Tags:** click to add from popular tags on Delicious. Tags:

At the bottom, there are buttons for 'Delete', 'Save', and 'Cancel'. The sidebar on the left includes a 'User login' section with fields for 'Username' and 'Password', and a 'New Tutorials' section with a list of tutorial links.

Figura 2.4. Captura de pantalla para agregar una nueva página en Delicious.

El sitio web de Delicious ofrece una simple pero interesante navegación sobre las páginas agregadas a la cuenta. Tan sólo hay que tipear una palabra clave o bien seleccionar algunos de los tags creados (hacia la derecha en la Figura 2.5), para poder visualizar los recursos que cumplen con esa condición.

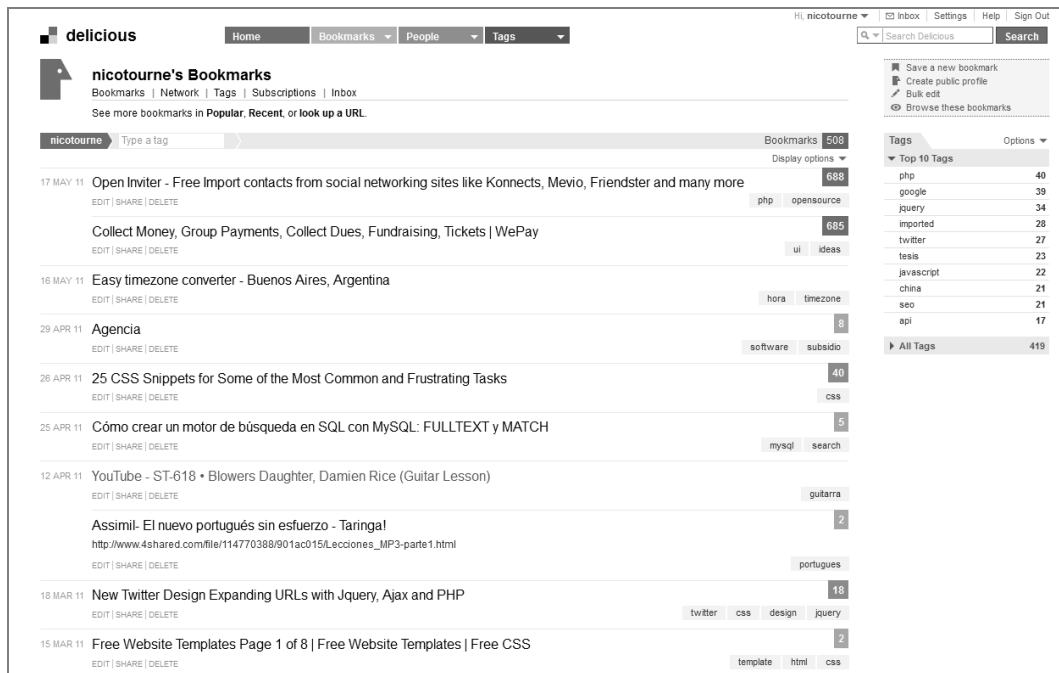


Figura 2.5. Navegación de los documentos marcados en Delicious partir de los tags.

Otra de las características interesantes consiste en la posibilidad de realizar búsquedas por medio de tags sobre todas las páginas que han sido almacenadas en el sistema por los distintos usuarios. Como cualquier motor de búsqueda, la consulta devuelve todos aquellos documentos que contengan alguno de los tags ingresados, pero presentándolos en forma ordenada de acuerdo a la cantidad de veces que el documento ha sido taggeado, sumado a otros factores como la fecha de creación y la cantidad de visitas, por ejemplo.

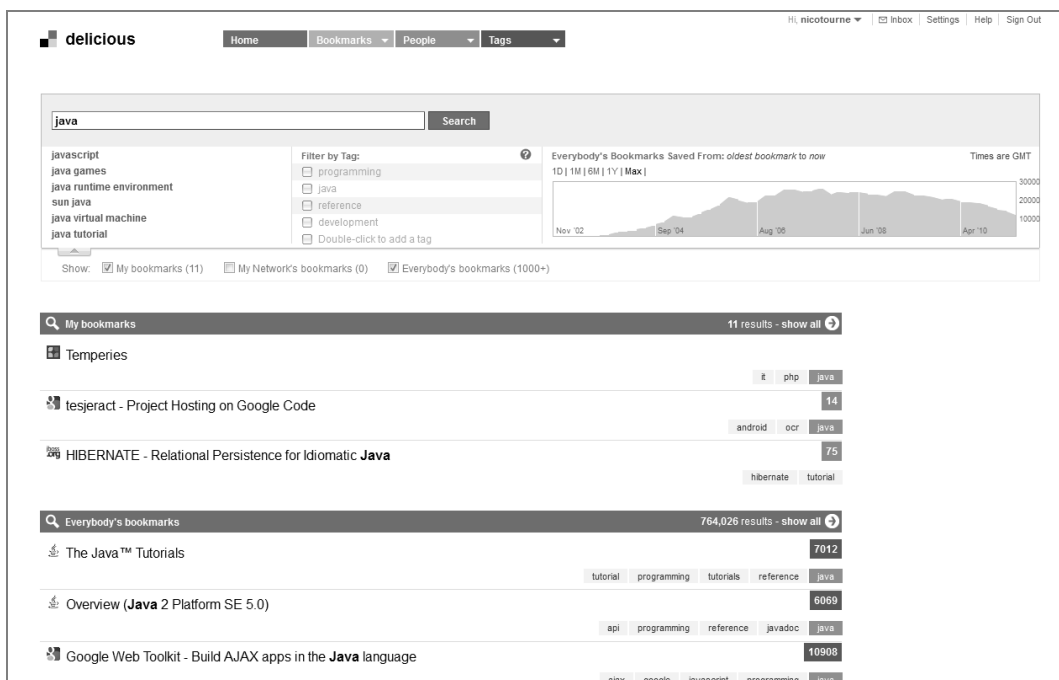


Figura 2.6. Búsqueda de páginas web almacenadas en Delicious basándose en tags.

Algo interesante de Delicious es que posee una API que permite a los desarrollados poder interactuar fácilmente con el sistema. Es propiedad de Yahoo! desde el año 2003. El servicio es totalmente gratuito.

El resto de los servicios de marcadores sociales presentan características muy similares, entre ellos se puede nombrar:

- *BlinkList* – Se autodenominan el iTunes para páginas web. Intuitivo, fácil de usar e increíblemente rápido. www.blinklist.com
- *Digg.com* – Combina marcadores sociales, blogging y sindicación con una organización sin jerarquías, con control editorial democrático, lo cual permite que se publiquen artículos sobre una gran variedad de géneros. Los usuarios envían relatos de noticias y recomendaciones de páginas web, y los ponen a disposición de la comunidad, quienes clasifican esos contenidos. www.digg.com
- *Bloglines* – Es un agregador de noticias para leer blogs y otras fuentes. www.bloglines.com
- *Reddit* – Sitio web de marcadores sociales que tiene la particular que los usuarios pueden votar a favor o en contra de los enlaces sugeridos, influyendo en qué tan destacados son. www.reddit.com
- *Kaboodle* – Inicialmente enfocado sobre el “social Shopping” (aprovechar las búsquedas de otras personas para facilitar sus propias compras), permite facilitar la comparación de ofertas, productos y servicios creando una página pública o privada de síntesis. Es posible conectarse con varias personas que tienen intereses comunes. www.kaboodle.com
- *Technorati* – Orientado principalmente a la búsqueda de contenido de blogs y sitios de noticias. www.technorati.com
- *Menéame* – Basado en noticias que ha copiado el modelo de Digg.com. En español. www.meneame.net
- *Copada.net* – Similar a Digg.com, donde los usuarios recomiendan noticias las cuales luego son calificadas y ordenadas de acuerdo a su puntuación. www.copada.net
- *Keegy* – Similar a Digg.com, pero con noticias nacionales. ar.keegy.com

Otros servicios en español: Memorizame³¹, dir.eccion.es³² y Favoriting.com³³ (el más popular en español).

³¹ <http://www.memorizame.com>

³² <http://dir.eccion.es>

³³ <http://www.favoriting.com>

Los marcadores sociales presentan varias **ventajas** sobre otras herramientas tradicionales como los motores de búsqueda y los directorios web que organizan la información en taxonomías. Toda la inclusión y clasificación de recursos está realizada por los mismos usuarios en lugar de un software que procesa información de manera automática según un determinado algoritmo.

Los recursos realmente útiles son marcados por un mayor número de personas. De esta forma se crea un ranking basado en el criterio de los mismos usuarios. Es una manera distinta de medir la utilidad de los recursos, en algún grado mejor que la que proporcionan otros sistemas automatizados como los que se basan en el número de enlaces externos (ej. PageRank de Google [31]).

También existen algunas **desventajas** de los sistemas basados en *tags*: no hay un método pre-establecido de palabras claves o categorías; no existe una estructura predefinida para los tags (ej. plural/singular, mayúsculas, acentos, etc.) lo cual puede llevar a errores a la hora de deletrear el mismo. Los *tags* pueden tener más de un significado y ofrecer resultados inexactos debido a confusiones entre sinónimos, los usuarios pueden crear *tags* demasiados personalizados con poco significado para otros. No existe forma para estructurar los *tags* de manera jerárquica (por ejemplo, un sitio puede ser etiquetado con los *tags* “queso” y “cheddar”, pero no existe forma de indicar que “cheddar” es un tipo de “queso”).

2.4. Clasificación de páginas web empleando tags

El problema de la categorización de páginas web empleando tags como recurso principal, ha sido abordado hasta el momento en algunos trabajos que se detallan a continuación.

En [32] se estudian tres diferentes pero relacionados tipos de metadatos vinculados con documentos web: anotaciones sociales provistas por los lectores de la web, anchor-text de los hipervínculos provistos por los autores de los documentos, y el log de consultas en las búsquedas de los usuarios tratando de encontrar estos recursos. El objetivo final es la investigación de varias características de los metadatos como ser longitud, novedad, diversidad y similitud; y la discusión de implicaciones teóricas y prácticas. Este estudio produjo algunos resultados interesantes. Entre ellos, se encontró que los marcadores sociales son particularmente útiles para identificar el “acerca de” los documentos web, o sea, para descubrir información que no está presente en el recurso mismo.

En [17] se exploran y analizan las anotaciones sociales y tagging prestando atención en su utilidad para la clasificación de documentos web, teniendo en cuenta que muchos usuarios agregan metadatos en forma no estructurada de *keywords*³⁴ para compartir contenido. Se interesa en descubrir qué clase de documentos son marcados más comúnmente por los usuarios finales, cómo los usuarios tienden a marcar estos documentos y, en particular, cómo la folsonomía generada por el usuario es comparada

³⁴ Palabras claves, que se escriben de manera especial para poder ser localizadas fácilmente.

con la taxonomía mantenida por expertos de la clasificación para el mismo conjunto de documentos. Describe qué puede ser deducido de los resultados de esta investigación y del desarrollo en áreas de la clasificación de documentos y la recuperación de información. Luego, en [33] se presenta un nuevo enfoque de personalización de las búsquedas web basadas en la colaboración del usuario y el compartir información de los documentos. La técnica de personalización propuesta separa la colección de datos y el perfil del usuario del sistema de información cuyo contenido y documentos indexados están siendo rastreados, por ejemplo los motores de búsqueda, y el uso de los marcadores sociales y el tagging para re-ranear los resultados de la búsqueda web.

Cómo se mencionó en [33], los marcadores sociales son utilizados en sistemas de recomendación e *information retrieval* (IR, ver Glosario). Se han realizado publicaciones bastante reconocidas, como [34] donde destacan el rápido crecimiento de los marcadores sociales en la web, aunque todavía no existan investigaciones fundamentales para estos sistemas. Aquí presentan un modelo formal y un nuevo algoritmo de búsqueda basado en folcsonomías, llamado FolkRank, que aprovecha la estructura de las mismas. Se han realizado varias investigaciones entre sistemas de marcadores sociales y búsqueda tradicional [35]. Se espera que los motores de búsqueda más populares puedan incorporar los marcadores sociales en sus sofisticados algoritmos.

Por otro lado, [36] analiza y evalúa la utilidad de las anotaciones sociales (de distintos servicios de marcadores) para clasificar páginas webs bajo una taxonomía como la propuesta por el Open Directory Project³⁵. Sus experimentos muestran resultados alentados en cuanto al uso de anotaciones sociales para este propósito, descubriendo que la combinación de estos metadatos con el contenido de las páginas web mejora aún más la performance del clasificador. Por último, [37] presenta una metodología para obtener y visualizar una *tag cloud* (ver Glosario) basado en el uso de mapas auto-organizados, donde las relaciones entre los tags son establecidas tomando en cuenta el contenido textual de los documentos taggeados.

También [38] realiza una investigación sobre la clasificación web respecto a sus características y algoritmos, y elabora un resumen del conocimiento aprendido a partir de las investigaciones ya existentes, mencionando futuros desarrollos y aplicaciones en cuanto a la clasificación web. Se descubrió que mientras que el uso apropiado de características textuales y visuales que residen directamente en la página pueden mejorar los resultados de la clasificación, las características de páginas vecinas facilitan información adicional significativa a la página a clasificar.

En [39] se investiga el uso de tags, provenientes de Delicious, para la clasificación de páginas web. El mecanismo de prueba presentado consiste en aplicar diferentes propuestas automáticas extendiendo un directorio web con nuevas URLs a partir de los tags colocados por los mismos usuarios. Los resultados mostraron que la precisión de los documentos clasificados se incrementa con el número de usuarios que taggearon una URL: más páginas web taggeadas mejora la clasificación.

³⁵ <http://www.dmoz.org>

Por otro lado, [40] aborda aspectos prácticos de la clasificación de páginas web que no son tratados por los frameworks clásicos de text mining. Estudia técnicas para la construcción automática de conjuntos de datos de entrenamiento a partir de los recursos disponibles de la web. Llega a la conclusión en que es factible construir clasificadores de alta precisión de la Web 2.0 con poco esfuerzo y que funcionen realmente bien a partir de distintas fuentes de datos.

Existe una interesante investigación en [41], el cual estudia el problema de clasificar objetos web presentes en distintos formatos, especialmente no-textuales, como ser imágenes y videos. Propone un eficiente algoritmo que no sólo utiliza tags como una vía para enriquecer semánticamente los objetos, sino que también deduce las categorías que le corresponden a los objetos sin tags a partir de la conexión implícita de los marcadores sociales. Los resultados indican que la exploración de los tags efectivamente aumenta la clasificación de los objetos web.

Así como se encuentran variadas investigaciones basándose en el contenido de Delicious y Flickr, también las hay a partir de Wikipedia³⁶. En [42] se proponen métodos de clasificación para predecir las categorías de los artículos de esta enciclopedia virtual. También en [43] se presenta un método genérico de clasificación basado en tags usando tanto Wikipedia como Open Directory Project, que se basa en categorizar primero los artículos de la enciclopedia y para luego mapearlos con los tags de algunas fotos de Flickr. Otra publicación interesante es [44], donde por un lado analiza las ventajas de utilizar información extraída de los marcadores sociales con el fin de mejorar la navegación dentro del sitio web de la Wikipedia; y por el otro, estudia cómo los mismos marcadores pueden perfeccionar el motor de búsqueda de la enciclopedia.

Por otro lado, en cuanto a sistemas de recomendación, [45] hace hincapié a que las complejas redes creadas por varias anotaciones a partir de los sistemas colaborativos, brindan la libertad de explorar *tags*, recursos e incluso otros perfiles de usuarios. Sin embargo, esta libertad ofrecida a los usuarios tiene un costo: vocabulario incontrolado que puede resultar en una redundancia de *tags* y ambigüedad que dificulta la navegación. Las técnicas de *data mining*, como *clustering*, proveen un remedio a estos problemas identificando tendencias y reduciendo el ruido. Los *tag clusters* también pueden ser usados como base para efectivos sistemas de recomendación personalizados. Aquí se presenta un algoritmo de personalización para recomendaciones en folcsonomías que se basa en *tag clusters* jerárquicos.

Por un lado, como se explicó anteriormente, los tags son útiles para organizar información, pero presentan algunas deficiencias como ser la creación manual y no controlada de los mismos. Por el otro, una ontología es una representación formal del conocimiento basado en un conjunto de conceptos dentro de un dominio y tomando en cuenta la relación entre esos conceptos. Las ontologías tienen un buen potencial para mejorar la organización, administración y comprensión de la información. En [46] se propone un enfoque automático de recomendación de nuevos tags para cualquier recurso web (tal como posee Delicious), basándose en estas ontologías.

³⁶ <http://www.wikipedia.org>

Resumen del capítulo 2

En este capítulo se realizó una introducción al dominio en el cual se desarrolla la presente investigación. Se introdujo el concepto de tagging colaborativo, enumerando tanto sus ventajas y desventajas, como así también las distintas clases de folcsonomías existentes: broad y narrow. Se mencionaron las críticas que sufre este tipo de sistema, como la falta de control terminológico debido a la libre elección de las etiquetas.

Luego, se explicó el funcionamiento de los marcadores sociales, teniendo como principal utilidad la de almacenar, clasificar y compartir enlaces en Internet. Se enumeraron algunos sitios que brindan este servicio, en especial, Delicious. También se citaron las ventajas y desventajas que presenta.

Por último, se examinó el uso de las etiquetas (o tags), listando sus principales utilidades y los dominios donde se aplica; sumado a un análisis de las investigaciones llevadas a cabo sobre clasificación de páginas web empleado tags, a través de diversos autores.

Capítulo 3: Recursos utilizados

3.1. Colección de datos CABS120k08

La colección de datos empleada ha sido confeccionada por Michael G. Noll en el 2008 y se denomina *CABS120k08* [32]. Consiste en una gran investigación que contiene casi 120 mil URLs (117.434 para ser exactos) con metadatos adicionales, basados en la intersección del log de consultas de búsquedas de AOL³⁷ (llamado *AOL500k*), las categorías asignadas en el *Open Directory Project*, los marcadores sociales de *Delicious.com*, los *anchor text*³⁸ de los enlaces y la popularidad de cada página representado por Google³⁹ *PageRank* [46]. Este dataset está disponible para poder descargarse desde Internet en el sitio web personal del autor⁴⁰.

El triunvirato de los metadatos

En [32] se define este dataset como un “triumvirato” de metadatos, compuesto por anotaciones sociales (tags), anchor texts (de los enlaces) y search queries (búsquedas web).

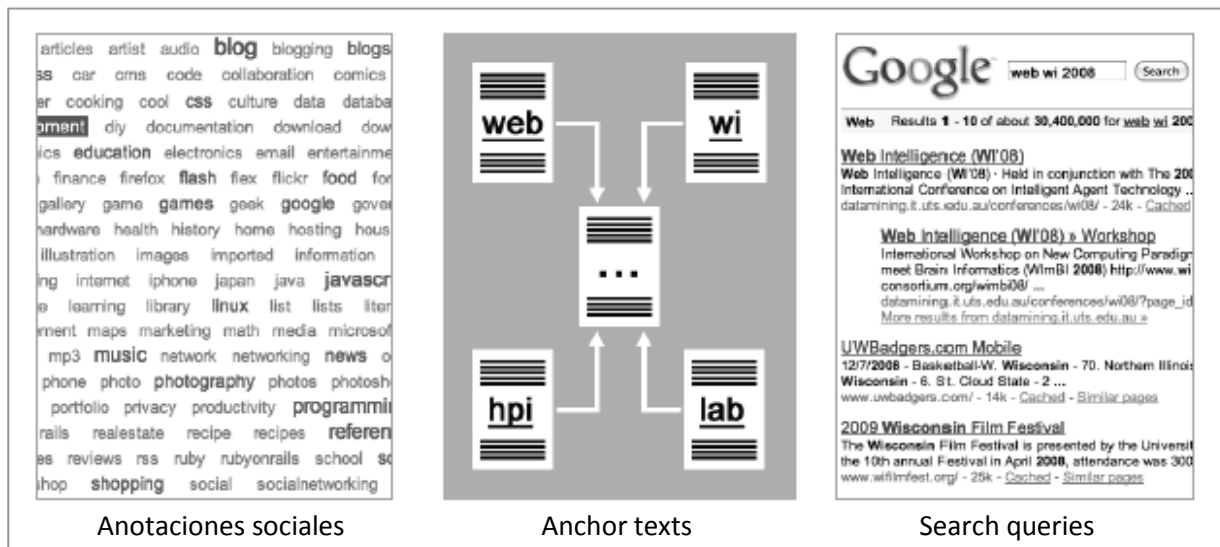


Figura 3.1. Triunvirato de metadatos.

El primer integrante de este triunvirato son las **anotaciones sociales**, que ya han sido explicadas en detalle en el Capítulo 2. Como segundo participante se presenta a los **anchor text** (también conocidos como *link label*) de los enlaces. Es el texto visible y cliqueable de un hipervínculo en una página web. Son comúnmente empleados para obtener más información sobre los vínculos entre las páginas, como así también para mejorar la indexación y la clasificación en los motores de búsqueda. Un anchor text

³⁷ Anteriormente *America Online, Inc.* Empresa estadounidense proveedora de medios y servicios de acceso a Internet.

³⁸ También conocido como *link label*, es el texto visible y cliqueable en un hipervínculo de una página web.

³⁹ <http://www.google.com>

⁴⁰ <http://www.michael-noll.com/wiki/CABS120k08>

comúnmente tiene menos de 60 caracteres. La forma en que se visualizan depende del navegador web utilizado.

Por último, las **search queries** representan al tercer integrante del triunvirato. No son más que las consultas que realizan los usuarios en los motores de búsqueda tradicionales, con el fin de obtener resultados que satisfagan su necesidad. Las search queries efectuadas en la web tienen la particularidad de no ser estructuradas y con frecuencia ambiguas, varían mucho de los lenguajes de consulta estándar que se rigen por estrictas reglas de sintaxis. Las consultas web pueden ser empleadas con varios fines, como por ejemplo, determinar el perfil de los usuarios para luego personalizar los resultados generados.

Composición de CABS120k08

Como se explicó anteriormente pero ahora en mayor detalle, esta colección se compone de:

Un **muestreo al azar del log de consultas de AOL500k**, la cual es una de las más grandes colecciones públicas de consultas de búsqueda de hoy en día. Consiste de 20 millones de búsquedas web obtenidas a partir de 650.000 usuarios de AOL Search durante tres meses, en el año 2006. Como resultado, alrededor de 1,6 millones de documentos web diferentes fueron visitados por los usuarios.

Para cada página de CABS120k08, se obtuvieron las **categorías del Open Directory Project (ODP)**. También conocido como *DMoz* (por *Directory Mozilla*), es un proyecto colaborativo en el que editores voluntarios listan y categorizan enlaces a páginas web. Cualquier persona puede sugerir un enlace en una categoría determinada que luego ha de ser aprobada por un editor. En la actualidad este directorio ha logrado un gran auge debido a que Google lo está utilizando como su directorio principal de búsqueda. *ODP* es propiedad de *Nestcape Communications Corporation*. Se catalogan como “el más grande y comprensivo directorio web editado por humanos”. Cuenta con aproximadamente 4,8 millones de documentos web organizados en alrededor de 590.000 categorías.

A su vez, se extrajeron **los tags desde Delicious** para cada página web. Como se explicó en 2.2, este servicio permite guardar en Internet las páginas favoritas que clásicamente se almacenan en los navegadores, y categorizarlas con un sistema de etiquetado denominado *tags*. No sólo un usuario puede almacenar enlaces a sitios web preferidos, sino que también permite compartirlos con otros usuarios del servicio y determinar cuántos tienen un determinado enlace guardado en sus marcadores. Posee una API⁴¹ que permite desarrollar aplicaciones que trabajen con el contenido de su gran base de datos, la cual se ha usado para la obtención de los *tags* de distintos enlaces al momento de crear la colección de datos *CABS120k08*.

Para obtener los **anchor text de cada página web**, el autor construyó un crawler paralelo distribuido. Cada instancia de este crawler realizó consultas en Google para obtener todos los links hacia la página en cuestión (empleando el operador especial “*link:<URL>*”, por ejemplo

⁴¹ Ver Glosario.

link:www.unicen.edu.ar). A continuación, se descargaron las páginas de resultados de Google y se extrajeron los links que apuntan hacia cada documento web. Por razones técnicas, sólo se pudieron procesar las primeras 100 páginas de resultados con links entrantes para cada página web.

Finalmente, para medir la **popularidad de cada documento** en la WWW, el autor decidió emplear Google PageRank [46], en parte porque este algoritmo ha sido bien estudiado y existe abundante literatura sobre el tema, como así también por la asociación entre Google y AOL⁴². El algoritmo de PageRank analiza la estructura de links que existe en la web, donde entran en juego los anchor texts de cada vínculo (provistos por los “autores web”) pero no está relacionado con las anotaciones sociales que se hallan en Delicious (generadas por los “lectores web”). Existen estudios previos [17, 47] que afirman que el algoritmo de PageRank es un buen indicador de popularidad social de un documento.

Finalmente, por definición, cada documento en la colección ha sido buscado por lo menos una vez (en AOL) y se encuentra clasificado en al menos una categoría (en ODP).

3.1.1. Visión general de la colección

A continuación, la tabla 3.1 presenta un *overview* de las estadísticas obtenidas en la colección de datos *CABS120k08*.

Información	Total	Comentarios
Total de documentos	117.434	Documentos presentes en <i>CABS120k08</i>
Total de categorías	84.663	Categorías presentes en los documentos
Total de búsquedas	2.617.326	Búsquedas presentes en los documentos
Total de <i>anchor texts</i>	2.242.621	<i>Anchor-texts</i> presentes en los documentos
Total de usuarios	3.383.571	Usuarios de Delicious, contemplados en <i>CABS120k08</i>
Total de marcadores ⁴³	1.289.563	Enlaces que han sido agregados a <i>Delicious</i> y se encuentran en <i>CABS120k08</i> Unicos: 9.1%

⁴² AOL emplea el motor de búsqueda de Google.

⁴³ También llamado *bookmarks*. Páginas web que han sido agregadas en Delicious por los usuarios, sin tener en cuenta que se les haya colocado un tag o no.

Total de <i>tags</i>	3.383.571	<i>Tags</i> que han sido escritos en <i>Delicious</i> y se encuentran en <i>CABS120k08</i> Unicos: 26.3%
Documentos categorizados	117.434	100,0%
Documentos buscados	117.434	100,0%
Documentos con anchor-text	95.230	81,1%
Documentos con marcador	59.126	50,3%
Documentos taggeados ⁴⁴	56.457	48,1%

Tabla 3.1. Estadísticas generales de CABS120k08.

Las probabilidades estimadas basadas en estas observaciones se muestran en la tabla 3.2, donde las dos primeras filas exhiben el porcentaje de documentos que presentan ambos tipos de datos en CABS120k08, mientras que a continuación se muestran diversas probabilidades condicionales relacionadas siempre con la información que representa a cada documento.

$P(\text{marcadores} \cap \text{anchor text})$	46,7%
$P(\text{taggeados} \cap \text{anchor text})$	44,7%
$P(\text{marcadores} \mid \text{anchor text})$	57,5%
$P(\text{taggeados} \mid \text{anchor text})$	55,2%
$P(\text{anchor text} \mid \text{marcadores})$	92,7%
$P(\text{anchor text} \mid \text{taggeados})$	93,0%

Tabla 3.2. Probabilidades estimadas.

Como primera impresión, en la tabla 3.1. se puede apreciar la alta probabilidad de que un documento haya sido marcado –agregado a *Delicious*–, representado por un 50,3%, comparado con el 81,1% de que tenga al menos un *incoming hyperlink*⁴⁵ con *anchor text*. Esto es particularmente interesante ya que *Delicious* –desde donde se extrajo la información sobre marcadores sociales– comenzó a funcionar en el 2003 mientras que la web –donde a partir de la estructura se derivaron los datos de anchor text– ha estado presente por mucho más tiempo. Siguiendo la suposición de que los *incoming hyperlinks* indican que un documento web es percibido como “interesante” o “importante” para el recurso que tiene el enlace, los marcadores cubren un 57,5% (en tabla 3.2, $P(\text{marcadores} \mid \text{anchor text})$) de los documentos web relevantes del corpus.

⁴⁴ Páginas web que los usuarios han sido agregado en *Delicious* y a los que les han colocado tags (al menos uno).

⁴⁵ También conocidos como *backlinks*, son los enlaces que recibe una determinada web desde otras páginas. El número de *backlinks* determina el de páginas que lo enlazan a través de un vínculo (puede ser en texto o gráfico).

Por otro lado, los marcadores son los que los *incoming hyperlinks* a los lectores web, una forma de indicar que un documento es relevante. En la colección de datos, existe un número reducido de documentos web que han sido marcados pero no tienen ningún *incoming hyperlink* con *anchor text*. Como los *incoming hyperlinks* son mayormente creados por autores web humanos, esta observación indica que ese puñado de páginas pueden ser realmente útiles pero no han sido descubiertas aún por los autores webs. Este resultado es similar a [48] el cual afirma que los marcadores sociales pueden ser empleados como un pequeño recurso para encontrar páginas web nuevas e indexadas, por ejemplo, páginas que no han sido descubiertas por los *web spiders*⁴⁶ automáticos.

Longitud

La longitud de las *search query*, definida como el número de palabras ingresadas en una búsqueda, ha sido estudiada previamente y varía alrededor de los 2,X términos [49]. Es interesante comparar la longitud de las search queries con la “longitud” de los marcadores sociales (número de tags) y anchor texts (cantidad de palabras).

Según el análisis realizado en el dataset CABS120k08 [32], la longitud de las search queries, marcadores sociales y anchor texts es de 2,89, 2,49 y 2,43, respectivamente. Aunque como primera impresión puede parecer que las longitudes de los marcadores sociales y anchor texts son prácticamente iguales, se encontró que estas varían significativamente de acuerdo a la popularidad de los documentos web, como muestra la figura 3.2.

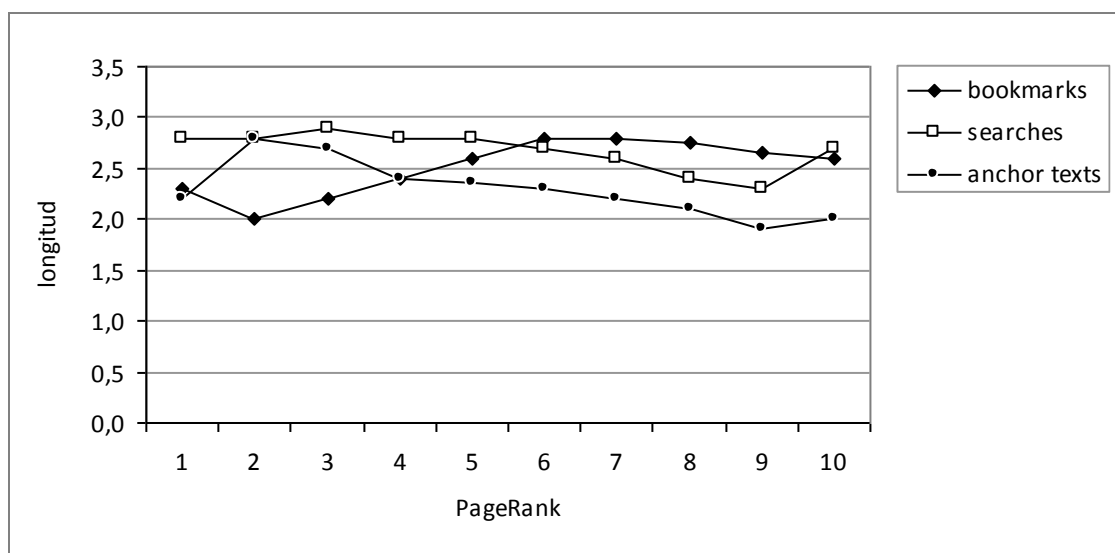


Figura 3.2. Longitud promedio de los marcadores sociales, search queries y anchor texts.

⁴⁶ Ver Glosario.

Existen correlaciones⁴⁷ negativas muy importantes para las search queries y anchor texts según la popularidad de los documentos. Spearman- r ⁴⁸ es -0,82 y -0,81, respectivamente. Por otro lado, hay una correlación positiva para los marcadores sociales: Spearman- r es +0,67. En la práctica, esto significa que los anchor texts proveen un gran volumen de datos para aquellos documentos web menos populares, mientras que los marcadores sociales hacen lo mismo pero con las páginas más conocidas, donde el punto de equilibrio se encuentra en PR⁴⁹ 4. Con respecto a las búsquedas web, la longitud promedio para las search queries es mayor que para los anchor texts a lo largo de todos los PageRanks. Comparándola con los marcadores sociales, la longitud de las search queries presenta un punto de equilibrio en PageRank 6 con los marcadores, y luego en PR 10.

Finalmente, se observa que la longitud promedio para los tres tipos de metadatos se encuentra alrededor de los 2 y 3 términos a lo largo de los distintos PageRanks. Parece ser que los usuarios prefieren usar sólo 2 o 3 términos por acción para los distintos problemas a afrontar (colocar un tag, crear un link o buscar en la web).

Novelty (novedad)

El autor analiza qué cantidad de “nuevos” datos son proporcionados por los marcadores sociales, anchor texts y search queries, con el objetivo de ver en qué grado pueden ayudar a la recuperación de información, por ejemplo. Se define *novelty* como el porcentaje de términos únicos que no se encuentran actualmente en el documento. Los términos para los marcadores sociales son representados por un conjunto único de tags que representa todas las veces que se taggeó ese documento; por ejemplo, si muchos usuarios agregaron el término “research” a un documento, sólo es tenido en cuenta una vez. Los términos para los anchor texts y search queries son definidos de manera similar. Cada documento es representado por un conjunto de palabras únicas, extraídas del *HEAD title*⁵⁰, *META keywords y description*⁵¹ y *BODY*⁵².

⁴⁷ En probabilidad y estadística, la correlación indica la fuerza y la dirección de una relación lineal entre dos variables aleatorias. Se considera que dos variables cuantitativas están correlacionadas cuando los valores de una de ellas varían sistemáticamente con respecto a los valores homónimos de la otra: si tenemos dos variables (A y B) existe correlación si al aumentar los valores de A lo hacen también los de B y viceversa. La correlación entre dos variables no implica, por sí misma, ninguna relación de causalidad.

⁴⁸ La correlación de Spearman (r_s) es una medida de relación lineal entre dos variables. Se diferencia de la correlación de Pearson en que utiliza valores medidos a nivel de una escala ordinal. Si alguna de las variables está medida a nivel de escala de intervalo/razón deberá procederse antes de operar el estadístico a su conversión en forma ordinal.

⁴⁹ Abreviación de PageRank.

⁵⁰ Tag de HTML donde se define el título del documento.

⁵¹ Tag de HTML donde el autor especifica un conjunto palabras claves relacionadas con el documento, invisibles para un visitante normal pero que son de gran valor para los navegadores web y distintos motores de búsqueda.

⁵² Tag de HTML que define el contenido propio de una página web (ej. texto, imágenes, video, etc.) que es interpretado por cada navegador y visualizado para los usuarios normales.

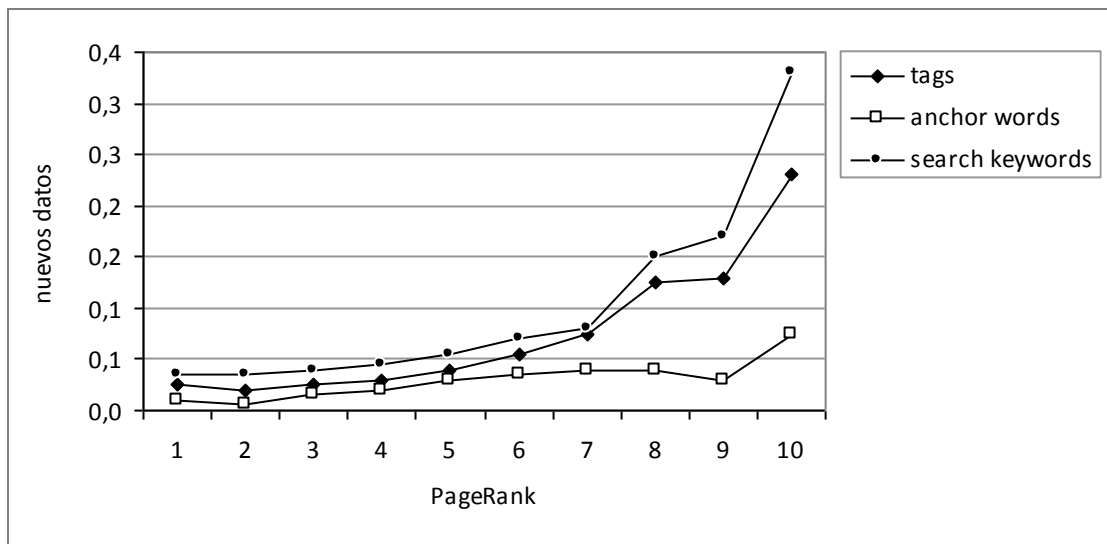


Figura 3.3. Porcentajes de nuevos datos provistos por los tags de los documentos, palabras de los anchor text y términos de búsqueda.

En la figura 3.3 se observa que la cantidad de nueva información es relativamente baja. Para los tres tipos de datos, este valor se encuentra por debajo del 6% para casi un 90% de los documentos del dataset *CABS120k08* (PR 0-5). Es interesante ver que las curvas que representan los *anchor text* y los *términos de búsqueda* muestran un comportamiento similar: ambas crecen con la popularidad de los documentos (a partir del PR 6). También se descubre que los tags proveen un mayor volumen de nuevos datos en comparación con los anchor texts. Esto indica que los tags son un mejor recurso para descubrir nueva información, particularmente para aquellos documentos populares.

Como conclusión, se afirma que la mayoría los metadatos disponibles provistos por los marcadores sociales, los anchor texts y las search queries sólo agregan una pequeña cantidad de nueva información a los documentos web. Esta observación es similar a los resultados obtenidos en [48], donde se descubre que un motor de búsqueda es poco probable que obtenga grandes beneficios de los tags provenientes de los marcadores sociales, si es que ya cuenta con acceso al contenido del documento.

Similitud

Se ha estudiado la relación de pares que existe entre los marcadores sociales, los anchor texts y las search queries [32]; o sea, qué tan similar es cada tipo de datos con respecto a los demás. El autor empleó las categorías definidas en el Open Directory Project, con el objetivo de investigar si cada tipo de metadatos es adecuado para llevar a cabo tareas de clasificación, ampliando de una manera los estudios realizados en [50, 51, 52]. Se empleó *cosine similarity*⁵³ como medida de similitud. Los resultados se muestran en la tabla 3.3.

⁵³ Es una medida de similitud entre dos vectores de n dimensiones encontrando el coseno del ángulo entre ellos. Comúnmente usada para comparar documentos en text mining.

	T	A	S	C
T	x	0,126	0,126	0,189
A	0,126	X	0,193	0,103
S	0,126	0,193	X	0,102
C	0,189	0,103	0,102	x

Tabla 3.3. Similitud de pares de tags (T), anchor texts (T), search queries (S) y categorías (S).

La mayor similitud se da entre los tags y las categorías (0,189) así como entre los anchor texts y las queries (0,193). Esta comparación sugiere que los tags se adaptan mejor a tareas de clasificación, mientras que los anchor texts se destacan en las búsquedas web. Sin embargo, esto no significa que las anotaciones sociales no pueden mejorar las búsquedas web. En [53] se utilizaron tags exitosamente para desambiguar búsquedas web mediante la explotación semántica implícita extraída de las folcsonomías.

3.1.2. Formato de los datos

El corpus es un archivo XML con la siguiente estructura.

a. Documentos

Cada documento web está representado como un `<document>` con los atributos que se listan a continuación:

url	URL del documento web
categories	Número de categorías de <i>Open Directory</i>
searches	Número de búsquedas de AOL Search en <i>AOL500k</i>
inlinks	Número de <i>incoming hyperlinks</i> con <i>anchor text</i>
users	Número de usuarios de Delicious que han marcado esta URL (igual al número de marcaciones de esa URL en Delicious)
top tags	Número de <i>top tags</i> de Delicious para esa URL, por ejemplo los <i>tags</i> más populares (máximo: 10)
tags	Número de <i>tags</i> únicos en Delicious para esta

	URL
pagerank	Google PageRank para esta URL
<category> elements	Categorías de Open Directory (ver más abajo)
<top_tags> elements	Top <i>tags</i> de Delicious en detalle (ver más abajo)
<bookmark> elements	Marcadores de los usuarios de Delicious incluyendo <i>tags</i> (ver más abajo)

Tabla 3.4. Formato de datos de los documentos en CABS120k08.

b. Categorías

Cada `<document>` contiene uno o más `<category>` elements, lo cual indica las categorías en las que está asignada esa URL dentro del Open Directory Project.

name	Nombre completo de la categoría con sub-categorías separados por "/" ej. "top/sports/golf/courses"
-------------	---

Tabla 3.5. Formato de datos de las categorías en CABS120k08.

c. Search queries

Cada `<document>` contiene uno o más `<search>` elements, lo cual indica aquellas *search queries* de AOL500k donde ese documento fue visitado como resultado de la búsqueda.

query	Las palabras usadas en la consulta
aol500k_id	El ID anónimo del usuario en AOL500k
date	El día en que se realizó la búsqueda, con el formato YYYY-MM-DD
time	La hora en la cual se hizo la búsqueda, con el formato HH:MM:SS
rank	La posición del documento en el listado de resultados cuando fue clickeado

Tabla 3.6. Formato de datos de las *search queries* en CABS120k08.

d. Incoming anchor texts

Cada `<document>` contiene uno o más `<inlink>` elements, que indica los *incoming hyperlinks* con *anchor texts* para ese documento.

anchor_text	El <i>anchor text</i> para el <i>incoming hyperlink</i>
--------------------	---

Tabla 3.7. Formato de datos de los *incoming anchor texts* en CABS120k08.

e. Delicious top tags

Cada `<document>` puede tener hasta 10 `<top_tag>` elements, que representan los top *tags* de Delicious en detalle para la actual URL.

name	El nombre del tag, ej. "news"
count	Número de veces que el <i>tag</i> ha sido usado por los usuarios

Tabla 3.8. Formato de datos de los top tags de Delicious en CABS120k08.

Nota: El número de `<top_tag>` elements de una URL es igual al valor del atributo "*tags*" del documento (ver más arriba).

f. Marcadores de los usuarios de Delicious

Cada `<document>` puede tener uno o más `<bookmark>` elements, que representan los marcadores que los usuarios de Delicious le han indicado al documento.

user	Nombre del usuario de Delicious que ha marcado la URL
tags	Lista de tags separadas con comas de los tags que le ha puesto el usuario al marcador
date	Fecha de creación del marcador, con el formato YYYY-MM

Tabla 3.9. Formato de datos de los usuarios de Delicious en CABS120k08.

g. Ejemplo

A continuación se muestra un ejemplo de la colección de datos CABS120k08.

```
<documents>
<document url="http://www.edletter.org/" users="10" categories="1"
  searches="29" inlinks="36" top_tags="5" tags="9" pagerank="6">
  <category name="top/reference/education/journals" />
  <search query="united states preschool teachers and statistics"
    aol500k_id="807613" date="2006-03-23" time="18:31:58" rank="12" />
  <search query="nclb and kindergarten" aol500k_id="7516545" date="2006-03-
    12" time="16:58:12" rank="16" />
  <search query="harvard education letters" aol500k_id="2229594"
    date="2006-03-21" time="01:43:37" rank="4" />
  ...
  <inlink anchor_text="Harvard Education Letter" />
  <inlink anchor_text="Home" />
  <inlink anchor_text="Harvard Education Letter" />
  <inlink anchor_text="www.edletter.org/" />
  ...
  <top_tag name="education" count="5" />
  <top_tag name="newsletter" count="2" />
  <top_tag name="research" count="3" />
  ...
  <bookmark user="mohandas" tags="edumags" date="2005-07" />
  <bookmark user="selahl" tags="pedagogy, teaching" date="2005-12" />
  <bookmark user="lllnelson2004" tags="edl600, edl671" date="2006-02" />
  ...
</document>
</documents>
```

Figura 3.4. Ejemplo de colección de datos CABS120k08.

3.2. Weka

Weka (siglas de Waikato Environment for Knowledge Analysis, *Entorno para Análisis del Conocimiento de la Universidad de Waikato*) es un conocido software para aprendizaje automático y *data mining* escrito en Java y desarrollado en la Universidad de Waikato. WEKA es un software libre distribuido bajo licencia GNU-GPL⁵⁴.

3.2.1. Descripción

El paquete Weka contiene una colección de herramientas de visualización y algoritmos para análisis de datos y modelado predictivo, unidos a una interfaz gráfica de usuario para acceder fácilmente a sus funcionalidades. La versión original de Weka fue un *front-end* en TCL/TK para modelar algoritmos implementados en otros lenguajes de programación, sumado a utilidades para pre-procesamiento de datos desarrolladas en C para hacer experimentos de aprendizaje automático. Esta versión original se diseñó inicialmente como herramienta para analizar datos procedentes del dominio de la agricultura, pero la versión más reciente basada en Java (WEKA 3), que empezó a desarrollarse en 1997, se utiliza en diferentes y variadas áreas, en particular con finalidades docentes y de investigación.

A continuación, se mencionan los hitos principales:

- En 1993, la Universidad de Waikato en Nueva Zelanda inició el desarrollo de la versión original de Weka (en TCL/TK y C).
- En 1997, se decidió re-escribir el código en Java incluyendo implementaciones de algoritmos de modelado.
- En 2005, Weka recibe de SIGKDD (*Special Interest Group on Knowledge Discovery and Data Mining*) el galardón “*Data Mining and Knowledge Discovery Service*”.
- Puesto en el ranking de Sourceforge.net el 19 de mayo de 2008: 248 (con 1.186.740 descargas).

3.2.2. Puntos fuertes y débiles de Weka

Los puntos fuertes de Weka son:

- Está disponible libremente bajo la licencia pública general de GNU.

⁵⁴ GNU-GPL es una licencia creada por la Free Software Foundation a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

- Es muy portable porque está completamente implementado en Java y puede correr en casi cualquier plataforma.
- Contiene una extensa colección de técnicas para pre-procesamiento de datos y modelado.
- Es fácil de utilizar por un principiante gracias a su interfaz gráfica de usuario.

Weka soporta varias tareas estándar de *data mining*, especialmente, pre-procesamiento de datos, clustering, clasificación, regresión, visualización y selección. Todas las técnicas de Weka se fundamentan en la asunción de que los datos están disponibles en un archivo plano (*flat file*) o una relación, en la que cada registro de datos está descrito por un número fijo de atributos (normalmente numéricos o nominales, aunque también se soportan otros). Weka también proporciona acceso a bases de datos vía SQL gracias a la conexión JDBC (*Java Database Connectivity*) y puede procesar el resultado devuelto por una consulta. No puede realizar *data mining* multi-relacional, pero existen aplicaciones que pueden convertir una colección de tablas relacionadas de una base de datos en una única tabla lista para ser procesada en Weka.

La principal carencia con la que cuenta Weka en la actualidad, es que sus algoritmos aún no cubren un área importante como es el modelado de secuencias.

3.2.3. Interfaces de Weka

Weka presenta cuatro posibles interfaces para trabajar: *Explorer*, *Experimenter*, *Simple CLI* y *Knowledge flow*, las cuales se explican a continuación.

Explorer

La interfaz *Explorer* (Explorador) dispone de varios paneles que dan acceso a los componentes principales como el panel *Preprocess*, que dispone de opciones para importar información de una base de datos, de un archivo CSV, etc., y para preprocesar estos datos utilizando los denominados algoritmos de *filtrado*. Estos filtros se pueden utilizar para transformar los datos (por ejemplo convirtiendo datos numéricos en valores discretos) y para eliminar registros o atributos según ciertos criterios previamente especificados. El panel *Classify* permite al usuario aplicar algoritmos de clasificación estadística y análisis de regresión (denominados todos *clasificadores* en Weka) a los conjuntos de datos resultantes, para estimar la exactitud del modelo predictivo resultante, y para visualizar predicciones erróneas, curvas ROC, etc., o el propio modelo (si este es susceptible de ser visualizado, como por ejemplo un árbol de decisión). El panel *Associate* proporciona acceso a las reglas de asociación aprendidas que intentan identificar todas las interrelaciones importantes entre los atributos de los datos. El panel *Cluster* da acceso a las técnicas de *clustering* o agrupamiento de Weka como por ejemplo el algoritmo K-means. Este es sólo una implementación del algoritmo expectación-maximización para aprender una mezcla de distribuciones normales. El panel *Selected attributes*

proporciona algoritmos para identificar los atributos más predictivos en un conjunto de datos. El panel *Visualize* muestra una matriz de puntos dispersos (*Scatterplot*) donde cada punto individual puede seleccionarse y agrandarse para ser analizados en detalle usando varios operadores de selección.

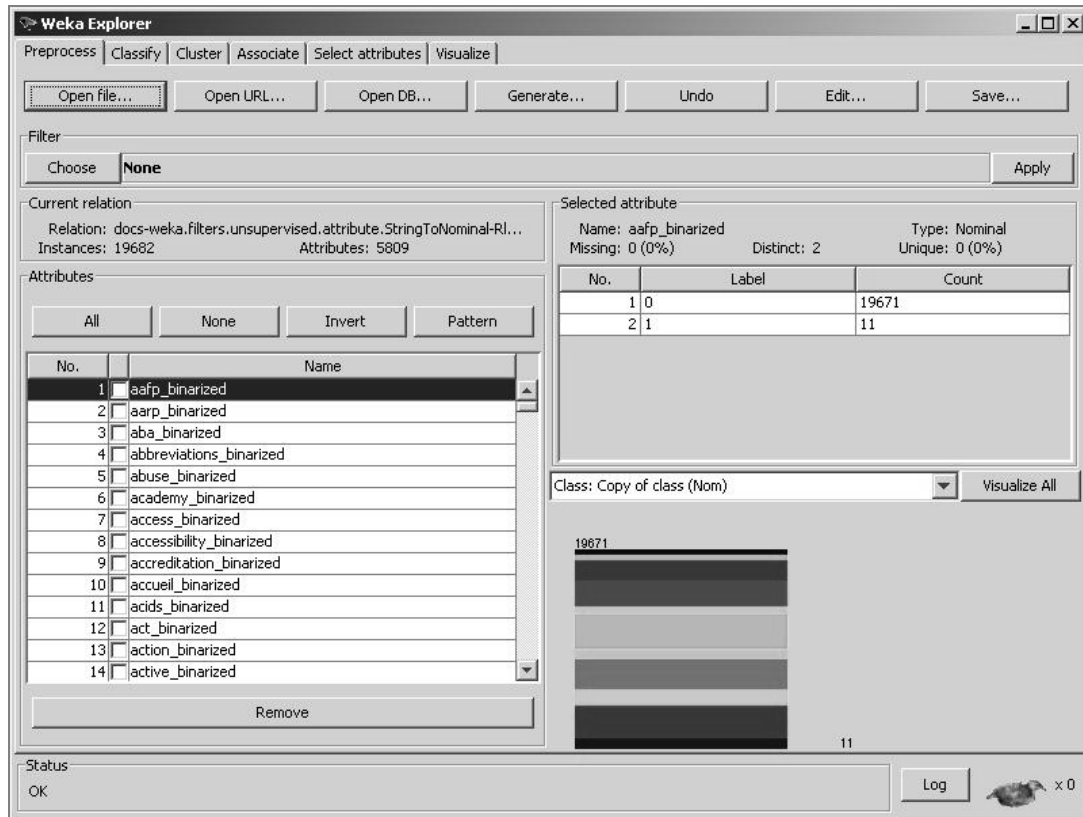


Figura 3.5. Captura de pantalla de la interface Explorer de Weka.

Experimenter

La interfaz *Experimenter* (Experimentador) permite al usuario crear, ejecutar, modificar y analizar experimentos de una manera más conveniente en comparación con el procesamiento de esquemas individuales. Por ejemplo, es posible crear un experimento que corra varios esquemas contra una serie de datasets y luego analizar los resultados para determinar si uno de los esquemas es (estadísticamente) mejor que los demás. Puede ser ejecutado mediante línea de comandos usando Simple LCI⁵⁵.

El *Experimenter* existe en dos variantes: *Simple*, con una sencilla UI que provee la mayoría de la funcionalidad necesaria para casi cualquier experimento. Y *Avanzada*, que permite un acceso completo a todas las capacidades del *Experimenter*. En ambos casos es posible configurar experimentos *estándares*, que son ejecutados localmente en una sola máquina; o experimentos remotos, que se encuentran distribuidos en distintos servidores. La distribución de los experimentos por un lado reduce el tiempo necesario para su ejecución, pero por otro requiere más trabajo en su instalación.

⁵⁵ Abreviatura de *Simple Command-Line Interface* (Interfaz Simple de Línea de Comandos); se trata de una consola que permite acceder a todas las opciones de Weka desde línea de comandos.

Simple CLI

Simple CLI es una abreviación de Simple Client⁵⁶. Esta interfaz proporciona una consola que permite introducir comandos de Weka. A pesar de tener una apariencia muy simple, es extremadamente potente ya que permite realizar cualquier operación soportada por Weka de forma directa. No obstante, es muy complicada de manejar ya que es necesario tener un conocimiento completo de la aplicación.

Esta interfaz no es muy utilizada a partir de la incorporación de *Explorer* o *Experimenter*. Actualmente, sólo es útil como herramienta de ayuda a la fase de pruebas.

Knowledge flow

Esta última interfaz de Weka es quizá la más cuidada y la que presenta de forma explícita el algoritmo interno del programa. Su funcionamiento es gráfico y se basa en crear un “circuito” que defina el experimento, a partir de la incorporación de distintos elementos base.

Por ejemplo, existen componentes que sirven como fuente de datos, desde donde se toman los archivos arff a clasificar; mientras que hay otros específicos para elegir el atributo que define la clasificación correcta (usualmente el último); sumado a los componentes que seleccionan el método de entrenamiento y el algoritmo de clasificación, etc. Todos estos componentes deben ser linkeados para crear el flujo de ejecución del experimento.

3.2.4. Formato de los archivos ARFF soportados por Weka

La estructura de un archivo con formato ARFF es muy sencilla. Para comenzar, veamos un ejemplo para luego explicarlo en detalle:

```
1 @relation weather
2
3 @attribute outlook {sunny, overcast, rainy}
4 @attribute temperature real
5 @attribute humidity real
6 @attribute windy {TRUE, FALSE}
7 @attribute play {yes, no}
8
9 @data
10 sunny,85,85,FALSE,no
11 sunny,80,90,TRUE,no
12 overcast,83,86,FALSE,yes
13 rainy,70,96,FALSE,yes
14 rainy,68,80,FALSE,yes
15 rainy,65,70,TRUE,no
```

⁵⁶ En español, *Cliente Simple*.

```
16 overcast,64,65,TRUE,yes
17 sunny,72,95,FALSE,no
18 sunny,69,70,FALSE,yes
19 rainy,75,80,FALSE,yes
20 sunny,75,70,TRUE,yes
21 overcast,72,90,TRUE,yes
```

Figura 3.6. Estructura de un archivo ARFF.

Como se ve en el ejemplo, este archivo se divide en tres secciones: @relation, @attribute y @data:

Sección 1: @relation <relation-name> (línea 1)

Todo archivo ARFF debe comenzar con esta declaración en su primera línea (no es válido dejarla en blanco). <relation-name> será una cadena de caracteres y si contiene espacios se debe colocar entre comillas.

Sección 2: @attribute <attribute-name> <datatype> (líneas de la 3 a la 7)

En esta sección se incluye una línea por cada atributo (o columna) que se vaya a incluir en el conjunto de datos, indicando su nombre y el tipo de dato. Con <attribute-name> se informa el nombre del atributo, que debe comenzar por una letra y si contiene espacios tendrá que estar entrecomillado. Con <datatype> se indica el tipo de dato para este atributo (o columna) que puede ser: numeric (numérico), string (texto), date [<date-format>] (fecha). En <date-format> se indica el formato de la fecha, que será del tipo "yyyy-MM-dd'T'HH:mm:ss". Por último, <nominal-specification>, que son tipos de datos definidos por el desarrollador y que pueden tomar una serie de valores cómo se indican (línea 3).

Sección 3: @data (a partir de la línea 9)

En esta sección se incluyen los datos propiamente dichos. Se separa con comas cada columna y todas las filas deben tener el mismo número de columnas, número que coincide con el de declaraciones @attribute que se añade en la sección anterior. Si no se dispone de algún dato, se coloca un signo de interrogación (?) en su lugar. El separador de decimales tiene que ser obligatoriamente el punto y las cadenas de tipo string tienen que estar entre comillas simples.

3.2.5. Modos de entrenamiento

Weka presenta cuatro modos de entrenamiento para ser empleados en cualquier algoritmo de clasificación. Estos se encuentran bajo la solapa denominada *Classify*, agrupados en *Test options*.

- *Use training set*: Con esta opción Weka entrenará el método con todos los datos disponibles y luego lo aplicará otra vez sobre los mismos.

- *Supplied test set*: Se basa en seleccionar explícitamente un archivo de datos con el que se probará el clasificador obtenido con el método de clasificación usado y los datos iniciales.
- *Cross-validation*: Consiste en realizar una validación cruzada estratificada del número de particiones dado (llamado *folds*, donde por defecto es 10). La validación cruzada consiste en: dado un número n se dividen los datos en n partes y, por cada una de ellas, se construye el clasificador con las $n-1$ partes restantes y se prueba con esa. Así por cada una de las n particiones.
Una validación cruzada es estratificada cuando cada una de las partes conserva las propiedades de la muestra original (porcentaje de elementos de cada clase).
- *Percentage splits*: Se define un porcentaje con el que se construirá el clasificador y con la parte restante se probará.

3.2.6. Algoritmos de clasificación

Weka cuenta con un gran número de métodos de clasificación y regresión listos para utilizar. Se encuentran agrupados, a grandes rasgos, en las siguientes familias:

- Bayes: Métodos basados en el paradigma del aprendizaje de Bayes.
- Funciones (*functions*): Métodos “matemáticos” (redes neuronales, regresiones, SVM⁵⁷, etc).
- Lazy: Métodos que utilizan el paradigma de aprendizaje perezoso, es decir, no construyen un modelo.
- Meta: Métodos que permiten combinar diferentes algoritmos de aprendizaje.
- Trees: Métodos que aprenden mediante la generación de árboles de decisión.
- Rules: Métodos que aprenden modelos que se pueden expresar como reglas.

En la presente investigación se van a utilizar los algoritmos de NaiveBayes (perteneciente a la familia de Bayes) y SMO (Funciones), muy empleados en la clasificación de texto, donde logran obtener buenos resultados. Existen diversos estudios que avalan la efectividad de estos clasificadores. Por ejemplo, [54] afirma que aunque la elección de la función de kernel es crucial para la mayoría de las aplicaciones SVM, para el caso de la clasificación de textos, las transformaciones basadas en la frecuencia de los términos presentan un mayor impacto en la performance de SVM. Por otro lado, [55] introduce un nuevo algoritmo de *active learning*⁵⁸ en SVM. En lo que respecta a Naive Bayes, [56]

⁵⁷ En español, *Máquinas de Vectores de Soporte*. Conjunto de algoritmos de aprendizaje relacionados con problemas de clasificación y regresión. Dado un conjunto de ejemplos de entrenamiento (de muestras) es posible etiquetar las clases y entrenar una SVM para construir un modelo que prediga la clase de una nueva muestra.

⁵⁸ Los algoritmos de machine learning, en lugar de utilizar un conjunto de entrenamiento seleccionado al azar, tienen acceso a un pool de instancias sin etiquetar pudiendo solicitar las etiquetas para un cierto número de ellas.

presenta una interpretación novedosa sobre el excelente desempeño de este clasificador; y [57] propone soluciones simples para algunos de los problemas existentes de este algoritmo.

Según [56], **Naive Bayes** es uno de los algoritmos de aprendizaje inductivos más eficientes y rápidos que existen. Formalmente, es un clasificador probabilístico simple, basado en la aplicación del teorema de Bayes con una fuerte hipótesis de independencia [58]. Una definición más descriptiva podría ser “modelo de características independientes”. Básicamente, un clasificador de Bayes asume que la presencia (o ausencia) de una característica específica de una clase no está relacionada con la presencia (o ausencia) de cualquier otra característica. Por ejemplo, una *fruta* puede ser considerada como una *manzana* si es roja, redonda y tiene 10 cm de diámetro. Aunque estas características dependan entre sí o de la existencia de otras, un clasificador Naive Bayes considera que todas estas propiedades contribuyen de manera independiente a la probabilidad de que esa *fruta* sea una *manzana*.

Dependiendo de la naturaleza exacta del modelo de probabilidad, los clasificadores de Naive Bayes pueden ser eficientemente entrenados en un entorno de aprendizaje supervisado. En un gran número de aplicaciones prácticas, la estimación de parámetros para los modelos de Naive Bayes emplean el método de la máxima verosimilitud; en otras palabras, uno puede trabajar con el modelo de Naive Bayes sin creer en la probabilidad bayesiana o el uso de cualquier método Bayesiano.

Las redes bayesianas son gráficos unidireccionales que usan probabilidades, específicamente el Teorema de Bayes, para ayudar a calcular la verosimilitud de los acontecimientos sobre la base de evidencias parciales. Para ello, basta con el siguiente ejemplo. Se supone una planta nuclear con un núcleo (*core*), la temperatura de ese núcleo (nodo “T”), la lectura del medidor de temperatura o la temperatura percibida (nodo “G”), la probabilidad de que el indicador de temperatura sea defectuoso (nodo “F_g”), la probabilidad de que la alarma se apague (nodo “A”) y de que la alarma esté defectuosa (nodo “F_a”). Es probable que la alarma se apague si la temperatura recibida es alta, y el indicador es más probable que falle si la temperatura también es alta. Esto se puede ver en el siguiente gráfico.

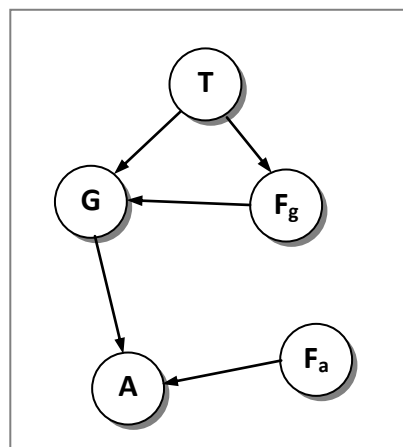


Figura 3.7. Red bayesiana del ejemplo de la planta nuclear.

Una ventaja de este clasificador, es que requiere una pequeña cantidad de datos de entrenamiento para la estimación de parámetros (medias y varianzas de las variables) necesarios para la clasificación.

Support Vector Machine (SVM) es un conjunto de algoritmos de aprendizaje propiamente relacionados con problemas de clasificación y regresión. Dado un conjunto de ejemplos de entrenamiento (de muestras) es posible etiquetar las clases y entrenar una SVM para construir un modelo que prediga la clase de una nueva muestra. Intuitivamente, una SVM es un modelo que representa a los puntos de muestra en el espacio, separando las clases por un espacio lo más amplio posible. Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, en función de su proximidad pueden ser clasificadas a una u otra clase. [7]

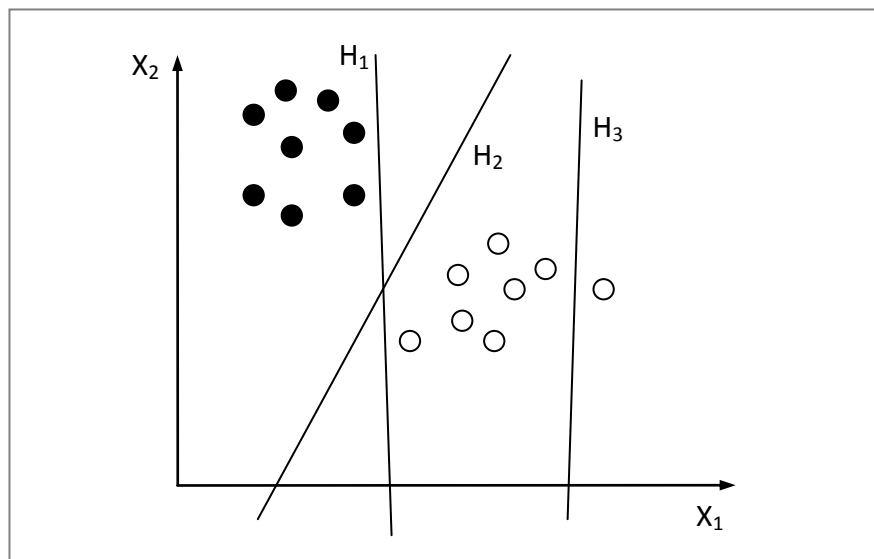


Figura 3.8. Conjunto de muestras separadas en distintos grupos o categorías (H_1 , H_2 y H_3) a partir de aplicar SVM.

Más formalmente, una SVM construye un hiperplano o conjunto de hiperplanos en un espacio de dimensionalidad muy alta (o incluso infinita) el cual puede ser utilizado en problemas de clasificación o regresión. Una buena separación entre las clases permitirá una clasificación correcta.

Dado un conjunto de puntos en el que cada uno pertenece a una de dos posibles categorías, un algoritmo basado en SVM construye un modelo que predice a qué categoría pertenece un nuevo punto. Los datos de entrada (los puntos) son vistos como un vector p -dimensional (una lista de p números). La separación en categorías se realiza mediante la construcción de un hiperplano N -dimensional que separa de forma óptima los datos.

El objetivo final de un algoritmo SVM es encontrar el hiperplano óptimo que separe en dos grupos el contenido del vector, tomando como punto de separación una variable predictora. De esta forma, los puntos del vector que se encuentren situados dentro de los márgenes de una categoría de la

variable estarán a un lado del hiperplano y los casos que se encuentren en la otra categoría estarán al otro lado.

3.2.7. Indicadores en la clasificación

Dentro de las opciones de clasificación con las que cuenta Weka, existe la posibilidad de obtener tanto el modelo construido como los distintos indicadores de la performance del clasificador empleado. Para ello, la opción *Output model* debe estar activa. La información aquí presentada es muy valiosa ya que permite analizar el algoritmo de clasificación desde otros puntos de vista, más allá de la exactitud con la que logra categorizar los documentos del dataset.

En la presente investigación se toman en cuenta para su análisis los siguientes indicadores: *Precision, Recall, F-measure* y *Error relativo absoluto*⁵⁹. La descripción y definición de cada uno de ellos puede encontrarse en el Anexo C.

Más adelante, se van a realizar pruebas con ambos clasificadores sobre distintos datasets, utilizando la configuración por defecto que ofrece cada uno, con el fin de deducir cuál es el que mejores resultados arroja. Finalmente se continuará trabajando con el elegido.

3.2.8. Salida generada

A continuación, se muestra cómo es la salida generada por Weka ante una clasificación:

```

=== Run information ===
Scheme:      weka.classifiers.bayes.NaiveBayes
Relation:     docs-weka.filters.unsupervised.attribute.StringToNominal-
Rlast-weka.filters.unsupervised.attribute.StringToWordVector-R1,2,3-W1000-
prune-rate-1.0-N0-stemmerweka.core.stemmers.NullStemmer-M1-
tokenizerweka.core.tokenizers.WordTokenizer -delimiters "
\r\n\t.,;:\'\"()?!"-weka.filters.unsupervised.attribute.Copy-R1-
weka.filters.unsupervised.attribute.Remove-R1-
weka.filters.unsupervised.attribute.NumericToBinary-unset-class-temporarily
Instances:    19583
Attributes:   4093
              [list of attributes omitted]
Test mode:    3-fold cross-validation

=== Evaluation on test split ===
=== Summary ===
Correctly Classified Instances      11173              57.0546 %
Incorrectly Classified Instances    8410              42.9454 %
Kappa statistic                    0.5099
Mean absolute error                 0.0811

```

⁵⁹ Abreviado de ahora en más como RAError, del inglés *Relative Absolute Error*.

Root mean squared error	0.2538						
Relative absolute error	50.7375 %						
Root relative squared error	89.7861 %						
Total Number of Instances	19583						
=== Detailed Accuracy By Class ===							
	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.419	0.009	0.732	0.419	0.533	0.959	621000
	0.694	0.067	0.667	0.694	0.68	0.89	400000
	0.487	0.028	0.604	0.487	0.539	0.859	1051780
	0.593	0.15	0.412	0.593	0.487	0.789	805301
	0.328	0.015	0.551	0.328	0.411	0.864	703200
	0.672	0.081	0.626	0.672	0.648	0.86	800520
	0.83	0.038	0.5	0.83	0.624	0.951	600020
	0.463	0.031	0.695	0.463	0.556	0.872	704310
	0.543	0.024	0.718	0.543	0.618	0.91	906014
	0.238	0.042	0.137	0.238	0.174	0.787	403010
	0.637	0.004	0.803	0.637	0.71	0.962	513000
Weighted Avg.	0.571	0.06	0.605	0.571	0.574	0.871	

Figura 3.9. Resultado entregado por Weka luego de finalizada una clasificación.

La primera parte de los resultados (*Run information*) hace mención a:

- Clasificador utilizado (*Scheme: ...NaiveBayes*)
- El dataset utilizado junto con todos los filtros que se han aplicado (*Relation: ...*).
- La cantidad de instancias con las que cuenta el dataset (*Instances: 19583*).
- La cantidad de atributos de las instancias del dataset (*Attributes: 4093*).
- El tipo de test seleccionado (*Test mode: Cross-validation con 3-folds*).

Mientras que la segunda parte presenta los resultados obtenidos. Por un lado, la sección *Summary* detalla la performance general de la clasificación:

- Instancias correctamente clasificadas, informando la cantidad y el porcentaje de las mismas (*Correctly Classified Instances: 11173 - 57,0546%*).
- Idem pero para las instancias mal clasificadas (*Incorrectly Classified Instances: 8410 - 42,9454%*).
- Estadísticas Kappa, que mide el acierto de predicciones con la clase actual. Esta estadística no es muy informativa ya que puede tener valores bajos incluso cuando hay valores altos de aciertos como en el caso de arriba (*Kappa statistic: 0,5099*).
- Error absoluto medio (*Mean absolute error: 0,0811*).
- Error cuadrático medio (*Root mean squared error: 0,2538*).

- Error absoluto relativo (*Relative absolute error: 50,7375%*).
- Error cuadrático relativo (*Root relative squared error: 89,7861%*).
- Cantidad total de instancias clasificadas, puede diferir de la cantidad inicial si encuentra que alguna instancia está mal formada (*Total Number of Instances 19.583*).

Y por último, la sección *Detailed Accuracy By Class* detalla los resultados entregados por las distintas métricas para la clasificación actual. En la figura 3.9 se puede apreciar que la última fila, denominada *Weighted Avg.*, es quien devuelve un promedio sobre cada indicador. Cabe aclarar que sólo se ha tenido en cuenta esta fila para el análisis de los indicadores a lo largo de la investigación.

3.3. Parser (CABS120k08 a formato ARFF)

En el presente trabajo, se ha construido un parser utilizando Java como tecnología, para transformar la colección de documentos de entrada CABS120k08 (léase Capítulo 3.1) en el formato empleado por la herramienta Weka (léase Capítulo 3.2).

A continuación, se presenta una vista de los módulos más representativos del parser. Luego, se detallarán cada uno de ellos.

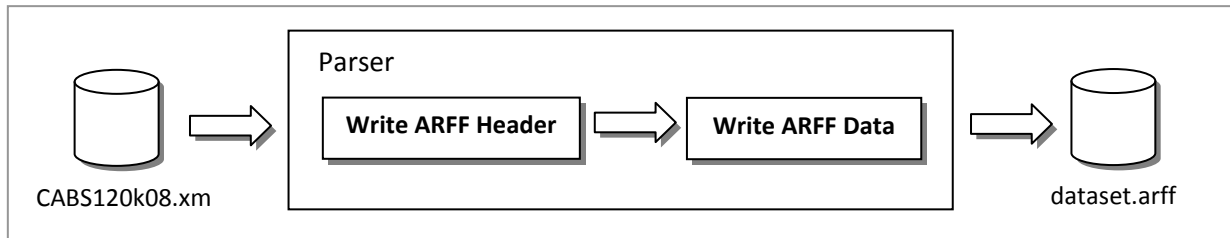


Figura 3.10. Vista de módulos del Parser.

Donde *CABS120k08.xml* representa la colección de datos inicial empleada, cómo se explicó en 3.1.

El procesamiento se divide en dos módulos principales: *Write ARFF Header* y *Write ARFF Data*. Cómo se menciona en 3.2.4, el formato de archivos que soporta la herramienta de clasificación consta de dos secciones bien definidas que son header y data. Es por ello que se crearon estos módulos dónde cada uno ataca de manera independiente la colección de entrada, generando el contenido correspondiente a estas secciones:

- *Write ARFF Header*: Simplemente obtiene la configuración inicial definida en el parser, y genera la primera parte del archivo ARFF, la cual consiste del nombre que se le da a la colección junto al listado de atributos mediante los cuales estarán definidos los documentos.
- *Write ARFF Data*: Este módulo es el que realiza la mayor parte del trabajo. Se encarga de leer los documentos contenidos en *CABS120k08.xml*, aplica distintos filtros –mencionados luego– y finalmente almacena la información en el archivo *dataset.arff*, que va a ser utilizado como entrada en la herramienta Weka.

3.3.1. Módulo *Write ARFF Header* del Parser

La tarea que realiza este módulo es relativamente sencilla. Consiste en generar las secciones *@relation* y *@attribute* presente en todo archivo ARFF. Como se detalla en 3.2.4, *@relation* consiste en una línea que representa el nombre del dataset en cuestión, mientras que *@attribute* incluye una línea por cada atributo (o columna) que se vaya a incluir en el conjunto de datos, indicando su nombre y tipo.

A continuación, se presenta un ejemplo de cómo sería el encabezado de uno de los dataset utilizado en el presente trabajo, más precisamente, el que incluye información tanto de las *queries* de AOL, *anchortext* y los *tags* de Delicious.

```
1 @relation sample
2
3 @attribute query string
4 @attribute anchortext string
5 @attribute tag string
6 @attribute class string
7 ...
```

Figura 3.11. Encabezado de un dataset que contiene los campos query, anchortext y tags.

Como se puede apreciar en la sección `@relation`, el nombre de este dataset es “sample”. Además, consta de cuatro atributos definidos en la sección `@attribute`, todos ellos de tipo string, llamados query, anchortext, tag y class. Este último, representa el código de la categoría asignada a cada documento. Es importante aclarar que por cuestiones de performance sólo se toma el primer nivel de la primera categoría asignada a cada documento (ver 3.1.3. b), de esta manera se evita la creación de un enorme dataset que luego tendría dificultades para ser ejecutado en Weka (ejemplo, problemas OutOfMemory⁶⁰ de la JVM⁶¹).

3.3.2. Módulo *Write ARFF Data* del Parser

Este módulo es el encargado de generar la sección `@data` perteneciente al dataset final. Como se explica en 3.2.4, `@data` incluye la información de los distintos documentos que luego serán utilizados por los clasificadores. La forma en que se escriben estos datos debe guardar consistencia con la declaración de los atributos de la sección `@attribute`.

Un ejemplo de un dataset generado se muestra a continuación, donde cada documento incluye información de las *queries* de AOL, *anchortext* y los *tags* de Delicious, seguidos por un código que identifica la categoría a la que ese documento pertenece dentro del dominio de *CABS120k08*.

⁶⁰ Error que se suele producir debido a que la JVM se queda sin memoria disponible para poder crear nuevos objetos.

⁶¹ Siglas de *Java Virtual Machine* (o Máquina Virtual de Java), es una máquina virtual de proceso nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el Java bytecode), el cual es generado por el compilador del lenguaje Java.

```

7 ...
8 @data
9 "shop mail free clinic", "site nacion phd", "chariti cancer", 3827
10 "store pop assault", "post dream sport", "ohio cincinnati", 6583
11 "aol online headlight", "site main", "car forum", 47363
12 "resident time arm", "austin lawn", "landscape forum", 8372
13 "plant power sugar", "inform india", "india", 3272
14 "pinture fish pic", "site fish list", "fish", 9473

```

Figura 3.12. Datos de cada documento en un dataset con los campos query, anchortext y tags.

Antes de describir en detalle el funcionamiento de *Write ARFF Data*, es mejor tener una noción de la arquitectura que se ha dispuesto para tal fin.

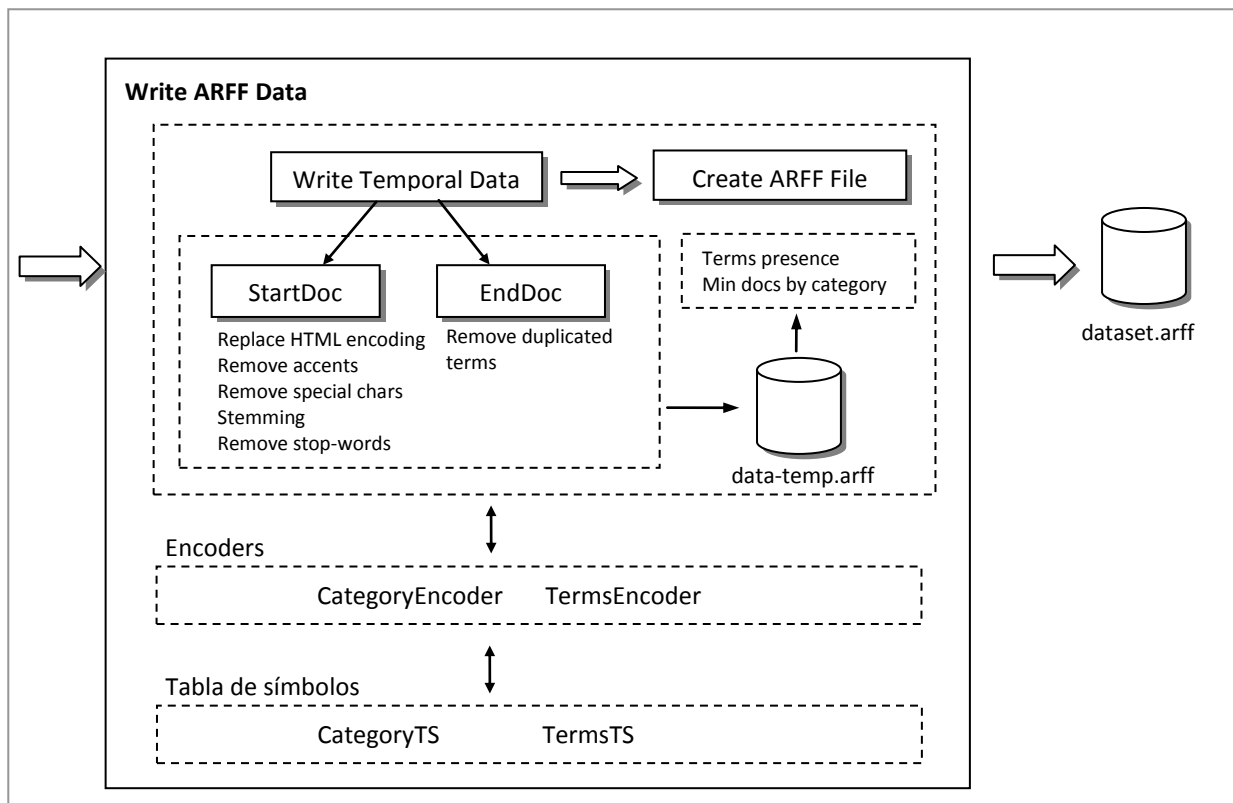


Figura 3.13. Arquitectura del módulo Write ARFF Data.

Básicamente, *Write ARFF Data* se compone de dos módulos principales denominados *Write Temporal Data* y *Create ARFF File*. El primero recibe como entrada la información de los documentos por parte de *CABS120k08.xml*, realiza un complejo procesamiento con los mismos y genera un dataset temporal denominado *data-temp.arff*.

A su vez, *Write Temporal Data* se puede volver a dividir en dos sub-módulos llamados *StartDoc* y *EndDoc*. El primero de ellos se encarga de tomar cada uno de los documentos de *CABS120k08* y aplicar distintos tratamientos, por ejemplo: reemplazar el código html, eliminar acentos, eliminar caracteres

especiales, etc. (ver 3.3.3). Mientras que el segundo módulo, *EndDoc*, es el responsable de eliminar los términos duplicados en cada tipo de cada documento y escribirlo físicamente en el dataset temporal.

Algo importante en este procesamiento es lo que se ha dado en llamar *Encoders*. Luego que *EndDoc* procesa cada uno de los documentos, interactúa con dos encoders denominados *CategoryEncoder* (lleva la cuenta de la cantidad de documentos que existen en cada categoría) y *TermsEncoder* (almacena la cantidad de ocurrencias de cada término en todos los documentos del dataset). Estos encoders son importantes en la etapa que se define a continuación.

Por último, *Create ARFF File* tiene la responsabilidad de generar el dataset final, en base al dataset temporal creado por *Write Temporal Data*. Para ello, recorre cada uno de los documentos en *data-temp.arff* y aplica dos tipos de filtros:

- 1) Considera sólo los documentos pertenecientes a las categorías que contengan un mínimo número de documento. Supongamos que una categoría tiene sólo 5 documentos. Este número no es suficiente ni confiable con fines de clasificación, por lo tanto se descarta en el presente análisis. La cantidad mínima de documentos que debe contener una categoría es configurada en el parser, siendo 100 el valor por defecto.
- 2) Para cada documento, elimina los términos que tienen una determinada mínima ocurrencia en todo el dataset. Para ser más claros, si un término tiene 3 ocurrencias en un total de 12.000 documentos, no tiene sentido utilizar ese término en la clasificación ya que no aporta información relevante. La cantidad mínima de ocurrencias permitidas es configurada en el parser, siendo 15 el valor por defecto.

3.3.3. Filtros aplicados al contenido de cada documento

Como se explicó anteriormente, el módulo *Write ARFF Data* del parser realiza cierto procesamiento de filtrado a la información contenida en cada documento del dataset. A continuación se detalla cada uno de los procesamientos que han sido implementados en Java:

- Reemplazar código html: Reemplazo de secciones de códigos html por simple código ASCII. Ejemplo: por un espacio en blanco, & por &, ` por a (similar para el resto de las vocales).
- Eliminar acentos: Reemplaza todas las vocales que tengan acentos por la misma vocal sin acento.
- Eliminar caracteres especiales: Existen muchos documentos con caracteres especiales (solo se considera a los caracteres A-Z y dígitos) que deben ser eliminados ya que entorpecen el trabajo del clasificador.

- *Stemming*: Es un método que se utiliza para reducir una palabra a su raíz, o mejor a un *stem* o lema. Se utilizó una implementación del algoritmo de Porter⁶² para el idioma inglés, ya que la mayoría de los documentos pertenecen a esta lengua.
- Eliminar *stop-words*: Remover los *stop-words* a partir de una lista de palabras en inglés. Los *stop-words* son palabras comunes como *a* y *the*, que no aportan información relevante a la clasificación, es por ello que se decide eliminarlas. Se cuenta con un diccionario con los 600 *stop-words* más conocidos del idioma inglés, que es el lenguaje utilizado en los documentos de CABS120k08.

El siguiente diagrama ilustra esta información:

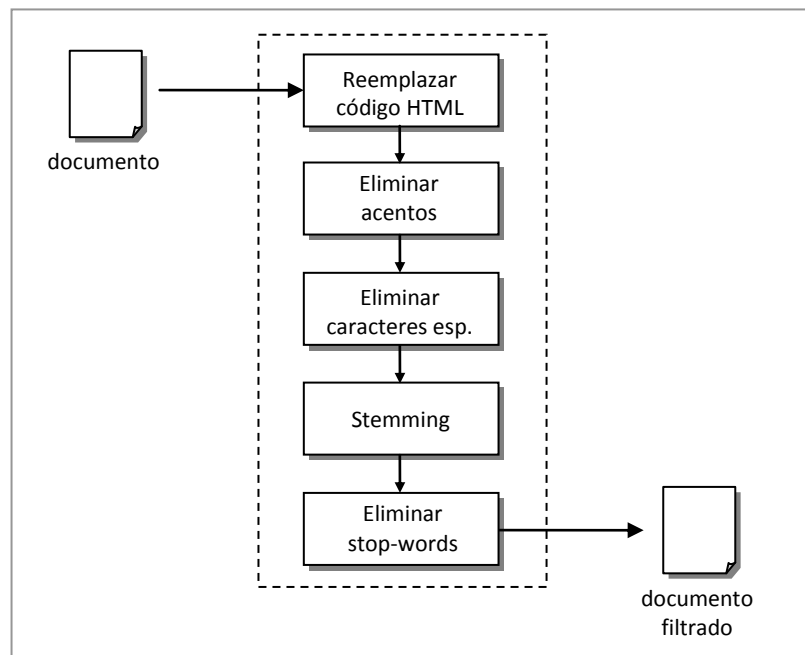


Figura 3.14. Filtrado aplicado al contenido de un documento.

⁶² Ver Anexo B.

Resumen del capítulo 3

En este capítulo se presentaron los recursos empleados a lo largo de la investigación. Primero, se describió la colección de datos CABS120K08 desde distintos puntos de vista. Por un lado, realizando una descripción del llamado “triunvirato de los metadatos”, compuesto por anotaciones sociales, anchor texts y search queries. Y por el otro, analizando la estructura, formato y fuente de datos del archivo CABS120k08.xml.

Luego, se describió la herramienta Weka, enumerando los dominios en los cuales se emplea junto a sus fortalezas y debilidades. Se explicaron las distintas interfaces (Explorer, Experimenter, Simple CLI y Knowledge flow), el formato de archivos soportado (ARFF) y los modos de entrenamiento y algoritmos de clasificación con el que cuenta (en especial, Naive Bayes y SVM).

Por último, se detalló el parser construido con la necesidad de convertir la colección de datos CABS120k08 al formato necesario por la herramienta Weka. Se analizó la arquitectura y responsabilidades de cada módulo, como así también las distintas clases de filtrado aplicados al contenido de los documentos.

Capítulo 4: Desarrollo de la investigación

4.1. Resumen

Utilizando el parser descrito en el capítulo anterior, se crearon distintos datasets con el fin de realizar diferentes experimentos, empleando siempre los algoritmos de clasificación provistos por Weka.

Estos experimentos llevan consigo un conjunto de etapas. La primera consta en realizar un pre-procesamiento a cada uno de los datasets antes de ejecutar los algoritmos de clasificación. Weka contiene muy variadas funciones de filtrado que se encargan básicamente de transformar los datos (ej. convertir datos numéricos en valores discretos) o eliminar registros o atributos según ciertos criterios especificados. Finalizado este filtrado, lo siguiente es ejecutar un algoritmo de clasificación que determine el comportamiento de cada dataset a la hora de categorizar automáticamente los documentos web. Como se verá más adelante, Weka ofrece distintas configuraciones y optimizaciones a la hora de ejecutar el clasificador. Se trata de utilizar variantes de ellas, para luego compararlas y obtener conclusiones.

El último paso consiste en analizar los resultados obtenidos para determinar cuál es la combinación de datasets que mejores rendimientos presenta. Siendo luego la idea de aplicar distintas técnicas de optimización al dataset obtenido, con el fin de mejorar aún más el grado de clasificación final.

4.2. Generación de los datasets

Como se explicó en 3.1, esta recopilación de documentos consta de información proveniente de las consultas de búsqueda de AOL (*queries*), los marcadores de Delicious.com (*tags*) y los *anchor-text* de los enlaces.

Con el fin de estudiar si los *tags* aportan mayor información a la clasificación en comparación con las fuentes de datos convencionales como *queries* y *anchor text*, se decidió generar distintos dataset que involucren los mismos documentos pero donde cada uno de ellos esté representado por una fuente de datos diferente, utilizando también la combinación de las mismas. En total, cada dataset estuvo compuesto por 19.583 documentos pertenecientes al conjunto de los casi 120 mil incluidos en CABS120k08. Se seleccionó este subconjunto ya que eran los únicos que contenían información proveniente de los tres recursos estudiados.

En resumen, los datasets que se generaron fueron:

- Dataset #1: sólo *queries*.
- Dataset #2: sólo *anchor-text*.

- Dataset #3: sólo *tags*.
- Dataset #4: combinación de *queries* y *anchor-text*.
- Dataset #5: combinación de *queries* y *tags*.
- Dataset #6: combinación de *anchor-text* y *tags*.
- Dataset #7: combinación de *queries*, *anchor-text* y *tags*.

Desde el punto de vista del parser, solo fue necesaria una modificación en la configuración, para que ante cada corrida del mismo le permita seleccionar los campos deseados para la generación del archivo ARFF.

A partir de este momento, cada paso siguiente se aplicó en forma paralela a los siete datasets distintos, comparando los resultados obtenidos y analizando las razones de tal comportamiento. Finalmente se decidió cuál es el dataset que mayor información aporta a la clasificación, para luego continuar estudiando la práctica de diversas optimizaciones al mismo.

4.3. Pre-procesamiento de cada dataset en Weka

Una vez generado cada uno de los dataset, el siguiente paso consistió en ejecutar los algoritmos de clasificación en Weka los cuales se encargan de analizar su efectividad. Para ello, fue necesario realizar un procesamiento a cada dataset obtenido por el parser, con el objetivo de convertirlo en el formato de entrada necesario para su clasificación.

A continuación, se muestra un ejemplo del dataset original (anchortext+tags en este caso) y más adelante se presenta cómo queda definido luego de aplicar los distintos procesamientos en Weka.

```
@relation docs

@attribute tag string
@attribute anchortext string
@attribute class string

@data
"site nacion phd fundacion visit investigacion org foundat research nation
para cancer sobr org/ ","chariti cancer ",621000
"abc post dream nightmar sport cincinnati ohio ","ohio cincinnati ",400000
"site rennlist squidootrad main ","car forum porsch ",1051780
"austin landscap rainbird mark bull vike","landscap sprinkler ",805301
"site bulldog fish relat qualiti fishi onli list fishyfish ","fish ",1051780
"inform ethanol link india ","india ",703200
"game board discount ","shop game boardgam ",800520
"asburi southsid johnni juke ","music band ",400000
"kilt children schedul sport athlet ","shop kilt cloth ",800520
...
```

Figura 4.1. Dataset ARFF original, antes de realizar cualquier procesamiento en Weka.

Vale aclarar que la herramienta Weka contiene lo que se denomina *algoritmos de filtrado*. Estos filtros pueden utilizarse para transformar los datos (por ejemplo, convirtiendo datos numéricos en valores discretos) o para eliminar registros o atributos según ciertos criterios previamente especificados.

A continuación, se detallan los pasos realizados para preprocesar cada dataset antes de ejecutarlo en el algoritmo de clasificación correspondiente.

1. Carga de la colección de datos original (representada por el archivo `dataset.arff`) en Weka. Después de cargarlo exitosamente, el sistema muestra algunas estadísticas como el número de atributos, su tipo y la cantidad de instancias (filas o documentos en la sección `@data`).

2. Elección del filtro *StringToNominal* y aplicarlo. Este filtro convierte un atributo de tipo cadena en un tipo nominal. Se aplica al atributo *class*, que es el que representa la categoría a la que pertenece cada documento.

3. Luego, se selecciona el filtro *StringToWordVector* y se lo aplica con los siguientes parámetros: `attributeIndices=1-3` (considerar todos los atributos a excepción de *class*), `useStoplist = true`, `wordsToKeep = 10000`, `tokenizer = WordTokenizer`, para el resto de la configuración se utiliza la que viene por defecto. Este filtro convierte los atributos de tipo String en un conjunto de atributos representando la ocurrencia de las palabras del texto.

4. Como Weka mueve el atributo *class* al segundo lugar, hay que llevarlo al final utilizando los filtros *Copy* y luego *Remove*, ya que no existe un filtro que realice esto automáticamente.

5. Finalmente, se aplica el filtro *NumericToBinary* a todos los atributos a excepción de *class*. Este filtro convierte los datos en formato numérico a binario. Si el valor de un dato es 0 o desconocido, el valor en binario resultante será 0.

6. Por último, se guardan los cambios en un nuevo archivo ARFF, que va a ser el utilizado para correr los algoritmos de clasificación.

El resultado de estos pasos es un dataset de Weka que utiliza una representación binaria de los documentos. A continuación se muestra un ejemplo de cómo quedó definido el dataset original de la Figura 4.1.

```

@relation 'docs-weka.filters.unsupervised.attribute.StringToNominal-Rlast-
weka.filters.unsupervised.attribute.StringToWordVector-R1,2-W1000-prune-
rate-1.0-N0-stemmerweka.core.stemmers.NullStemmer-M1-
tokenizerweka.core.tokenizers.WordTokenizer -delimiters \"
\\r\\n\\t.,;:\\\\'\\\\\\\"()?!\"-weka.filters.unsupervised.attribute.Copy-R1-
weka.filters.unsupervised.attribute.Remove-R1-
weka.filters.unsupervised.attribute.NumericToBinary-unset-class-temporarily'

@attribute aafp_binarized {0,1}
@attribute aarp_binarized {0,1}
@attribute aba_binarized {0,1}
@attribute abbrevi_binarized {0,1}
@attribute abus_binarized {0,1}
@attribute academi_binarized {0,1}
@attribute access_binarized {0,1}
@attribute accredit_binarized {0,1}
@attribute accueil_binarized {0,1}
@attribute ach_binarized {0,1}
@attribute achiev_binarized {0,1}
...
@attribute class
{621000,400000,1051780,805301,703200,800520,600020,704310,906014,403010,5130
00}

@data
{159 1,181 1,409 1,670 1,675 1,721 1,722 1,738 1,861 1,929 1,1077 1,3561 1}
{320 1,592 1,731 1,780 1,868 1,962 1,1007 1,1088 1,1177 1,1531 1,2222 1,5103
400000}
{161 1,408 1,609 1,929 1,963 1,2085 1,5103 1051780}
{95 1,128 1,146 1,217 1,242 1,488 1,505 1,606 1,619 1,799 1,1234 1,1405
1,1765 1,1916 1,2143 1,3504 1,3634 1,4176 1,5103 805301}
{586 1,818 1,848 1,929 1,1785 1,1892 1,2055 1,5103 1051780}
{510 1,518 1,585 1,2651 1,5103 703200}
{132 1,423 1,923 1,1853 1,4335 1,5103 800520}
{15 1,113 1,204 1,217 1,226 1,287 1,349 1,782 1,900 1,1119 1,2100 1,2224
1,5103 805301}
{668 1,1117 1,3374 1,5103 400000}
...

```

Figura 4.2. Dataset ARFF obtenido luego de realizar un procesamiento en su contenido a partir de los filtros que provee Weka.

Se pueden apreciar algunas diferencias en las distintas secciones de ambos datasets (Figura 4.1 y Figura 4.2). Por ejemplo, en @relation se modificó el nombre del dataset, generando uno nuevo que hace mención a todos los filtros aplicados. Además, en el nuevo dataset no existen más los dos primeros atributos (tag y anchortext), sino que ellos se encuentran mezclados sin distinción alguna y binarizados, o sea, su posible valor puede ser 0 o 1 (ausencia o presencia). El atributo class se define

como una colección finita de valores que puede tomar. Por último, en la sección @data, cada documento se detalla como los atributos que lo representan junto al class asignado.

4.4. Clasificación y Análisis de resultados

Luego de generar los distintos datasets, el siguiente paso consistió en clasificarlos en Weka con el objetivo de determinar cuál es la combinación que mejores resultados arroja (o menor cantidad de instancias mal clasificadas presenta). En esta investigación, se ha decidido utilizar *Cross-validation* y *Percentage splits* como modos de entrenamiento, ya que son los métodos más standards que existen para clasificar una colección de datos.

Por lo tanto, utilizando a manera de ejemplo los clasificadores *Naive Bayes* y *SMO* (con su configuración por defecto) y ambos métodos de entrenamiento, se está en condiciones de generar las distintas predicciones para cada uno de los siete datasets creados anteriormente.

Análisis de resultados

En todos los casos, primero se estudiaron los resultados de precisión (*accuracy*) con la configuración por defecto de *percentage splits* (66% del total de documentos se emplean como entrenamiento) y *cross-validation* (10 folds), para los algoritmos de clasificación *Naive Bayes* y *SMO*. Cabe aclarar que el último de ellos funciona con un kernel, siendo posible definir cuál: *PolyKernel* (kernel polinomial) y *RBFKernel* (*radial basis* o función básica radial). El primero de ellos permite modelar conjunciones en orden polinomial, además de ejecutarse de manera más rápida. Mientras que el segundo, realiza selecciones en círculos (o hiperesferas), logrando mejores resultados.

Utilizando el clasificador *Naive Bayes*, se muestran los resultados en la siguiente tabla, seguido por su representación gráfica.

Dataset (NaiveBayes)	Percentage split: 66% % instancias correctamente clasificadas	Cross-validation: 10 folds
query	42,20%	41,29%
anchortext	43,68%	45,00%
tags	55,17%	58,48%
query+anchortext	51,04%	49,92%
query+tags	53,91%	53,47%
anchortext+tags	57,92%	60,38%
query+anchortext+tags	57,51%	56,91%

Tabla 4.1. Resultados de aplicar el clasificador *NaiveBayes* con los valores por defecto de *percentage splits* y *cross-validation*.

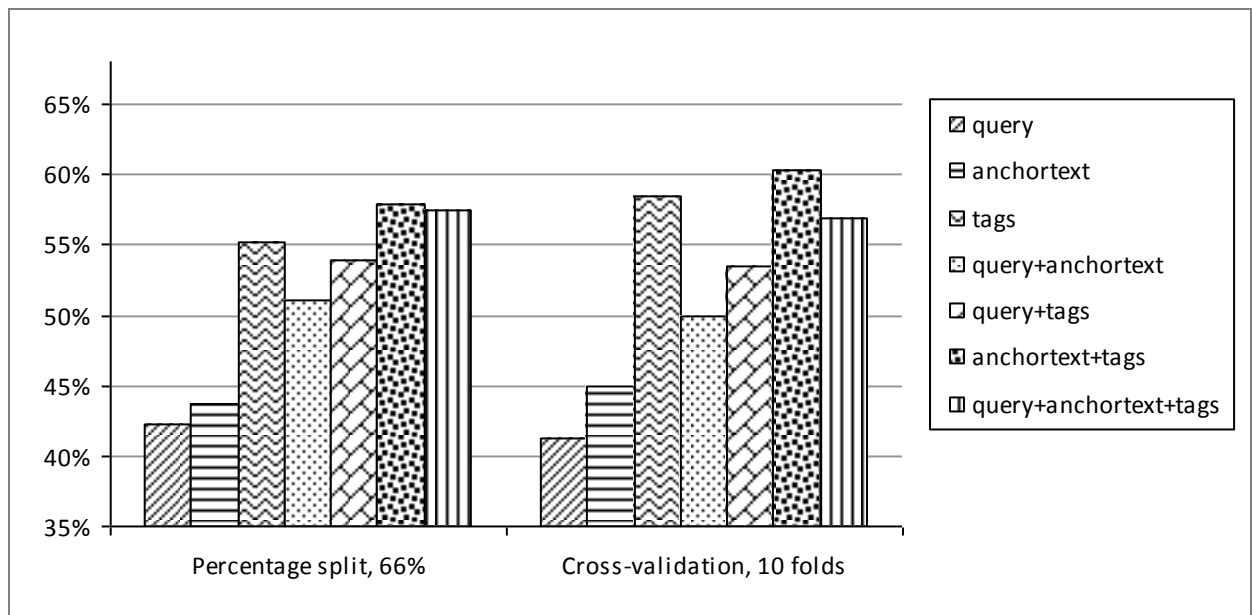


Gráfico 4.1. Representación de los resultados de clasificación utilizando NaiveBayes para 66% de instancias de entrenamiento y 10 folds, respectivamente.

A continuación se muestran los resultados obtenidos usando otros indicadores para la misma clasificación.

Dataset (NaiveBayes)	Precision	Recall	F-measure	RAError
Percentage split: 66%				
query	0,522	0,422	0,467	68,49%
anchortext	0,518	0,437	0,474	66,36%
tags	0,642	0,522	0,576	55,65%
query+anchortext	0,544	0,510	0,526	59,16%
query+tags	0,553	0,539	0,546	54,50%
anchortext+tags	0,599	0,579	0,589	49,51%
query+anchortext+tags	0,628	0,575	0,600	50,62%
Cross-validation: 10 folds				
query	0,539	0,413	0,468	68,78%
anchortext	0,539	0,460	0,496	63,92%
tags	0,648	0,585	0,615	52,36%
query+anchortext	0,553	0,499	0,525	59,72%
query+tags	0,610	0,535	0,570	54,82%
anchortext+tags	0,647	0,604	0,625	47,00%
query+anchortext+tags	0,609	0,569	0,588	50,88%

Tabla 4.2. Indicadores obtenidos a partir del clasificador Naive Bayes con valores por defecto de percentage splits y cross-validation.

Como se define en el Anexo C, se denomina **precisión** a la fracción de instancias correctas entre aquellas en las que el algoritmo *crea* que pertenecen al subconjunto relevante. Para el caso de Percentage split, el dataset compuesto solamente por tags obtiene el mayor grado de precisión con un

valor de 0,642 (léase 64,2%). ¿Qué significa esto? Que el algoritmo determinó que casi el 65% de la cantidad de los documentos que creyó que estaban bien clasificados... realmente lo estuvieron. Mientras que con Cross-validation los resultados variaron muy levemente. El dataset tags también logró el mayor valor de precisión con 64,8% seguido luego por anchortext+tags y query+tags.

En ambos casos, cabe destacar que los datasets compuestos por tags son los que logran un mejor grado de precisión en comparación con las otras fuentes de datos (query y anchortext).

Con respecto al análisis del **recall**, definido como la fracción de instancias correctas entre todas las que realmente pertenecen al conjunto relevante, se aprecia que para ambos modos de entrenamiento el dataset anchortext+tags es quien logra la mejor performance. Por ejemplo, empleando Percentage Split se obtiene que el total de los documentos correctamente clasificados que el algoritmo determinó representa un 57,90% del total de los documentos relevantes. En contraposición, el dataset compuesto solamente por el conjunto de datos query tuvo la actuación más pobre logrando algo más del 40% para este indicador.

Hasta el momento, la precisión y el recall nos dan un indicio de la importancia de las distintas fuentes de datos para la presente investigación.

El indicador **f-measure** es una medida de precisión de una prueba que tiene en cuenta tanto la precisión como el recall para realizar su cálculo. En el modo de entrenamiento percentage split, los dataset anchortext+tags y query+anchortext+tags son los que logran el mejor resultado como se puede apreciar en la tabla 4.2. Mientras que para el modo de cross-validation, nuevamente anchortext+tags es quien se destaca por sobre el resto, seguido de cerca en este caso por el dataset compuesto sólo por tags.

Por último, el **RAError** da un indicio de la calidad de la medida, calculado como el error absoluto total normalizado (dividiéndolo por el total del error absoluto del indicador simple). En la tabla 4.2 se puede lograr ver que el dataset compuesto solamente por tags es quien presenta la mayor tasa de error, con un valor que ronda el 68% en ambos modos de entrenamiento. Mientras que por otro lado, anchortext+tags alcanza algo menos del 50% de índice de error, convirtiéndolo en la colección de datos que mejor comportamiento consigue bajo este indicador.

También se puede ver que los datasets compuestos por tags son los que consiguen los valores más bajos de error, esto da una pista de la calidad que esta fuente de datos representa en la clasificación de documentos web.

De la misma forma que se realizó con NaiveBayes, pero ahora empleando el clasificador SMO con PolyKernel.

Dataset (SMO-PolyKernel)	Percentage split: 66% % instancias correctamente clasificadas	Cross-validation: 10 folds
query	44,47%	45,95%
anchortext	47,71%	48,11%
tags	63,74%	64,12%
query+anchortext	51,78%	53,20%
query+tags	60,76%	61,38%
anchortext+tags	64,34%	65,40%
query+anchortext+tags	61,91%	63,90%

Tabla 4.3. Resultados de aplicar el clasificador SMO (PolyKernel) con los valores por defecto de percentage splits y cross-validation.

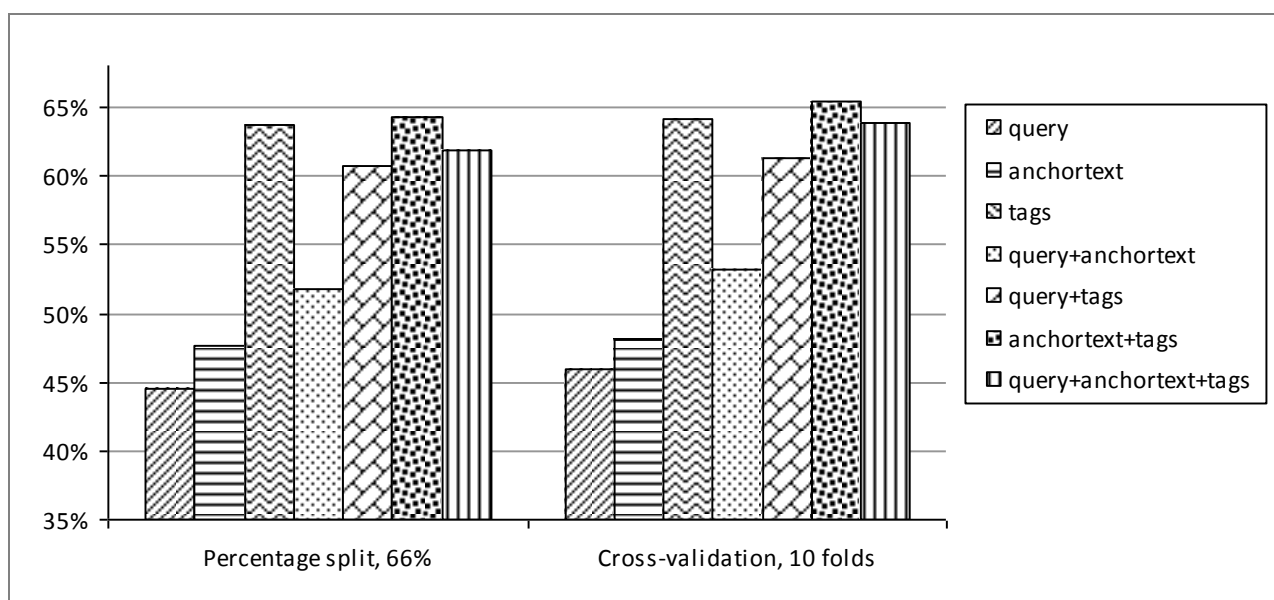


Gráfico 4.2. Representación de los resultados de clasificación utilizando SMO (PolyKernel) para 66% de instancias de entrenamiento y 10 folds, respectivamente.

En cuanto a los indicadores, para este caso sería:

Dataset (SMO-PolyKernel)	Precision	Recall	F-measure	RAError
Percentage split: 66%				
query	0,458	0,455	0,456	96,66%
anchortext	0,491	0,477	0,484	96,35%
tags	0,665	0,647	0,656	94,97%
query+anchortext	0,520	0,518	0,519	95,91%
query+tags	0,609	0,608	0,608	95,13%
anchortext+tags	0,648	0,643	0,645	94,89%
query+anchortext+tags	0,621	0,619	0,620	95,03%
Cross-validation: 10 folds				
query	0,477	0,460	0,468	96,53%
anchortext	0,516	0,504	0,510	96,14%
tags	0,698	0,686	0,692	94,76%
query+anchortext	0,548	0,543	0,545	93,99%
query+tags	0,639	0,635	0,637	93,89%
anchortext+tags	0,687	0,675	0,681	94,13%
query+anchortext+tags	0,651	0,649	0,650	93,32%

Tabla 4.4. Indicadores obtenidos a partir del clasificador SMO (PolyKernel) con valores por defecto de percentage splits y cross-validation.

Al igual que en el análisis realizado sobre la tabla 4.2, correspondiente a los resultados de los indicadores obtenidos para el clasificador NaiveBayes, los tags presentan un comportamiento similar. Por ejemplo, para los indicadores tanto de precisión como de recall y f-measure; los datasets tags, anchortext+tags y query+anchortext+tags son los que logran el mayor desempeño. Se sigue demostrando el potencial de este peculiar recurso de datos.

Por otro lado, el indicador RAError se comporta de igual manera que en el caso anterior, donde el dataset query es quien presenta la mayor tasa de error. Pero en este caso, este error alcanza un valor extremadamente alto, con algo más del 96% en ambos modos de entrenamiento, mientras que empleando el clasificador NaiveBayes no llega al 70% (ver tabla 4.2).

Por último, utilizando también SMO pero con RBFKernel.

Dataset (SMO-RBFKernel)	Percentage split: 66% % instancias correctamente clasificadas	Cross-validation: 10 folds
query	38,66%	40,26%
anchortext	36,07%	37,28%
tags	48,42%	50,38%
query+anchortext	43,78%	45,98%
query+tags	45,20%	47,29%
anchortext+tags	49,67%	51,48%
query+anchortext+tags	47,88%	49,52%

Tabla 4.5. Resultados de aplicar el clasificador SMO (RBFKernel) con los valores por defecto de percentage splits y cross-validation.

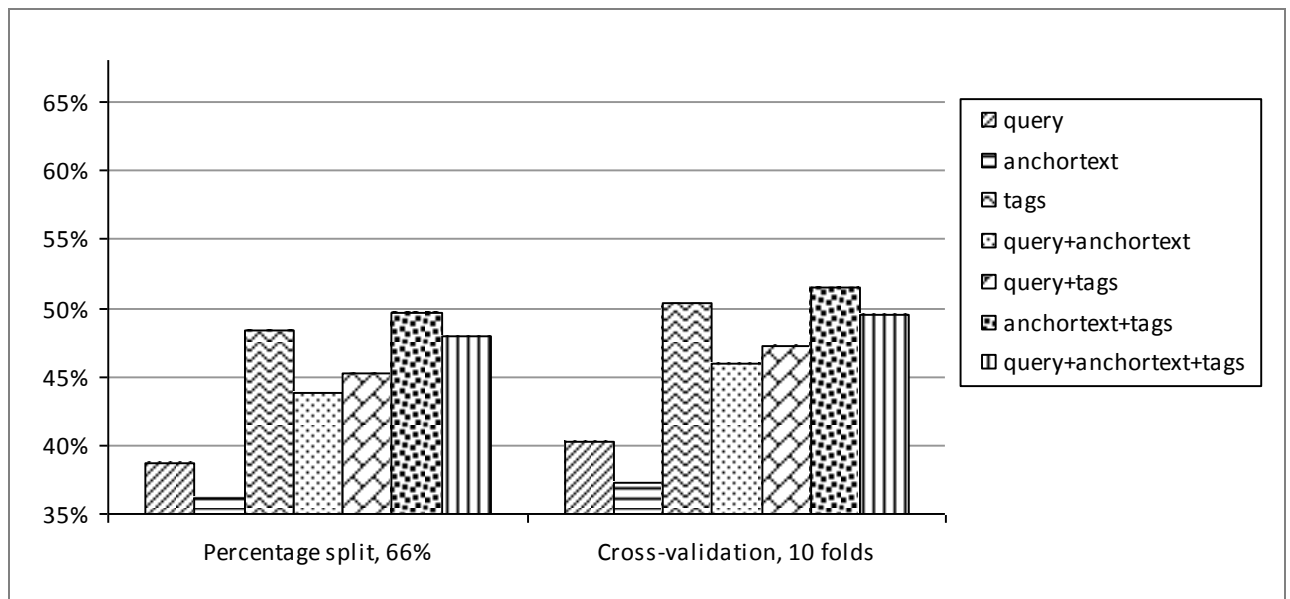


Gráfico 4.3. Representación de los resultados de clasificación utilizando SMO (RBFKernel) para 66% de instancias de entrenamiento y 10 folds, respectivamente.

Y en este caso, el resultado de los indicadores tiene la siguiente forma:

Dataset (SMO-RBFKernel)	Precision	Recall	F-measure	RAError
Percentage split: 66%				
query	0,571	0,387	0,461	97,75%
anchortext	0,469	0,361	0,408	97,75%
tags	0,644	0,484	0,553	96,62%
query+anchortext	0,533	0,483	0,507	96,50%
query+tags	0,623	0,533	0,574	95,87%
anchortext+tags	0,606	0,564	0,584	95,80%
query+anchortext+tags	0,620	0,595	0,607	95,39%
Cross-validation: 10 folds				
query	0,597	0,404	0,482	96,78%
anchortext	0,490	0,377	0,426	96,97%
tags	0,673	0,504	0,576	95,84%
query+anchortext	0,557	0,502	0,528	95,63%
query+tags	0,651	0,576	0,611	94,43%
anchortext+tags	0,633	0,588	0,610	94,85%
query+anchortext+tags	0,648	0,619	0,633	94,44%

Tabla 4.6. Indicadores obtenidos a partir del clasificador SMO (RBFKernel) con valores por defecto de percentage splits y cross-validation.

Estos indicadores se manifiestan de igual manera que los análisis llevados a cabo empleando los clasificadores NaiveBayes y SMO (PolyKernel), dónde se reafirma la importancia del recurso tags en la clasificación de documentos web.

Como se puede apreciar en las pruebas realizadas, SMO con PolyKernel es el que mejor resultado presenta, seguido luego por Naive Bayes, y por último SMO con RBFKernel. Por lo tanto, de ahora en adelante se trabajará con SMO (con PolyKernel) como el clasificador por defecto para el resto de la investigación.

En cuanto a los datasets empleados, el representado por anchortext+tags es el que mejor comportamiento presenta, utilizando tanto el clasificador Naive Bayes como así también SMO (en sus dos versiones). Sin embargo, esto no es suficiente, siendo todavía necesario experimentar con distintos porcentajes de entrenamiento empleando el modo *Percentage splits*⁶³. Los splits se utilizan para apreciar cómo evoluciona el número de instancias correctamente clasificadas cuando varía la cantidad de ejemplos de entrenamiento.

Por lo tanto, se analizaron los resultados obtenidos utilizando *Percentage splits* para distintos porcentajes de instancias de entrenamiento, que varían desde 5% hasta 90%, siempre clasificando con SMO con PolyKernel (de ahora en más, simplemente SMO). Cabe destacar que cada dataset está representado por un total de 19.583 documentos. Los distintos subconjuntos de documentos que se utilizan en el entrenamiento del clasificador contienen 980 instancias (para el 5% de documentos), 2.938 (15%), 4.896 (25%), 7.834 (40%), 9.792 (50%), 12.728 (65%), 14.687 (75%), 15.667 (80%), 16.646 (85%) y finalmente 17.625 (90%).

Dataset (SMO)	% instancias correctamente clasificadas - Percentage splits				
	5%	15%	25%	40%	50%
query	29,74%	36,65%	39,21%	42,09%	42,95%
anchortext	32,06%	38,95%	42,72%	45,37%	46,33%
tags	51,38%	58,59%	60,28%	62,28%	63,50%
query+anchortext	35,68%	42,97%	46,35%	49,16%	49,99%
query+tags	48,64%	54,40%	56,95%	59,08%	59,72%
anchortext+tags	48,83%	56,43%	60,35%	62,88%	63,13%
query+anchortext+tags	48,81%	55,13%	57,61%	59,54%	59,49%

Dataset (SMO)	65%	75%	80%	85%	90%
query	44,14%	45,73%	45,82%	46,85%	46,01%
anchortext	47,65%	48,46%	49,80%	50,25%	50,71%
tags	64,34%	65,09%	65,07%	65,67%	65,62%
query+anchortext	51,99%	51,42%	51,62%	52,09%	51,83%
query+tags	60,57%	61,39%	61,98%	61,86%	61,95%
anchortext+tags	64,94%	65,41%	66,05%	66,96%	66,70%
query+anchortext+tags	61,36%	62,72%	62,90%	62,95%	62,87%

Tabla 4.7. Resultados de aplicar el clasificador SMO a distintos porcentajes de documentos de entrenamiento para Percentage splits.

Como refleja la tabla 4.7, se observa que el dataset que mejores resultados obtiene en la clasificación corresponde al compuesto por anchortext+tags, logrando un **66,96%** de instancias

⁶³ Los resultados obtenidos con *Cross-validation* presentan el mismo comportamiento, por lo tanto se decidió utilizar un solo modo de entrenamiento (*Percentage splits*).

correctamente clasificadas utilizando un 85% de documentos de entrenamiento. Esta es la primera deducción analítica que se puede obtener. A manera de ayuda en la comprensión de estos resultados, se presenta el siguiente gráfico:

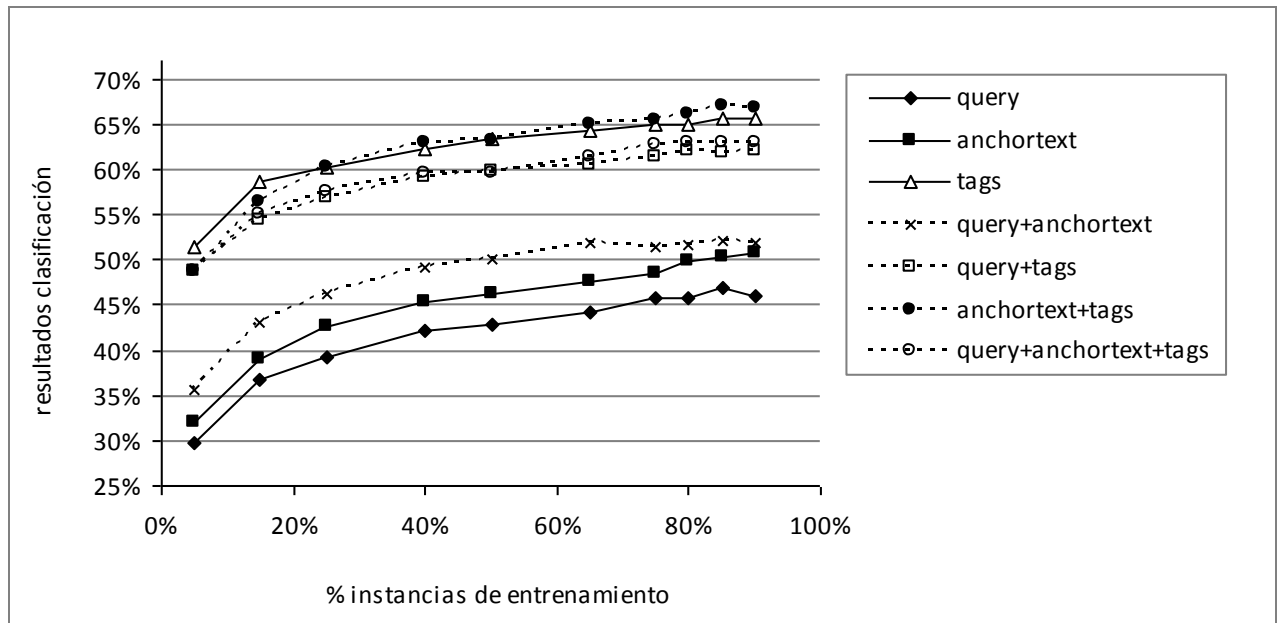


Gráfico 4.4. Representación de los resultados de clasificación utilizando SMO y distintos % de instancias de entrenamiento.

A simple vista se aprecia que cuando la cantidad (o porcentaje) de instancias de entrenamiento es baja, los resultados son pobres. Pero a medida que aumentan, se logran mejores clasificaciones llegando al umbral en el 85% de instancias. La razón de esto es simple: a medida que al clasificador se lo entrena con un mayor número de datos, éste puede comprender mejor el dominio del dataset con el que está trabajando y por lo tanto, realizar una mejor tarea (léase *mejorar el grado de clasificación*).

Tal como se afirmó anteriormente, el dataset anchortext+tags es el que mejores resultados presenta, seguido luego por la colección compuesta solo por tags y finalmente el constituido por query+anchortext+tags.

Además, se aprecia que los tags son el recurso que mayor información aporta a la clasificación, siendo anchortext el segundo más importante. Es por ello que la combinación de anchortext y tags genera los mejores resultados. Las queries son sólo un recurso que perjudica la categorización. Esto se puede apreciar, primero, ya que el dataset compuesto solamente por queries es el que peores resultados arroja. Segundo, si se toma cualquier dataset (no compuesto por queries) y luego se lo combina con estas, se logra una degradación en los resultados obtenidos. Por ejemplo, el caso anchortext+tags sumándole las queries (query+anchortext+tags) disminuye sus resultados notoriamente.

Los tags son realmente útiles en la clasificación, mayormente debido a que proveen información adicional que no está presente en los documentos mismos, la cual se encuentra directamente

relacionada en cómo las páginas web son percibidas por los mismos usuarios/visitantes. Por otro lado, según [32] las queries presentan mayor diversidad con respecto a los anchortext y los tags. La razón podría ser que las búsquedas en Internet son acciones más “al azar” con respecto a los demás recursos utilizados en la investigación (queries vs tags, anchortext). En cambio, los usuarios crean tags o links con anchortext sólo luego de haber leído el documento o haber percibido su utilidad. Este proceso parece servir como “filtro de ruido”, cosa que las queries carecen. Similarmente, los usuarios no sólo tienen problemas en encontrar información relevante en la WWW, sino que también tienen inconvenientes en formular buenas queries de búsqueda. Además, efectos adicionales como el hecho que los usuarios se acostumbren a la corrección ortográfica automática⁶⁴ por parte de los motores de búsqueda puede aumentar la diversidad de las consultas.

Hasta el momento se puede afirmar que los tags provenientes de Delicious influyen positivamente en la clasificación, tal como se pensó en la sección 1.3. Básicamente las razones son acertadas en lo que se ha discutido a lo largo del informe, destacando el hecho que los tags brindan información adicional sobre el contenido del recurso que no está presente en los documentos mismos.

A continuación, se decidió estudiar la aplicación de distintas mejoras lingüísticas con el fin de optimizar los resultados alcanzados hasta el momento. Esto se discute en el siguiente capítulo.

Resumen del capítulo 4

En el presente capítulo se realizaron distintos experimentos sobre el dataset CABS120k08, con el fin de determinar cuál es la combinación de metadatos (tags, anchor texts y queries) que mejor rendimiento presenta al ser procesada mediante un algoritmo de clasificación. A partir del parser construido anteriormente, se generaron siete distintos datasets (almacenados como archivos ARFF) empleando los mismos documentos, pero representados cada uno por una fuente de datos distinta, utilizando también la combinación de las mismas.

Luego, se realizó un cierto procesamiento a cada dataset con el fin de convertirlo en el formato de entrada necesario para su posterior clasificación. Para ello se aplicaron los algoritmos de filtrado de Weka.

Por último, empleando los métodos de entrenamiento Cross-validation y Percentage splits y los algoritmos Naïve Bayes y SMO, se clasificó cada uno de los datasets. Se obtuvo que una variante de SMO es el categorizador que mejor rendimiento logra, mientras que anchortext+tags es la combinación ideal de metadatos a procesar. Se apreció que los resultados entregados por el clasificador mejoran a medida que el número de instancias de entrenamiento aumenta. Además, se determinó que los tags son realmente útiles en la clasificación, debido a que proveen información adicional muy valiosa que no está presente en los documentos mismos.

⁶⁴ Se refiere a la opción *Did you mean* o *Usted quiso decir* que se puede encontrar en los motores de búsqueda más populares como Google (<http://www.google.com>), Yahoo! (<http://search.yahoo.com>) o Bing (<http://www.bing.com>).

Capítulo 5: Optimizaciones

5.1. Objetivo

El objetivo de la siguiente etapa es tomar el dataset que mejores resultados presentó anteriormente (*anchortext+tags*) e implementar una serie de cambios lingüísticos en la generación del mismo para lograr acrecentar aún más los resultados de la clasificación. Se toma como referencia el algoritmo de clasificación SMO empleando Percentage splits.

Se define como *baseline*⁶⁵ el dataset *anchortext+tags*. Los resultados obtenidos en 4.5 se muestran a continuación:

Dataset (SMO)	% instancias correctamente clasificadas - Percentage splits				
	5%	15%	25%	40%	50%
baseline	48,83%	56,43%	59,35%	61,88%	63,13%
Dataset (SMO)	65%	75%	80%	85%	90%
	63,94%	64,41%	64,61%	64,96%	64,55%

Tabla 5.1. Resultados de aplicar el clasificador SMO a distintos tamaños del conjunto de entrenamiento para el dataset *anchortext+tags*.

Se aplicaron distintas optimizaciones a este baseline. Para cada una de ellas se realizó la misma clasificación con el fin de estudiar si ese cambio al dataset original logra una optimización en los resultados.

5.2. Sin aplicar stemming

Como se explicó en 3.3.3, a cada uno de los dataset utilizados en 4.5 se aplicó el filtro de stemming⁶⁶, que básicamente la funcionalidad que tiene es la de reducir una palabra a su raíz. En este caso, se generó el mismo dataset pero sin aplicar stemming, por lo tanto no se realizó ningún tipo de modificación al contenido de la colección de documentos original. Tomando como referencia la figura 3.14, se presenta el siguiente diagrama:

⁶⁵ Del inglés *línea base*, es una revisión aprobada de un documento o archivo fuente, a partir del cual se pueden realizar cambios subsiguientes para luego compararlos con el original (el *baseline*).

⁶⁶ Ver Anexo B.

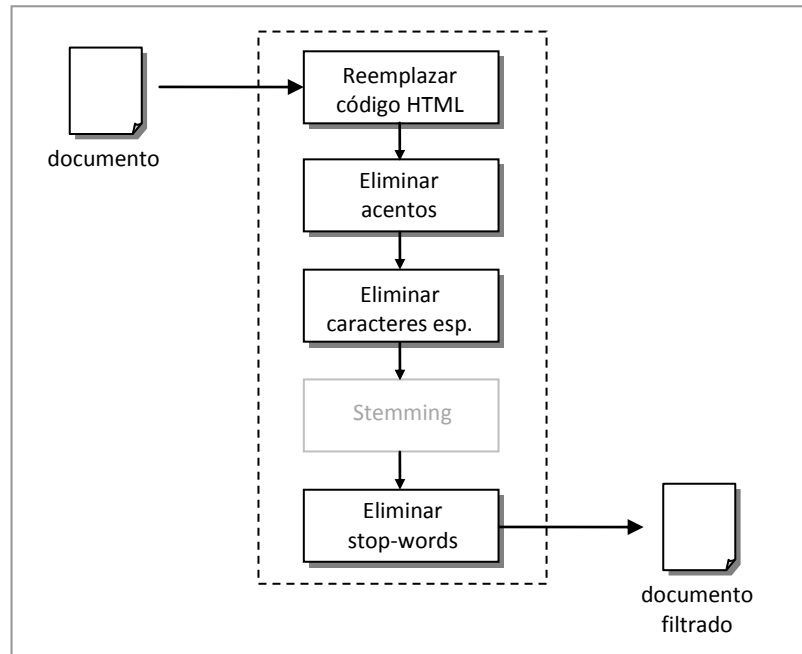


Figura 5.1. Filtrado aplicado al contenido de un documento, sin incluir Stemming.

Los resultados luego de aplicar SMO con Percentage splits fueron los siguientes (se muestran también los resultados del dataset tomado como baseline para facilitar la comparación):

Dataset (SMO)	% instancias correctamente clasificadas - Percentage splits				
	5%	15%	25%	40%	50%
baseline	48,83%	56,43%	59,35%	61,88%	63,13%
no stemming	50,26%	51,82%	52,27%	54,73%	55,89%

Dataset (SMO)	65%	75%	80%	85%	90%
baseline	63,94%	64,41%	64,61%	64,96%	64,55%
no stemming	57,02%	58,18%	59,39%	59,89%	59,21%

Tabla 5.2. Resultados de aplicar SMO a distintos tamaños del conjunto de entrenamiento para el dataset anchor-text+tags sin aplicar stemming.

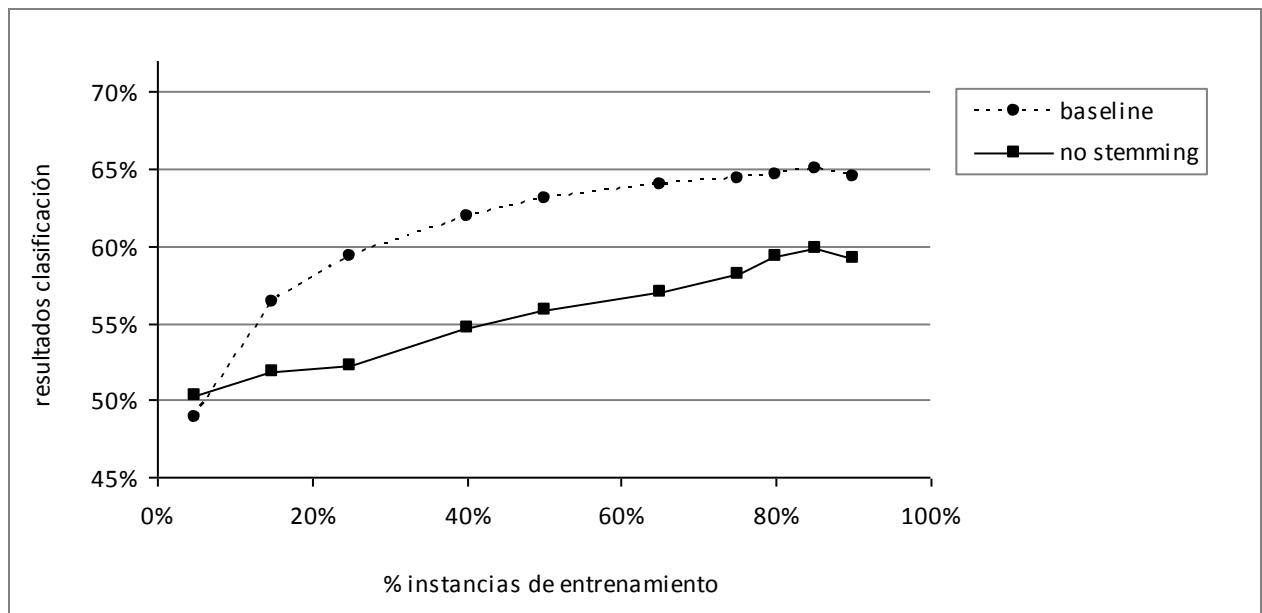


Gráfico 5.1. Representación gráfica de los resultados del clasificador SMO con distintos tamaños del conjunto de entrenamiento para anchortext+tags sin aplicar stemming.

Como se aprecia en la tabla, los resultados obtenidos para el mismo dataset pero sin aplicar stemming resultan más pobres que el original. La justificación es bastante directa. Mirando el contenido de ambos datasets, se encuentran casos como el siguiente:

Dataset <i>baseline</i> término (ocurrencias)	Dataset <i>anchortext+tags sin stemming</i> término (ocurrencias)
compute (53)	computer (28) compute (16) computadora (8) computation (1)

Tabla 5.3. Comparación de distintos términos antes y después de aplicar stemming.

¿Cuál es el significado? El dataset sin stemming contiene los términos “computer”, “compute”, “computadora” y “computation” con 28, 16, 8 y 1 ocurrencias respectivamente. Mientras que al aplicarse stemming en el dataset baseline, todos estos términos fueron resumidos a “compute”, resultando en 53 ocurrencias.

A la hora ejecutar el algoritmo de clasificación, SMO encuentra que “computer”, “compute”, “computadora” y “computation” son términos totalmente diferentes entre ellos (aunque a simple vista se note que están relacionados). Por lo tanto, se supone que el clasificador *aprendió* que aquellos documentos que contienen el término “computer” es muy probable que le corresponda la categoría “Computers”, ya que el elevado número de ocurrencias (28) de esta palabra sumado al resto de la información de los documentos, provoca que obtenga esa deducción. Pero cuando debe procesar el

término “computation” en un documento, debido a que sólo existe en una sola instancia dentro de las 19.583 que contiene el dataset, este no tiene la suficiente “fuerza” como para relacionarse con la categoría “Computers”, por lo tanto es probable que se le asigne alguna categoría errónea.

Mientras que por otro lado, en el dataset baseline, estos cuatro términos fueron reducidos al único término en común llamado “compute”. Entonces todos los documentos que contengan algunas de las palabras en cuestión van a verse representadas de igual manera en el dataset generado antes de procesarse por el algoritmo de clasificación. Es por ello que los resultados obtenidos fueron mejores.

Como punto negativo, pueden encontrarse casos de términos que semánticamente sean distintos pero al aplicarse stemming se unifiquen en un significado en común que para algunos de ellos puede no ser el correcto.

A continuación, se muestran los resultados obtenidos usando otros indicadores para el dataset *no stemming*, siempre realizando una comparación con el baseline.

Dataset (SMO)	Precision	Recall	F-measure	RAError
Percentage split: 66%				
baseline	0,648	0,643	0,645	94,50%
<i>no stemming</i>	<i>0,596</i>	<i>0,576</i>	<i>0,586</i>	<i>96,20%</i>

Tabla 5.4. Indicadores obtenidos a partir del clasificador SMO con valores por defecto de percentage splits para los dataset baseline y no stemming.

En esta tabla se aprecia que los indicadores precision y recall (y por consiguiente f-measure) presentan un mejor comportamiento en el dataset baseline en comparación con el generado sin aplicar stemming. Lo mismo ocurre con RAError, donde la tasa de error para la colección de datos tomada como base es menor (94,50% vs 96,20%), aunque todavía sigue siendo muy alta.

Conclusión: Se descartó la utilización de un dataset sin aplicar stemming, ya que como se aprecia en la tabla 5.2, presentó un porcentaje de instancias correctamente clasificadas inferior al obtenido con el dataset baseline.

5.3. Aplicación de sinónimos (WordNet)

Luego de inspeccionar el contenido de los datasets, se han encontrado casos que influyen negativamente en la clasificación y se suponen que pueden ser evitados. Se está hablando de la presencia de términos sintácticamente distintos pero que semánticamente significan lo mismo, o sea, *sinónimos*. Por ejemplo, los términos “earth” y “globe” son palabras diferentes pero a simple vista se deduce que se refieren a un mismo significado: el globo terráqueo. El clasificador interpreta estos términos como si fueran completamente distintos, ya que sólo se centra en su sintaxis. Este es uno de los tantos casos encontrados en el conjunto de datos.

Entonces la idea fue modificar el dataset baseline empleando un algoritmo que aplique sinónimos a los distintos términos de cada documento. De esta forma, lo que se intenta es no dejar escapar ese significado semántico al clasificador.

La herramienta que se usó para llevar a cabo este desarrollo fue WordNet⁶⁷ (ver *Anexo B: WordNet*), el cual es una enorme base de datos léxica del idioma inglés (se recuerda que el dataset está compuesto por documentos de esta lengua). WordNet agrupa las palabras en conjuntos de sinónimos llamados *synsets*, proporcionando definiciones cortas y generales, y almacenando las relaciones semánticas entre estos conjuntos de sinónimos. Esta herramienta ofrece una API en Java de libre uso, que se ajusta de manera perfecta al desarrollo de esta investigación.

La manera en que se aplican los sinónimos en la generación del dataset es la siguiente: en el procesamiento de cada término, como se explicó en 3.3.3, se incorpora la búsqueda de sinónimos mediante WordNet. Si efectivamente existen, se los incorpora a la información asociada a ese documento. O sea, si el término A presenta los sinónimos A1, A2 y A3, el documento ahora quedaría conformado por A, A1, A2 y A3, junto al resto de su contenido. Como paso siguiente, se aplica stemming y eliminación de stop words tanto para los términos originales como para todos sus sinónimos asociados. Por último, el clasificador procesa los términos del documento sin realizar ningún tipo de distinción alguno. El siguiente diagrama exhibe dónde se aplicó este nuevo proceso dentro del flujo de filtrado actual.

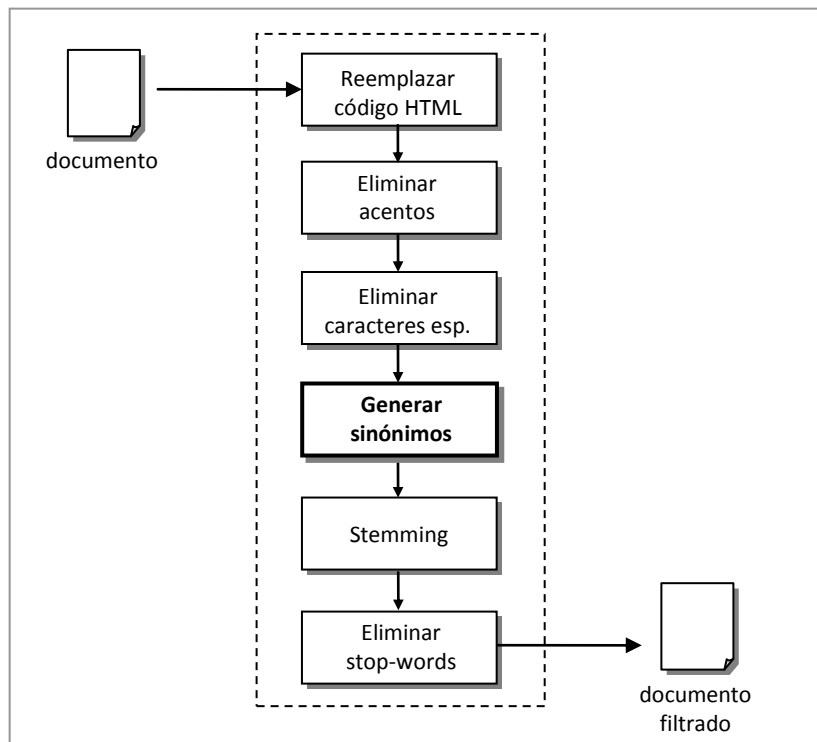


Figura 5.2. Filtrado aplicado al contenido de un documento, incluyendo la generación de sinónimos.

⁶⁷ Más información, consultar el sitio web oficial en <http://wordnet.princeton.edu>

A continuación, se muestran los resultados luego de aplicar SMO con Percentage splits para este nuevo dataset (también se listan los resultados del baseline para facilitar la comparación):

Dataset (SMO)	% instancias correctamente clasificadas - Percentage splits				
	5%	15%	25%	40%	50%
baseline	48,83%	56,43%	59,35%	61,88%	63,13%
<i>sinónimos</i>	50,32%	54,92%	55,13%	56,19%	56,22%

Dataset (SMO)	65%	75%	80%	85%	90%
	65%	75%	80%	85%	90%
baseline	63,94%	64,41%	64,61%	64,96%	64,55%
<i>sinónimos</i>	50,32%	54,92%	55,13%	56,19%	56,22%

Tabla 5.5. Resultados de aplicar SMO a distintos tamaños del conjunto de entrenamiento para el dataset anchortext+tags con sinónimos.

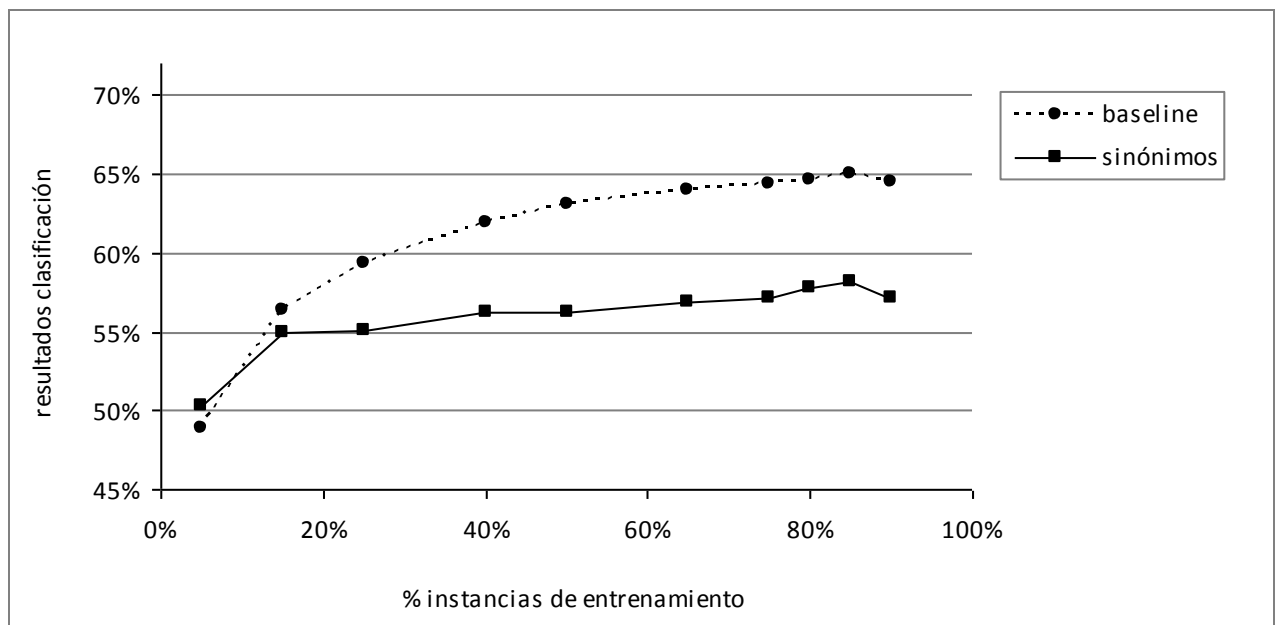


Gráfico 5.2. Representación gráfica de los resultados del clasificador SMO con distintas combinaciones de Cross-validation para anchortext+tags con sinónimos.

Claramente se puede apreciar que los resultados del nuevo dataset son peores que los del baseline. La mejor marca es de 56,22% de instancias correctamente clasificadas, empleando un 90% de documentos de entrenamiento.

¿Por qué los resultados empeoraron? Si bien hubo casos donde los sinónimos fueron ventajosos (ej. términos “earch” y “globe”), esta técnica provocó la incorporación de demasiada información para muchas de estas palabras. Por ejemplo, para el caso del término “computer”, los sinónimos encontrados fueron “computing machine”, “computing device”, “data processor”, “electronic computer”... lo cual provocó que el algoritmo de clasificación tenga un mayor volumen de datos a la

hora de trabajar, donde el porcentaje de presencias/ocurrencias de esos datos en el dataset no fue lo suficientemente grande como para tomar buenas decisiones a la hora de clasificar las instancias.

Al igual que en la tabla 5.3, comparación de algunos indicadores de baseline y el dataset sinónimos en este caso.

Dataset (SMO)	Precision	Recall	F-measure	RAError
Percentage split: 66%				
baseline	0,648	0,643	0,645	94,50%
sinónimos	0,588	0,565	0,576	97,15%

Tabla 5.6. Indicadores obtenidos a partir del clasificador SMO con valores por defecto de percentage splits para los dataset baseline y sinónimos.

El análisis es similar al realizado anteriormente en la tabla 5.3, donde los indicadores que se tienen en cuenta para el nuevo dataset son más bajos que los obtenidos con el baseline.

Conclusión: Se descartó la utilización de un dataset aplicando sinónimos sin tratamiento alguno. Como se ve en la tabla 5.5, los resultados de la clasificación obtenidos con este nuevo conjunto de datos son inferiores a los logrados previamente con el dataset baseline.

5.4. Aplicando spell check

Otra de las potenciales optimizaciones a realizar en el dataset baseline tiene que ver con los errores de ortografía encontrados en los documentos. Supongamos el ejemplo del término “photography” el cual ha sido escrito en un par de oportunidades como “photographi”. Como se puede ver, el segundo término es incorrecto. El problema que presenta este caso es que el algoritmo de clasificación interpreta ambos términos como si fueran distintos. Por lo tanto, el plan sería simplemente poder corregir cada una de las palabras que estén mal escritas en el dataset.

¿De dónde provienen los términos escritos erróneamente? En su mayoría del recurso de los tags. Tiene cierta lógica ya que los marcadores sociales representan una descripción de los documentos realizada por los mismos usuarios, que muchas veces escriben con errores de ortografía o bien utilizan ciertos modismos (principalmente americanos, los cuales no están definidos en ningún tipo de diccionario sino que forman parte del lenguaje coloquial). En el caso de los anchor-texts no se encontraron demasiados errores. Recordemos que estos últimos son generados por los editores de las páginas webs para el resto de los usuarios (a diferencia de los tags, que son creados por y para los usuarios, generalmente), lo que supone que tienen mayor cuidado a la hora de escribir.

A nivel diseño, la solución empleada fue aplicar un algoritmo de *spell check*⁶⁸ a cada uno de los documentos del dataset. Lo que hizo fue chequear que cada una de las palabras de la colección de datos esté escrita correctamente. De no ser así, fue reemplazada por el término correcto sugerido por el algoritmo. En el caso que no exista ninguna palabra correcta (se puede deber a que el término no se halla en el diccionario), entonces se ignora y se elimina del dataset. Se modificó el procesamiento explicado en 3.3.3 de la siguiente manera:

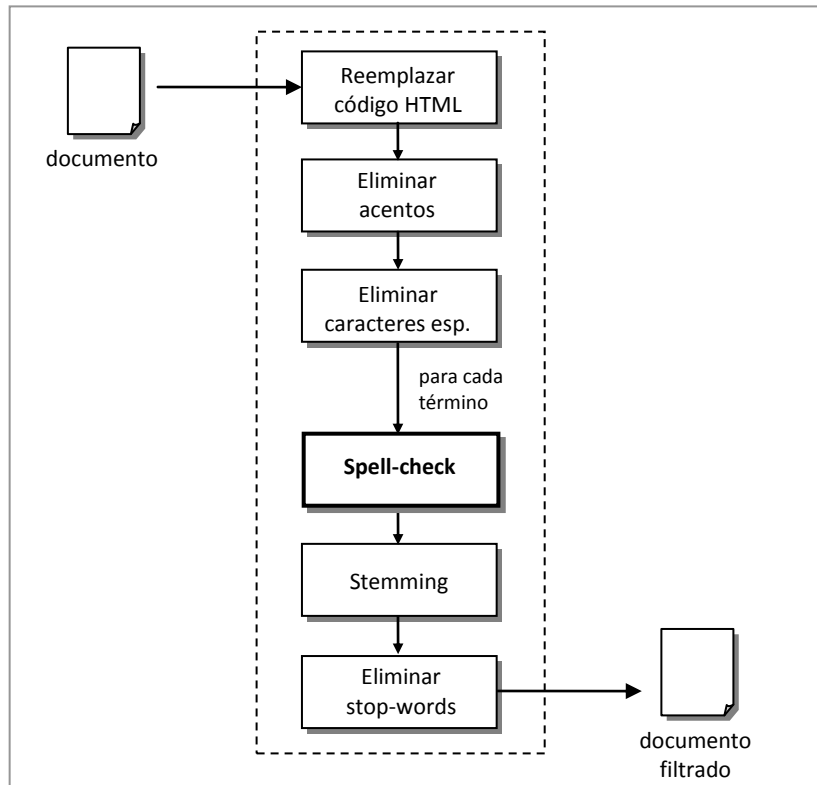


Figura 5.3. Filtrado aplicado al contenido de un documento, incluyendo un spell checker.

Desde el punto de vista técnico, existen distintas librerías Java que brindan este servicio. Si bien todas se basan en el mismo principio y trabajan de manera muy similar, varían en la implementación de cuestiones que tienen que ver con el análisis de similitud que realizan entre los distintos términos, como así también el hecho que cada una cuenta con su propio diccionario. Es por ello que se decidió probar con alguna de ellas para luego comparar su comportamiento en cuanto a los resultados obtenidos por el clasificador. Cabe destacar que en todos los casos se trabajó con la configuración por defecto provista por cada spell-checker.

La primer librería de spell checker es la provista por *Tumba!*⁶⁹. Esta permite trabajar con distintos diccionarios (inglés, español, portugués, italiano, etc). Su funcionamiento es bastante sencillo, una vez cargado el idioma con el que se desea operar, la librería recibe un término y retorna la palabra más similar del diccionario (puede ser exactamente la misma que recibió, si está escrita correctamente)

⁶⁸ Corrector ortográfico.

⁶⁹ Tumba! fue un motor de búsqueda optimizado para la web portuguesa. Se ha dejado de dar soporte en 2009.
<http://xldb.fc.ul.pt/wiki/Tumba!>

usando *Distance Levenshtein*⁷⁰, *Phonetic similarity*⁷¹, *Keyboard Proximity*⁷² y otras heurísticas de medición de similitud.

A continuación, se muestran los resultados de la clasificación utilizando SMO y Porcentaje splits, al igual que los casos anterior. También se listan los resultados del baseline, para facilitar la comparación.

Dataset (SMO)	% instancias correctamente clasificadas - Percentage splits				
	5%	15%	25%	40%	50%
baseline	48,83%	56,43%	59,35%	61,88%	63,13%
spell check Tumba	52,23%	58,15%	61,56%	63,17%	64,01%

Dataset (SMO)	65%	75%	80%	85%	90%
	65%	75%	80%	85%	90%
baseline	63,94%	64,41%	64,61%	64,96%	64,55%
spell check Tumba	66,15%	68,56%	69,18%	70,12%	69,25%

Tabla 5.7. Resultados de aplicar el clasificador SMO a distintos tamaños del conjunto de entrenamiento para el dataset anchortext+tags aplicando spell check de Tumba!

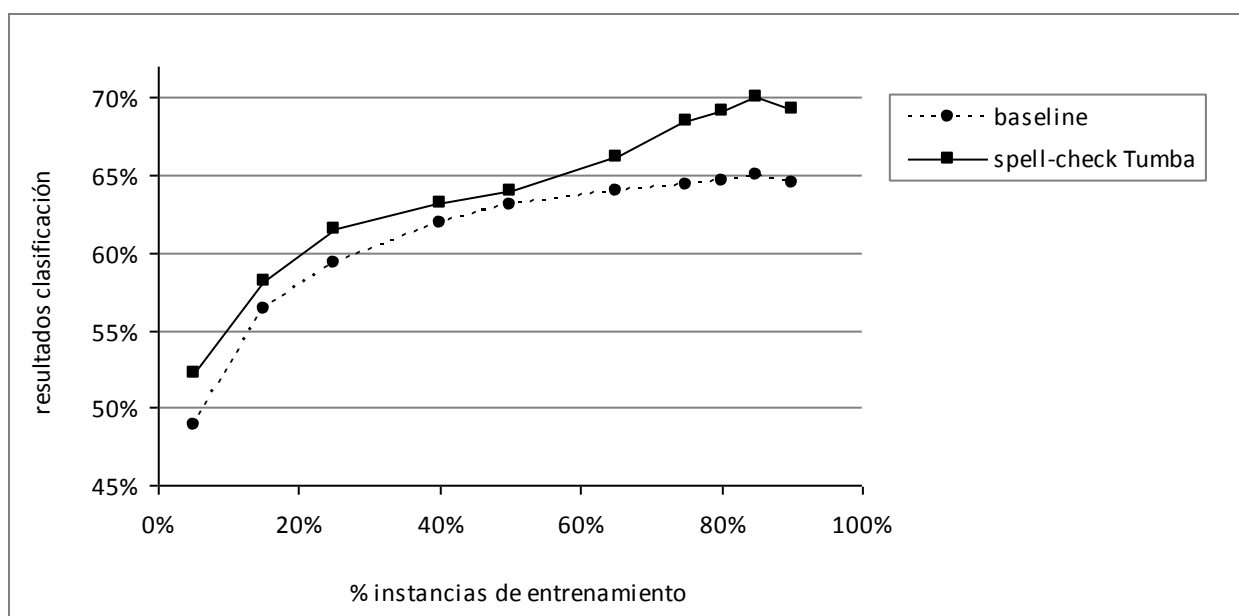


Gráfico 5.3. Representación gráfica de los resultados del clasificador SMO con distintas combinaciones de Cross-validation para anchortext+tags aplicando el spell check de Tumba!

Como se aprecia en la tabla, el dataset donde se aplicó el spell check de Tumba presenta mejores resultados que el tomado como baseline. El pico más alto es de 70,12% de instancias correctamente clasificadas, utilizando un 85% de documentos de entrenamiento.

⁷⁰ Distancia de Levenshtein (distancia de edición o distancia entre palabras): Se llama al número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra. Se entiende por operación, bien una inserción, eliminación o la sustitución de un carácter.

⁷¹ Similitud fonética: Teoría fonética que permite comparar cualquier par de eventos fonéticos y clasificarlos en una escala de similitud relativa, basándose en la forma acústica, fisiológica y perceptiva en que los eventos están relacionados entre sí.

⁷² Proximidad entre palabras: Distancia en el espacio, medido en palabras, entre dos palabras o dos frases claves.

Tabla comparativa de indicadores para el dataset baseline y el obtenido con el spell check Tumba.

Dataset (SMO)	Precision	Recall	F-measure	RAError
Percentage split: 66%				
baseline	0,648	0,643	0,645	94,50%
spell check Tumba	0,735	0,720	0,727	92,10%

Tabla 5.8. Indicadores obtenidos a partir del clasificador SMO con valores por defecto de percentage splits para los dataset baseline y spell check Tumba.

Los resultados de los indicadores utilizando esta librería de spell check se ven mejorados, como se aprecia en la tabla 5.8.

La segunda librería que se utilizó es *JaSpell*⁷³. Es una suite desarrollada en Java para el manejo de todo lo referido a spell checking. Provee una API que permite a los desarrolladores agregar spell-checkers a cualquier aplicación Java de manera muy simple. Provee soporte para los idiomas inglés y portugués, con lista de acrónimos y nombres propios que el sistema supone de ignorar. Los diccionarios son archivos de texto regulares donde cada línea contiene una palabra y una frecuencia de palabra asociada.

De manera similar al caso anterior, se muestran los resultados de la clasificación aplicando SMO y Percentage splits para el dataset generado utilizando JaSpell como spell-checker. Cabe destacar que la generación de los datasets con esta librería tomó mucho más tiempo que el obtenido con Tumba.

Dataset (SMO)	% instancias correctamente clasificadas - Percentage splits				
	5%	15%	25%	40%	50%
baseline	48,83%	56,43%	59,35%	61,88%	63,13%
spell check JaSpell	52,17%	57,99%	60,18%	63,95%	65,12%
Dataset (SMO)	65%	75%	80%	85%	90%
baseline	63,94%	64,41%	64,61%	64,96%	64,55%
spell check JaSpell	67,29%	68,26%	70,12%	71,25%	70,38%

Tabla 5.9. Resultados de aplicar el clasificador SMO a distintos tamaños del conjunto de entrenamiento para el dataset anchortext+tags aplicando spell check de JaSpell.

⁷³ <http://jaspell.sourceforge.net/>

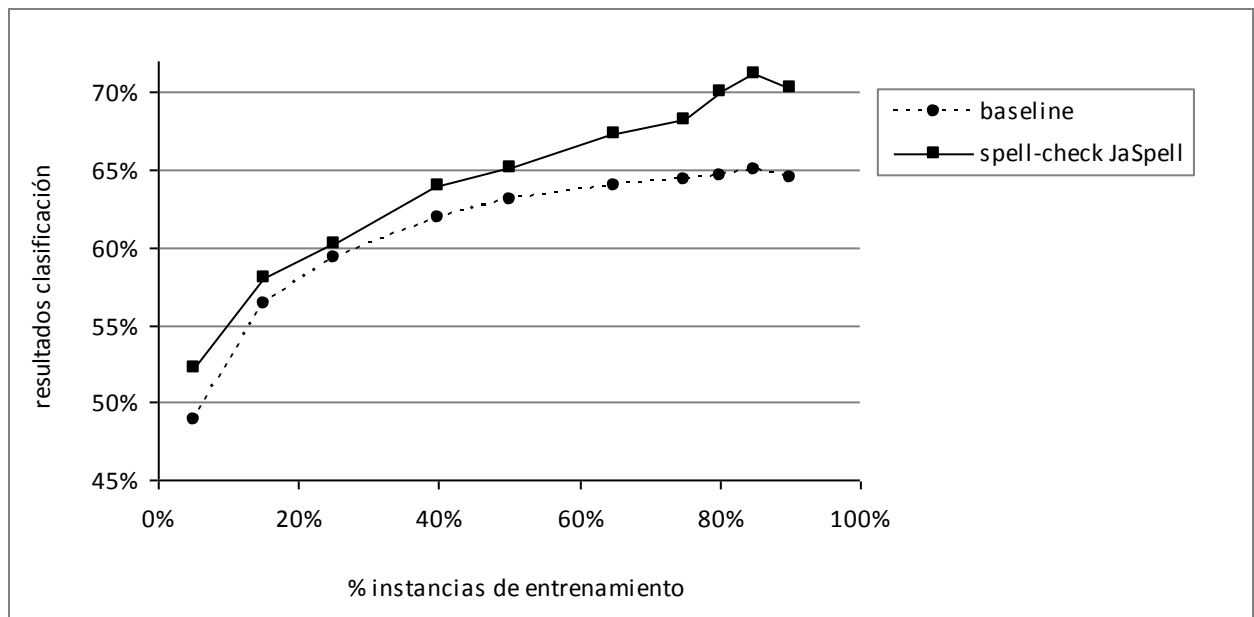


Gráfico 5.4. Representación gráfica de los resultados del clasificador SMO con distintas combinaciones de Cross-validation para anchortext+tags aplicando el spell check de JaSpell.

Al igual que con el primer spell checker, los resultados se vieron mejorados empleando este segundo corrector ortográfico. Para el 85% de instancias de entrenamiento se obtuvo un 71,25% de instancias correctamente clasificadas, siendo este el punto más alto.

Al igual que tabla 5.8, pero con spell check JaSpell.

Dataset (SMO)	Precision	Recall	F-measure	RAError
Percentage split: 66%				
baseline	0,648	0,643	0,645	94,50%
spell check JaSpell	0,760	0,735	0,747	91,56%

Tabla 5.10. Indicadores obtenidos a partir del clasificador SMO con valores por defecto de percentage splits para los dataset baseline y spell check JaSpell.

También en este caso, el spell check empleando JaSpell logró mejores resultados en cuanto a sus indicadores, siempre tomando como punto de comparación el dataset baseline.

La comparación entre las dos librerías analizadas se va a realizar luego de estudiar una tercera (y última) alternativa, denominada *Hunspell*⁷⁴. Es un corrector ortográfico y un analizador morfológico diseñado para idiomas de una morfología rica y codificación de caracteres o palabras compuestas complejas, diseñado originalmente para el idioma húngaro. Hunspell está basado en MySpell⁷⁵ y es

⁷⁴ <http://hunspell.sourceforge.net/>

⁷⁵ Era el corrector ortográfico por defecto de OpenOffice.org Writer, parte de la suite de oficina libre OpenOffice.org.

compatible con versiones anteriores de diccionarios de MySpell. Es utilizado en OpenOffice.org⁷⁶, Mozilla Firefox 3⁷⁷ & Thunderbird⁷⁸, Google Chrome⁷⁹ y también empleado por software propietario como Mac OS X⁸⁰ y Opera⁸¹, entre otros.

De la misma manera que se llevó a cabo con los spell-checkers anteriores, se generó un dataset utilizando Hunspell como su corrector ortográfico. Los resultados obtenidos en la clasificación son los siguientes:

Dataset (SMO)	% instancias correctamente clasificadas - Percentage splits				
	5%	15%	25%	40%	50%
baseline	48,83%	56,43%	59,35%	61,88%	63,13%
spell check Hunspell	52,12%	57,89%	60,15%	62,13%	64,63%

Dataset (SMO)	% instancias correctamente clasificadas - Percentage splits				
	65%	75%	80%	85%	90%
baseline	63,94%	64,41%	64,61%	64,96%	64,55%
spell check Hunspell	65,87%	66,98%	68,28%	69,12%	68,21%

Tabla 5.11. Resultados de aplicar el clasificador SMO a distintos tamaños del conjunto de entrenamiento para el dataset anchortext+tags aplicando spell check de Hunspell.

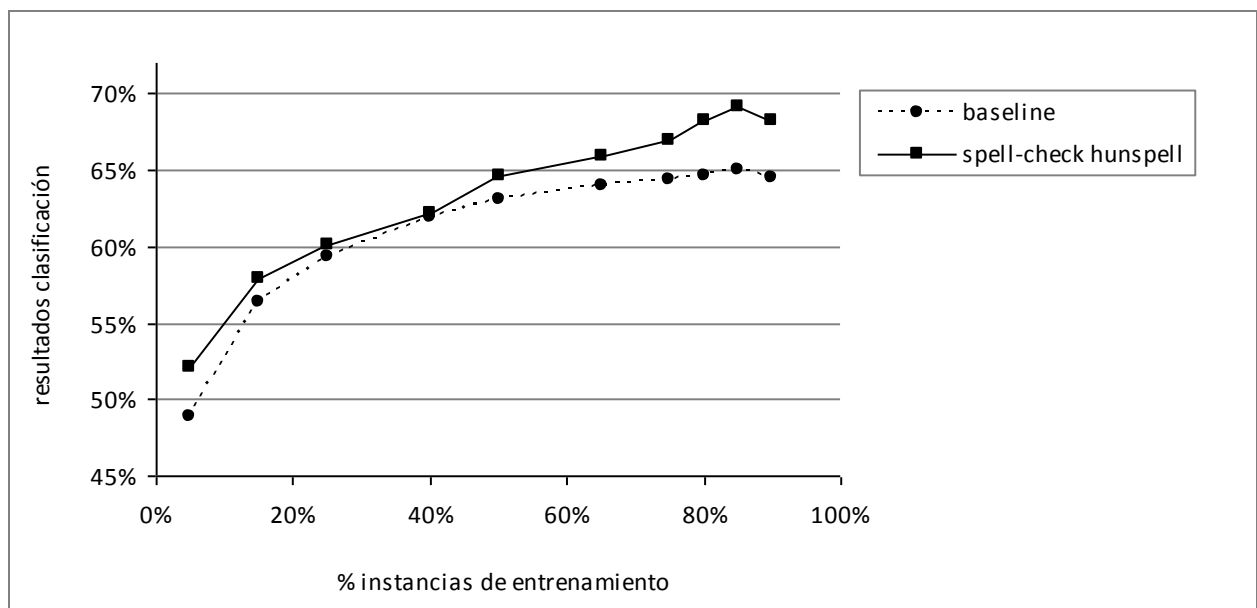


Gráfico 5.5. Representación gráfica de los resultados del clasificador SMO con distintas combinaciones de Cross-validation para anchortext+tags aplicando el spell check de Hunspell.

⁷⁶ Es una suite ofimática libre (código abierto y distribución gratuita) que incluye herramientas como procesador de textos, hoja de cálculo, presentaciones, herramientas para el dibujo vectorial y base de datos. Está disponible para varias plataformas.

⁷⁷ Es un navegador web libre (y uno de los más populares) descendiente de Mozilla Application Suite, desarrollado por la Corporación Mozilla, la Fundación Mozilla y un gran número de voluntarios externos.

⁷⁸ Es un cliente de correo electrónico de la Fundación Mozilla.

⁷⁹ Es un navegador web desarrollado por Google y compilado con base en componentes de código abierto.

⁸⁰ Es un sistema operativo desarrollado y comercializado por Apple Inc. que ha sido incluido en su gama de computadoras Macintosh desde 2002.

⁸¹ Es un navegador web y suite de Internet creado por la empresa noruega Opera Software.

De la misma forma que los spell-checkers Tumba y JaSpell, los resultados se vieron mejorados utilizando Hunspell. La mejor marca fue de 69,12% de instancias correctamente clasificadas para el 85% de los casos.

De la misma forma que se hizo con los dos spell check anterior, pero en este caso usando Hunspell.

Dataset (SMO)	Precision	Recall	F-measure	RAError
Percentage split: 66%				
baseline	0,648	0,643	0,645	94,50%
<i>spell check Hunspell</i>	0,702	0,695	0,698	93,38%

Tabla 5.12. Indicadores obtenidos a partir del clasificador SMO con valores por defecto de percentage splits para los dataset baseline y spell check Hunspell.

Utilizando Hunspell también se logra superar la performance obtenida por el dataset baseline en cuanto a sus indicadores.

Como se puede apreciar en las pruebas realizadas, los tres spell-checkers consiguen mejorar la performance del baseline, logrando su mejor marca cuando el clasificador utiliza un 85% de instancias de entrenamiento. La forma en que se comportan los resultados de las clasificaciones es muy similar, basta con ver los gráficos 5.3, 5.4 y 5.5, pero es JaSpell quien logra una pequeña diferencia con respecto al resto, seguramente porque maneje un diccionario más completo lo que provoca que su algoritmo de sugerencias funcione mejor.

La justificación de por qué los resultados se vieron mejorados luego de aplicar spell check (cualquiera de ellos) está relacionado con lo que se discutió antes de implementar esta optimización. Volviendo al caso de *photography* vs *photographi*, la primera palabra existe 215 veces en el dataset baseline, mientras que la segunda, la cual se encuentra escrita incorrectamente, aparece 29 veces en el mismo conjunto de datos. Por lo tanto, al corregir todas las ocurrencias de *photographi* en el nuevo dataset, el término *photography* es el único que sobrevive con 244 ocurrencias (215 + 29). Como este caso existen muchos otros. Por lo tanto, el algoritmo de clasificación se evitó de andar lidiando con términos incorrectos que no hacían más que confundir y bajar la performance de la clasificación. Esta es simplemente la razón de la mejora.

Como resumen, se presenta una tabla y gráfico comparativo de los resultados obtenidos con los tres spell-checkers, junto al dataset baseline.

Dataset (SMO)	% instancias correctamente clasificadas - Percentage splits				
	5%	15%	25%	40%	50%
baseline	48,83%	56,43%	59,35%	61,88%	63,13%
<i>spell check Tumba</i>	52,23%	58,15%	61,56%	63,17%	64,01%
<i>spell check JaSpell</i>	52,17%	57,99%	60,18%	63,95%	65,12%
<i>spell check Hunspell</i>	52,12%	57,89%	60,15%	62,13%	64,63%

Dataset (SMO)	65%	75%	80%	85%	90%
baseline	63,94%	64,41%	64,61%	64,96%	64,55%
<i>spell check Tumba</i>	66,15%	68,56%	69,18%	70,12%	69,25%
<i>spell check JaSpell</i>	67,29%	68,26%	70,12%	71,25%	70,38%
<i>spell check Hunspell</i>	65,87%	66,98%	68,28%	69,12%	68,21%

Tabla 5.13. Comparación de los resultados obtenidos al aplicar el clasificador SMO a distintos tamaños del conjunto de entrenamiento para los tres spell-checkers.

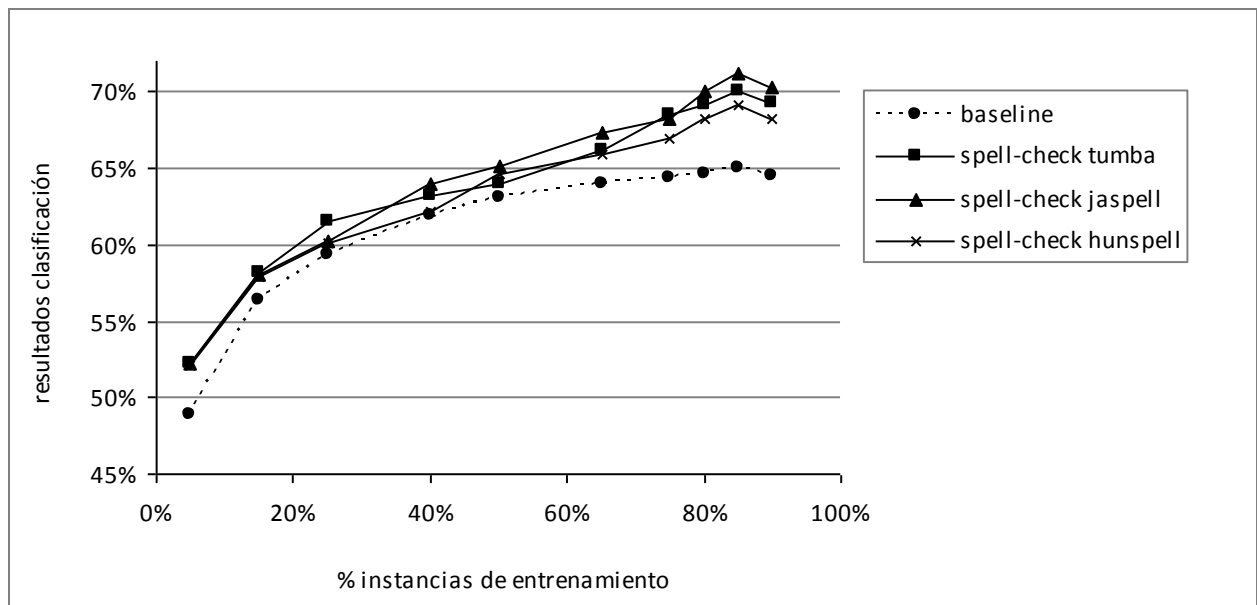


Gráfico 5.6. Representación gráfica de los resultados del clasificador SMO con distintas combinaciones de Cross-validation para anchortext+tags para los tres spell-checkers.

Conclusión: Esta optimización fue aceptada ya que mejora los resultados de la clasificación obtenido con el dataset baseline. Ello se puede ver en la tabla 5.13, donde el spell-checker de JaSpell es quien registra el porcentaje más alto de instancias correctamente clasificadas.

5.5. Aplicando spell check mejorado

Tomando como base 5.4 donde se logran mejoras con respecto al dataset baseline, la idea es tratar de optimizar el spell-checker corrigiendo algunas deficiencias encontradas. Como se explicó anteriormente, cuando el corrector ortográfico encuentra una palabra incorrecta, lo que hace a continuación es buscar una sugerencia para ese término (como el caso de *photography* y *photographi*). Si no existe ninguna sugerencia totalmente válida, entonces esa palabra se descarta. Esto provoca pérdida de información por parte del documento.

Para evitar esto último, la idea es lograr descartar el menor número de palabras que, al menos para el spell-checker, resulten incorrectas. Para ello, se realizó un análisis de los términos eliminados por el corrector ortográfico ante la generación de un nuevo dataset, y se llegó a la conclusión que la mayoría corresponde a uno de dos grupos: abreviaciones por un lado (ej. “acct.” por “account” o “ins.” por “insurance”), y palabras en otro idioma que no sea el inglés (ej. “foundation” por “fundación”), por el otro. Al igual que en 5.4, los tags siguen siendo el principal recurso en lo que hace a la generación de términos incorrectos en el dataset, por las razones antes mencionadas. Una vez definidos estos patrones, la idea fue implementar distintos algoritmos y métodos que ataquen de manera independiente cada uno de estos sucesos. Se considera un caso exitoso cuando un término que previamente había sido descartado por el spell-checker, finalmente puede ser puesto en consideración y formar parte del dataset final.

En la siguiente página se presenta un diagrama que grafica el nuevo flujo de filtrado al cual fueron sometidos los documentos del dataset.

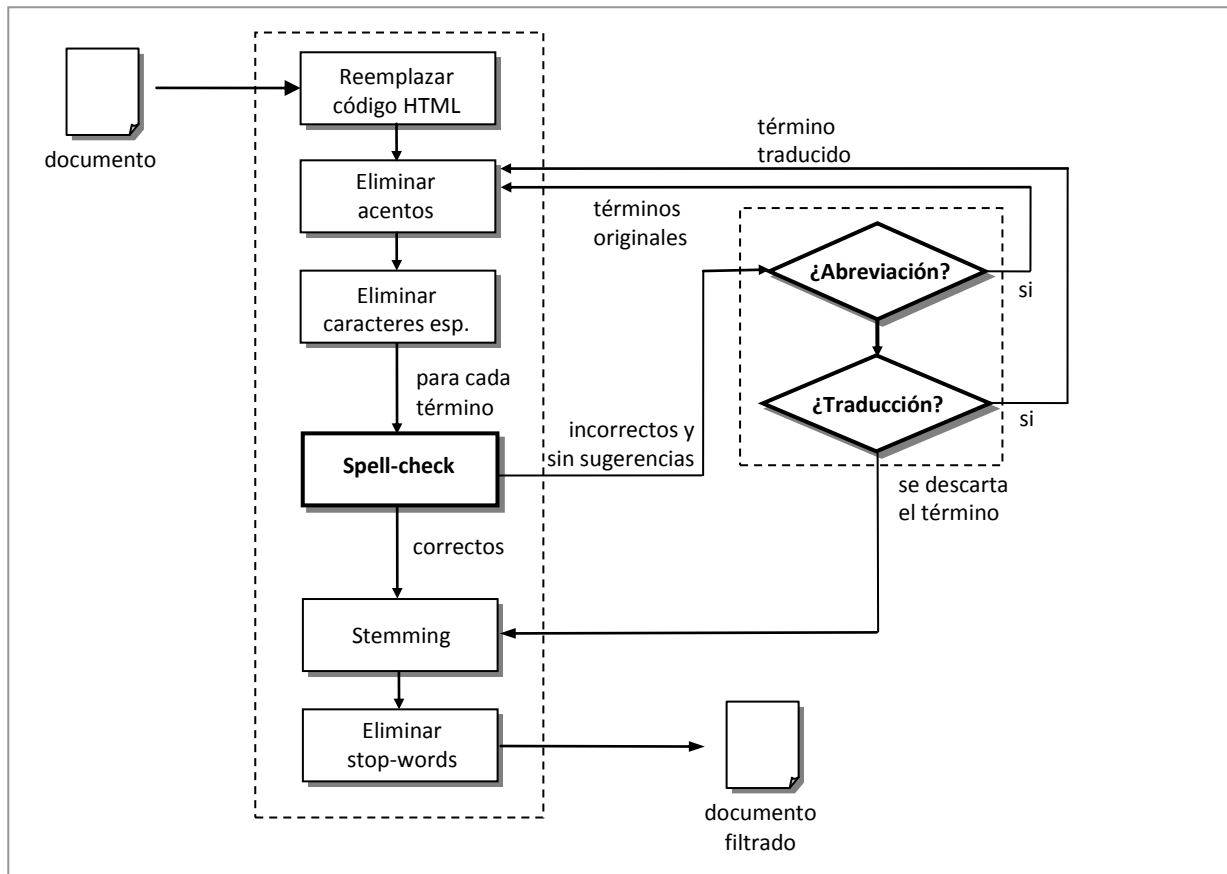


Figura 5.4. Filtrado aplicado al contenido de un documento, incluyendo un spell checker mejorado.

El funcionamiento es el siguiente: una vez que el corrector ortográfico (Spell-check) determina una colección de términos incorrectos y sin sugerencia alguna para un documento, entran en juego los nuevos filtros (ubicados hacia la derecha en la Figura 5.4). Primero, se chequea si el término incorrecto pertenece a alguna abreviación en inglés (¿Abreviación? en la figura). En caso afirmativo, todos los términos originales correspondientes a esa abreviación son procesados nuevamente ya que, por ejemplo, puede ser que alguno de ellos contenga caracteres especiales y/o acentos. Si este no es el caso, se verifica que el término no esté escrito en otro idioma. De ser así, se realiza la traducción al inglés y se continúa con la misma lógica previa. Por último, si el término no aplica a ninguno de los casos anteriores, es descartado. Cabe destacar que si el spell-check no encuentra ningún término incorrecto dentro de un documento, entonces no hay participación de los nuevos módulos. Finalmente, a todos los términos correctos de un documento se les aplica stemming y eliminación de stop words.

Técnicamente, cada uno de estos dos nuevos filtros fue desarrollado en Java de manera independiente. Para el caso de las *abreviaciones*, se confeccionó un diccionario que cuenta con las más comunes y usadas del idioma inglés. Esta información se obtuvo de [59]. El funcionamiento es bastante simple, se creó una API que recibe el término abreviado, devolviendo el término original si lo hubiere.

Mientras que por otro lado, para las traducciones se estudiaron distintas librerías Java, entre ellas *Web Translator Java API*⁸², pero finalmente se decidió por *Google API Translate Java*, que es una librería basada en el traductor de Google. Es una de las más completas y eficaces que existen, pero tiene un solo defecto a los términos prácticos de esta investigación: no se ejecuta localmente en la máquina dónde se realiza el procesamiento sino que hace llamadas mediante web service a Google para resolver cada traducción. Por consiguiente, se triplicó el tiempo que demoró la generación de este nuevo dataset. Esta librería consta de una API que recibe tres parámetros: los términos a traducir, el idioma en que esos términos se encuentran y el idioma deseado para convertirlo (en este caso, Inglés). Con respecto al segundo parámetro, al desconocer la lengua en que se encuentran escritos los términos a procesar, se decidió utilizar las más comunes ordenadas por el número de ocurrencias (español, portugués, francés y alemán), estas fueron cuestiones meramente prácticas. La idea es probar si el término tiene una traducción al español, de no ser, se chequea con el portugués, y de la misma manera para los demás casos.

Hasta el momento, la cantidad de términos descartados era de aproximadamente el 12%. Luego de aplicar la nueva técnica de spellcheck mejorado, este número se redujo a un 10,3%, lo que denota que existe un 1,7% de palabras “recuperadas” que pasaron a ser tenidas en cuenta por el algoritmo de clasificación.

A continuación, se muestran los resultados de la clasificación utilizando SMO y Porcentaje splits, al igual que los casos anterior. También se listan los resultados del baseline y del spell-check original, para facilitar la comparación. Cabe aclarar que la librería spell-check utilizada fue JaSpell, ya que es la que mejores resultados obtuvo de las analizadas previamente.

Dataset (SMO)	% instancias correctamente clasificadas - Percentage splits				
	5%	15%	25%	40%	50%
baseline	48,83%	56,43%	59,35%	61,88%	63,13%
spell check JaSpell	52,17%	57,99%	60,18%	63,95%	65,12%
spell check mejorado	52,60%	58,18%	60,45%	64,93%	66,46%
Dataset (SMO)	65%	75%	80%	85%	90%
baseline	63,94%	64,41%	64,61%	64,96%	64,55%
spell check JaSpell	67,29%	68,26%	70,12%	71,25%	70,38%
spell check mejorado	67,73%	68,82%	71,29%	72,35%	70,92%

Tabla 5.14. Resultados de aplicar el clasificador SMO a distintos tamaños del conjunto de entrenamiento para el dataset anchortext+tags aplicando spell check mejorado.

⁸² Ver Links de interés, para más información.

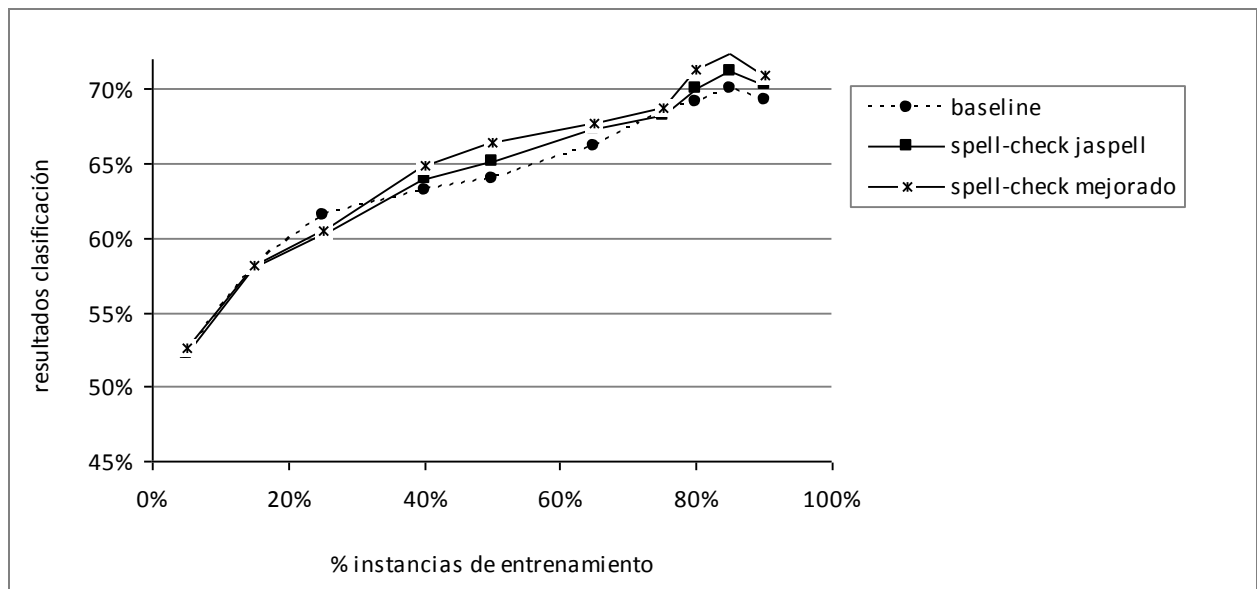


Gráfico 5.7. Representación gráfica de los resultados del clasificador SMO con distintas combinaciones de Cross-validation para anchortext+tags aplicando el spell check JaSpell y el mejorado.

Como se puede apreciar en la tabla 5.14, los resultados utilizando el *spell-check mejorado* varían leve pero positivamente en comparación con el spell-check original (*JaSpell*). La mejor marca es de 72,35% utilizando un 85% de instancias de entrenamiento. Esta ligera mejora se debe a los nuevos términos presentes en el dataset, que antes eran descartados por el corrector y ahora son puestos nuevamente en consideración del clasificador, previo chequeo y certificación que correspondían en realidad a información válida pero representada de manera incorrecta (ej. abreviación).

Por último, comparación de los indicadores con el dataset obtenido mediante el spell check mejorado.

Dataset (SMO)	Precision	Recall	F-measure	RAError
Percentage split: 66%				
baseline	0,648	0,643	0,645	94,50%
spell check JaSpell	0,760	0,735	0,747	91,56%
spell check mejorado	0,768	0,742	0,755	91,34%

Tabla 5.15. Indicadores obtenidos a partir del clasificador SMO con valores por defecto de percentage splits para los dataset baseline, spell check JaSpell y spell check mejorado.

En este caso, el spell check mejorado supera en cuanto a números de sus indicadores no sólo al dataset baseline, sino también a JaSpell.

Conclusión: Esta optimización fue aceptada y reemplazó a la anterior (ver 5.4) ya que mejora los resultados de la clasificación obtenido con el dataset baseline.

Resumen del capítulo 5

El objetivo fue tomar como “baseline” al dataset anchortext+tags que mejores resultados presentó en la etapa anterior, e implementar una serie de cambios con el fin de optimizar los resultados de la clasificación.

Se llevaron a cabo distintos intentos de optimización. Entre ellos, el primero consistió en no realizar stemming a cada uno de los términos del documento. Los resultados no fueron buenos, debido a la presencia de varios de ellos aparentemente iguales pero escritos de forma distinta (ej. en plural y singular), lo que influye negativamente en la clasificación.

Luego, se descubrió que el dataset está compuesto de un gran número de sinónimos. Por lo tanto, se empleó WordNet para reducir todos estos términos a uno en común. Se obtuvieron pobres resultados ya que se incorporó demasiada información extra para muchos términos, lo que provocó que el clasificador cuente con un mayor volumen de datos a la hora de procesar.

Por último, la idea fue eliminar los errores de ortografía presentes en los documentos, provenientes en su mayoría de los tags. Se emplearon 3 distintos algoritmos de spell-check, consiguiendo mejorar la performance del baseline en todos los casos. Luego, se realizaron algunas mejoras para descartar el menor número de términos incorrectos, lo que provocó optimizar aún más los resultados finales.

Conclusiones y discusión

La hipótesis de la presente investigación consiste en analizar la importancia de los marcadores sociales en la clasificación de documentos web. Diversos investigadores coinciden en que los tags pueden beneficiar la categorización automática, brindando para ello varias razones que fundamentan tal teoría. La certeza de esta suposición, como así también el valor que presentan estos marcadores sociales, fue la motivación necesaria para comenzar con el estudio.

Distintas investigaciones [36, 47] han llegado a la conclusión que los tags proveen importante información adicional que no está presente en los propios documentos, lo cual resulta en un novedoso recurso de interés en el campo de *data mining*. Se dice que los tags ayudan a identificar el “acerca de” los documentos [32]; siendo un recurso particularmente trascendente para aquellas nuevas, pero populares, páginas web; ya que el número de in-links o anchor-text que poseen es relativamente bajo lo cual no aporta información relevante.

Los tags son generalmente más diversos que los anchor-text, lo cual por un lado resultan en ser más ruidosos y potencialmente menos útiles. Pero por otro lado, [50] dedujo que los tags proveen un resumen multi-facético de los documentos web. Desde este punto de vista, la diversidad de los tags podría ser una ventaja ya que capturan información y significados que los anchor-text quizá pierdan. Con respecto al “ruido” que generan, [32] propone que la mejor forma de evitarlo es considerar sólo los *top n* tags, que es una técnica comúnmente utilizada para crear las llamadas nubes de tags (del inglés *tags clouds*) que se pueden encontrar en diversos sitios y blogs mayormente.

En este trabajo se propuso estudiar la efectividad de los algoritmos de clasificación a partir de los marcadores sociales provistos por los propios usuarios. Para ello, se seleccionó la colección de datos CABS120k08 y se construyó un parser para convertirlo al formato necesario a ser procesado por la herramienta Weka (diseñada para clasificación y aprendizaje de máquina en general). Vale señalar que el parser se diseñó de manera tal de permitir la generación de distintas combinaciones de datasets de manera sencilla.

Los experimentos llevados a cabo empleando esta base de datos se centraron en el estudio de la importancia de los tags para la clasificación de documentos web, y en el desarrollo de mejoras lingüísticas con el fin de disminuir los problemas provocados por la libre creación de los mismos (entre ellos, presencia de sinónimos, plurales y singulares para el mismo término, errores ortográficos, etc). Bajo este propósito, se evaluaron y pusieron en práctica diversos tratamientos de texto; entre ellos vale destacar la aplicación de stemming, la eliminación de stop-words, el empleo de spell-checkers y la implementación de traducciones al inglés.

Centrándonos en esta investigación, se ha demostrado que los tags efectivamente aportan valor a la clasificación automática de documentos web, tal como se describe en la sección 4.5. Se concluyó que la fusión de tags y anchortext resulta en la combinación ideal para la generación de datasets, siempre desde el punto de vista de los resultados obtenidos por parte del algoritmo de clasificación (SMO con PolyKernel). Adicionalmente, si se hace una clasificación individual de cada recurso, resulta que los tags son el que mayor aporte realiza a la clasificación, seguido luego por los anchortext y por último, las queries. Las razones por las cuales los marcadores sociales llegan a este logro básicamente son las mismas que las descritas anteriormente. De esta manera, se logra una concordancia entre lo que varios autores han deducido en diversos estudios y la hipótesis planteada en un comienzo.

Una vez definido como dataset ideal al compuesto por la información proveniente de los anchortexts y tags, se logró optimizarlo a partir de la aplicación de un spell checker con el fin de corregir errores ortográficos. Para aquellos términos supuestos de estar mal escritos, se realizó un chequeo adicional para considerar el caso que se trate de abreviaciones o bien se encuentren en otro idioma que no sea el inglés.

Nuestros experimentos con anotaciones sociales muestran resultados alentadores en cuanto a la utilidad de este novedoso –y no tanto– recurso en el campo del *data mining*, y más precisamente en la clasificación automática de documentos web. Esta investigación es sólo el comienzo. Existe un sinfín de posibilidades y beneficios que entregan los marcadores sociales y su positiva influencia en la categorización de documentos.

Existen diversas formas de mejorar o extender los resultados obtenidos, por ejemplo, se podría realizar un estudio más profundo de los tags intentando filtrar aquellos que pudieran considerarse subjetivos, o tratando de resolver los problemas de sinonimia (palabra que tiene varios significados) o polisemia (presencia de varias acepciones para una misma palabra o signo lingüístico) que presenta.

Por otro lado, en el servicio de Delicious, los usuarios pueden escribir notas, que no son más que meras descripciones en un lenguaje natural acerca de las páginas que taggean. Se podría agregar esta información a la fuente de datos CABS120k08 para luego evaluar si se comporta como un recurso valioso extra a la hora de mejorar los resultados obtenidos mediante el algoritmo de clasificación. Para ello, habría que realizar un tratamiento similar al empleado en los tags (eliminación de stop-words, spelling, etc) y filtrando aquellos comentarios que generen “ruido”. Adicionalmente, se podría considerar la “popularidad” de los documentos webs en Delicious. Por popularidad se refiere a aquellos documentos que han sido taggeados por un gran número de usuarios, lo cual le otorga cierta significancia dentro del dominio. Por lo tanto, los algoritmos de clasificación podrían verse alimentados por estos documentos *destacados* de manera que, en la etapa de entrenamiento, influyan en mayor relevancia que el resto de los documentos.

Por último, una extensión a esta investigación podría ser el considerar otros servicios populares similares a Delicious, con el fin de obtener nuevos marcadores sociales con el cual alimentar el algoritmo de clasificación. Con respecto a esto último, una fuente de datos bastante distinta pero

igualmente de importante podrían ser los links compartidos en las redes sociales más populares como Facebook⁸³ o Twitter⁸⁴ (los cuales poseen una potente API), obteniendo el texto que acompaña a cada mensaje como la información relacionada al documento web, como si se tratara de un marcador social.

Hay que tener en cuenta que esta investigación se realizó utilizando documentos en inglés. Sería interesante ver la forma de modificar el parser con el fin de que el algoritmo de clasificación de Weka pueda procesar datasets con documentos en distintos idiomas.

Trabajos futuros

Una importante aplicación de este estudio estaría estrechamente relacionada con el campo de las búsquedas web. Por ejemplo, Google tiene una opción llamada *SafeSearch* (en español, *búsqueda segura*) que impide que aparezcan páginas con contenido sexual explícito entre los resultados de la búsqueda. De esta forma, si un documento web es taggeado por diferentes usuarios con términos referentes a este dominio, entonces el algoritmo de clasificación categorizaría a este sitio con la “categoría prohibida” y por lo tanto no aparecería en los resultados si el filtro se encontrase activo. [60]

Los resultados obtenidos en [47] sugieren que, comúnmente, los tags realizan un mayor aporte a la categorización general (o amplia) de los documentos en comparación con las categorizaciones específicas de un dominio en particular (los usuarios prefieren utilizar términos generales en lugar de términos específicos a la hora de marcar un documento [17]). De esta forma, los marcadores sociales pueden ser útiles para la desambiguación de palabras claves en la consulta para el caso de la personalización de búsquedas web.

Por otro lado, la clasificación automática basada en tags podría ser empleada a la hora de sugerir categorías que refinan o expandan una búsqueda web, vinculando estas categorías con los términos a buscar por medio de los *top tags* asociados a cada una de ellas. Un estudio similar se puede encontrar en [61].

Tal como se mencionó en el Prólogo, el directorio ODP es administrado en forma manual, lo que supone que todos los documentos son categorizados por personas responsables de tal tarea. Con la incorporación de los conceptos aprendidos en esta investigación, se podrían reducir los tiempos en que un sitio web es aprobado y finalmente incluido en el directorio. El proceso podría consistir en utilizar un método automático para la mayoría de los sitios web, y en uno manual para aquellos documentos donde su clasificación no sea tan trivial. Algo similar al empleado por Ansearch (ver sección 1.2).

Finalmente, este estudio podría ser útil en el ámbito del marketing y la publicidad online, donde distintas herramientas podrían rastrear los sitios pertenecientes a una determinada categoría con el

⁸³ <http://www.facebook.com>

⁸⁴ <http://twitter.com>

objetivo de colocar allí sus banners de publicidad. Por ejemplo, si una empresa internacional como Nike tiene pensado lanzar una nueva línea de botines de fútbol en nuestro país, podría utilizar distintos tags relacionados al producto en cuestión (por ejemplo: “botines”, “fútbol”, “nike”, “argentina”, “Messi”, etc) junto a las categorías ya calculadas para obtener todos los sitios relacionados con fútbol en Argentina, y luego decidir en cuáles publicitar su producto.

Personalmente, considero que son amplias e interesantes las posibilidades que brindan los marcadores sociales en la clasificación de documentos web, pudiéndose emplear en distintos dominios y variadas aplicaciones.

Bibliografía

- [1] D. Godoy. Clasificación. « Análisis y Recuperación de Información » Facultad de Ciencias Exactas. 2007.
- [2] C.-C. Huang, S.-L. Chuang, L.-F. Chien. « Liveclassifier: Creating hierarchical text classifiers through web corpora ». En WWW '04: Proceedings of the 13th International Conference on World Wide Web, New York, NY, pp. 184–192. ACM Press. 2004.
- [3] C.-C. Huang, S.-L. Chuang, L.-F. Chien. « Using a web-based categorization approach to generate thematic metadata from texts ». ACM Transactions on Asian Language Information Processing (TALIP) 3(3), 190–212. 2004.
- [4] C. Chekuri, M. Goldwasser, P. Raghavan, E. Upfal. « Web search using automated classification ». En Proceedings of the Sixth International World Wide Web Conference, Santa Clara, CA. Poster POS725. 1997.
- [5] H. Chen, S. Dumais. « Bringing order to the Web: Automatically categorizing search results ». En CHI '00: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, New York, NY, pp. 145–152. ACM Press. 2000.
- [6] L. Getoor, C. Diehl. « Link mining: A survey ». SIGKDD Explorations Newsletter Special Issue on Link Mining 7(2). 2005.
- [7] J. Fürnkranz. « Web mining ». En O. Maimon, L. Rokach (Eds.), The Data Mining and Knowledge Discovery Handbook, pp. 899–920. Berlin. 2005.
- [8] B. Choi, Z. Yao. « Web page classification ». En W. Chu, T. Y. Lin (Eds.), Foundations and Advances in Data Mining, Volume 180 of Studies in Fuzziness and Soft Computing, pp. 221–274. Berlin. 2005.
- [9] O.-W. Kwon, J.-H. Lee. « Web page classification based on k-nearest neighbor approach ». En IRAL '00: Proceedings of the 5th International Workshop on Information Retrieval with Asian languages, New York, NY, pp. 9–15. ACM Press. 2000.
- [10] D. Shen, Z. Chen, Q. Yang, H.-J. Zeng, B. Zhang, Y. Lu, W.-Y. Ma. « Web-page classification through summarization ». En SIGIR '04: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, NY, pp. 242–249. ACM Press. 2004.
- [11] M.-Y. Kan. « Web page classification without the web page ». En WWW Alt. '04: Proceedings of the 13th International World Wide Web Conference Alternate Track Papers & Posters, New York, NY, pp. 262–263. ACM Press. 2004.
- [12] G. Attardi, A. Gulli, F. Sebastiani. « Automatic web page categorization by link and context analysis ». En C. Hutchison, G. Lanzarone (Eds.), Proceedings of THAI'99, First European Symposium on Telematics, Hypermedia and Artificial Intelligence, Varese, IT, pp. 105–119. 1999.

- [13] D. Mladenic. « Turning Yahoo into an automatic web-page classifier ». 1998.
- [14] T. O'Reilly. What is Web 2.0.
<http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>. 2004.
- [15] Web 2.0 - <http://www.paulgraham.com/web20.html>. 2005.
- [16] Semantic Web Activity W3C - <http://www.w3.org/2001/sw>
- [17] M. G. Noll, C. Meinel. « Exploring Social Annotations for Web Document Classification ». SAC '08: Proceedings of 23rd International ACM Symposium on Applied Computing, Fortaleza, Ceará, Brazil. 2008.
- [18] J. Udell, A. Mathes. Folksonomies - « Cooperative Classification and Communication Through Shared Metadata » - <http://www.adammathes.com/academic/computer-mediated-communication/folksonomies.html>. 2004.
- [19] M. G. Noll. CABS120k08. <http://www.michael-noll.com/wiki/CABS120k08>. 2008.
- [20] Weka 3 - Data Mining with Open Source Machine Learning Software in Java.
<http://www.cs.waikato.ac.nz/ml/weka>
- [21] M. Guy, E. Tonkin. « Folksonomies. Tidying up tags? ». D-Lig Magazine Volume 12 Number 1. 2006.
- [22] U. Ali Mejias. « A del.icio.us study - Bookmark, Classify and Share: A mini-ethnography of social practices in a distributed classification community ». Ideant.
- [23] A. Mathes. « Folksonomies - Cooperative Classification and Communication Through Shared Metadata ». Computer Mediated Communication. Graduate School of Library and Information Science. University of Illinois Urbana-Champaign. 2004.
- [24] A. Mathes. « Folksonomies – cooperative classification and communication through shared metadata » Computer Mediated Communication. 2004.
- [25] S. Golder, B. Huberman. « Usage patterns of collaborative tagging systems » Journal of Information Science, vol. 32, no. 2, pp. 198-208. 2006.
- [26] How to organize and categorize messages with labels in Gmail -
<http://email.about.com/od/gmailtips/qt/et032006.htm>
- [27] F. Abel, N. Henze, R. Kawase, D. Krause. « The impact of multifaceted tagging on learning tag relations and search ». IVS – Semantic Web Group & L3S Research Center. Leibniz University Hannover, Germany.
- [28] T. Vander Wal. «Explaining the Granular Social Network»
<http://www.vanderwal.net/random/category.php?cat=153>. 2008.
- [29] T. Vander Wal. « Explaining and showing broad and narrow folksonomies ».
http://www.personalinfocloud.com/2005/02/explaining_and_.html. 2005.

- [30] D. Ben Ayed Mezghani, S. Zribi Boujelbene, N. Ellouze. « Evaluation of SVM Kernels and Conventional Machine Learning Algorithms for Speaker Identification ». Dept. Electrical Engineering at the National School of Engineer of Tunis. 2010.
- [31] What is Google Page Rank? - <http://www.goingup.com/pagerank/pagerankwork.html>
- [32] M. G. Noll, C. Meinel. « The Metadata Triumvirate: Social Annotations, Anchor Text and Search Queries » University of Potsdam. 2008.
- [33] M. G. Noll. « Personalization 2.0: Web Search Personalization via Social Bookmarking and Tagging » Postdam, Alemania. 2007.
- [34] A. Hotho, R. Jäschke, C. Schmitz, G. Stumme. « Information retrieval in flockonomies: Search and ranking ». The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006, vol. 4011 of LNCS, pp. 411-426, Springer. 2006.
- [35] B. Krause, A. Hotho, G. Stumme. « A comparision of social bookmarking with traditional search ». Advances in Information Retrieval, vol. 4956, pp. 101-113. 2008.
- [36] A. Zubiaga, R. Martínez, V. Fresno. « Getting the Most Out of Social Annotations for Web Page Classification » ETSI Informática, UNED.
- [37] A. Zubiaga, A. García-Plaza, V. Fresno, R. Martínez. « Content-based Clustering for Tag Cloud Visualization » ASONAM 2009, International Conference on Advances in Social Networks Analysis and Mining, pp. 316-319, Athens, Greece. 2009.
- [38] X. Qi, B. D. Davison. « Web Page Classification. Features and algorithms ». Department of Computer Science & Engineering. Lehigh University. 2007.
- [39] S. Aliakbary, H. Abolhassani, H. Rahmani, B. Nobakht. « Web Page Classification using Social tags ». 2009 International Conference on Computational Science and Engineering. 2009.
- [40] S. Banerjee, M. Scholz. « Leveraging Web 2.0 Sources for Web Content Classification ». 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. 2008.
- [41] Z. Yin, R. Li, Q. Mei, J. Han. « Exploring Social Tagging Graph for Web Object Classification ». Dept. of Computer Science, University of Illions at Urbana-Champaign.
- [42] Z. Gantner, L. Schmidt-Thieme. « Automatic Content-based Categorization of Wikipedia ». Proceeding of the 2009 Workshop on the People's Web Meets. Suntec, Singapore. 2009.
- [43] S. Overell, B. Singbjorns, R. van Zwol. « Classifying Tags using Open Content Resources ». Multimedia and Information Systems and Yahoo! Research. London, UK and Barcelona, Spain.
- [44] A. Zubiaga. « Enhancing Navigation on Wikipedia with Social Tags » Wikimania 2009. Buenos Aires, Argentina. 2009.

- [45] A. Shepitsen, J. Gemmell, B. Mobasher, R. Burke. « Personalized recommendation in social tagging systems using hierarchical clustering ». Proceedings of the 2008 ACM Conference on Recommender Systems (RecSys 08), pp. 259-266. 2008.
- [46] L. Page, S. Sergey. « The PageRank Citation Ranking: Bringing Order to the Web ». Proceedings of Advances in Neural Information Processing Systems. 1998.
- [47] M. G. Noll, C. Meinel. « Authors vs. Readers - A Comparative Study of Document Metadata and Content in the WWW » University of Potsdam, Canadá. 2007.
- [48] P. Heymann, G. Koutrika, H. Garcia-Molina. « Can social bookmarking improve web search? » En Proceedings of 1st ACM International Conference on Web Search and Data Mining (WSDM'08), páginas 195–206. ACM. 2008.
- [49] A. Spink, D. Wolfram, M. B. J. Jansen, T. Saracevic. « Searching the web: The public and their queries ». Journal of the American Society for Information Science and Technology, 52(3):226–234. 2001.
- [50] S. Bao, G. Xue, X. Wu, Y. Yu, B. Fei, Z. Su. « Optimizing web search using social annotations. En WWW '07: Proceedings of the 16th international conference on World Wide Web » páginas 501–510, New York, NY, USA. ACM Press. 2007.
- [51] X. Wu, L. Zhang, Y. Yu. « Exploring social annotations for the semantic web ». En WWW '06: Proceedings of the 15th international conference on World Wide Web, pages 417-426. New York, NY, USA. AMC Press. 2006.
- [52] S. Xu, S. Bao, Y. Cao, Y. Yu. « Using social annotations to improve language model for information retrieval ». En CIKM '07: Proceedings of the 16th ACM conference on Conference on information and knowledge management, pages 1003-1006. New York, NY, USA. 2007.
- [53] C. man Au Yeung, N. Gibbins, N. Shadbolt. « Web search disambiguation by collaborative tagging ». En Proceedings of the Workshop on Exploring Semantic Annotations in Information Retrieval (ESAIR) at ECIR '08, pages 48-61. 2008.
- [54] E. Leopold, J. Kindermann. « Text Categorization with Support Vector Machines. How to represent texts in input space? ». GMD German National Research Center for Information Technology, Institute for Autonomous intelligent Systems, Schloss Birlinghoven, D-53754 Sankt Augustin, Germany. 2002.
- [55] S. Tong, D. Koller. « Support vector machine active learning with applications to text classification ». The Journal of Machine Learning Research. Volume 2, pages 45-66. 2002.
- [56] H. Zhang. « The Optimality of Naïve Bayes ». Faculty of Computer Science. University of New Brunswick. Fredericton, New Brunswick, Canada. 2004.
- [57] J. Rennie, L. Shih, J. Teevan, D. Karger. « Tackling the Poor Assumptions of Naïve Bayes Text Classifiers ». Artificial Intelligence Laboratory. Massachusetts Institute of Technology, Cambridge, MA 02139.

- [58] I. Rish. « An empirical study of the Naïve Bayes classifier ». Thomas J. Watson Research Center. Yorktown Heights, NY. 10598.
- [59] « The Oxford English Dictionary: List of Abbreviations ». 1996.
- [60] M. G. Noll. « Web Page Classification: An Exploratory Study of the Usage of Internet Content Rating Systems » LIASIT - Luxembourg International Advanced Studies in Information Technologies. Luxembourg. 2005.
- [61] K. Sugiyama, K. Hatano, M. Yoshikawa. « Adaptive web search based on user profile constructed without any effort from users » Proceedings of WWW 2004, pp. 675–684. 2004.
- [62] D. A. Hull. « Stemming Algorithms. A Case Study for Detailed Evaluation ». 1995.
- [63] J. Lovins. « Development of a stemming algorithm ». Mechanical Translation and Computational Linguistics. 1968.
- [64] R. Baeza-Yates, B. Ribeiro-Neto. « Modern Information Retrieval ». ACM Press/Addison Wesley. 1999.
- [65] J. Kamps, C. Monz, M. Rijke, B. Sigurbjörnsson. « Language-dependent and Language-independent Approach to Cross-Lingual Text Retrieval ». Comparative Evaluation of Multilingual Information Access Systems. 2004.
- [66] E. Airio. « Word normalization and decompounding in mono- and bilingual IR ». Informational Retrieval. 2006.
- [67] A. Viera, J. Virgil. « Uma revisão dos algoritmos de radicalização em língua portuguesa ». Information Research. 2007.
- [68] M. Porter. « An algorithm for suffix stripping ». Program, 14, pages 130-137. 1980.
- [69] C. Fox. « A stop list for general text ». ACM SIGIR Forum, Volume 24, pages 19-21. 1989.
- [70] R. Tsz-Wai Lo, B. He, I. Ounis. « Automatically Building a Stopword List for an Information Retrieval System ». Departement of Computing Science. University of Glasgow. 2005.
- [71] M. G. Noll. Delicious Python API. http://www.michael-noll.com/wiki/Del.icio.us_Python_API
- [72] Y. Karaban. WWW-Google-PageRank. <http://search.cpan.org/dist/WWW-Google-PageRank>
- [73] E. Hoffmann. « Standard statistical classifications: Basic principles ». United Nations Statistics Division. 1999.
- [74] D. Godoy. Clasificación - « Análisis y Recuperación de Información » Facultad de Ciencias Exactas. 2007.
- [75] D. H. Pink. Folksonomy, <http://www.nytimes.com/2005/12/11/magazine/11ideas1-21.html>. New York Times. 2005.

- [76] C. Firan, W. Nejdl, R. Paiu. « The benefit of using tag-based profiles » Proceedings of the 2007 Latin American Web Conference (LA-WEB 2007), pp. 32-41. 2007.
- [77] J. Stoyanovich, S. A. Yahia, C. Marlow, C. Yu. « Leveraging tagging to model user interests in del.icio.us » AAAI Spring Symposium on Social Information Processing (AAAI-SIP), pp. 104-109. 2008.
- [78] A. Mathes. « Folksonomies - Cooperative Classification and Communication Through Shared Metadata ». Computer Mediated Communication. Graduate School of Library and Information Science. University of Illinois Urbana-Champaign. 2004.
- [79] I. Steinwart, A. Christmann. « Support Vector Machine ». Information Science and Statistics. 2008.
- [80] T. Joachims. « Text categorization with Support Vector Machines: Learning with many relevant features ». Lecture Notes in Computer Science, 1998, Volume 1398/1998, 137-142, DOI: 10.1007/BFb0026683.

Links de interés

Google – Portal de internet cuyo principal producto es un motor de búsqueda. <http://www.google.com>

Bing – Es un motor de búsqueda, propiedad de Microsoft. <http://www.bing.com>

Yahoo! – Portal de internet que posee una importante gama de servicios. <http://www.yahoo.com>

Awstats – Herramienta open source de informes de análisis web. <http://awstats.sourceforge.net>

Webtrends – Herramienta de análisis web, orientado a social media. <http://www.webtrends.com>

Google Analytics – Servicio gratuito de estadísticas de sitios web, propiedad de Google. <http://www.google.com/analytics>

Michael G. Noll – Sitio web personal de uno de los autores más reconocidos en el ámbito de IR, donde pueden encontrarse todas sus publicaciones y otros materiales interesantes. <http://www.michael-noll.com>

ODP. Open Directory Project – Proyecto colaborativo multilingüe en el que editores voluntarios listan y categorizan enlaces a páginas web. La colección CABS120k08 fue construida tomando los documentos y categorías de este directorio. <http://www.dmoz.org>

Wikipedia – Proyecto de la Fundación Wikimedia que consiste de una enciclopedia libre y políglota. Sus artículos son redactados conjuntamente por voluntarios de todo el mundo. Ha sido una referencia importante a lo largo del desarrollo del presente trabajo. <http://www.wikipedia.org>

Social Bookmarking in Plain English – Video educativo que describe la utilidad y funcionamiento de los marcadores sociales. <http://www.commoncraft.com/bookmarking-plain-english>

Ansearch – Empresa australiana que presta servicios en Internet. Su principal producto es un motor de búsqueda. <http://www.ansearch.com/browse>

Flickr – Sitio web que permite almacenar, ordenar, buscar, vender y compartir fotografías y videos en línea, propiedad de Yahoo. <http://www.flickr.com>

Gmail – Servicio de correo electrónico gratuito, propiedad de Google. <http://www.gmail.com>

Google Safesearch – Opción del motor de búsqueda de Google que impide que aparezcan páginas con contenido sexual en los resultados. <http://goo.gl/mWqLj>

WordNet – Enorme base de datos léxica del idioma inglés. Ha sido utilizada como base para una de las optimizaciones llevadas a cabo en este trabajo. <http://wordnet.princeton.edu>

Tumba – Fue un motor de búsqueda optimizado para la web portuguesa. Se dejó de dar soporte en el 2009. <http://xldb.fc.ul.pt/wiki/Tumba!>

JaSpell – Suite desarrollada en Java para el manejo de todo lo referido a spell checking.
<http://jaspell.sourceforge.net/>

Hunspell – Librería Java de spell check. <http://hunspell.sourceforge.net>

Web Translator Java API – Librería Java que provee localmente traducción automática a distintos idiomas. <http://sourceforge.net/projects/webtranslator>

Google Translator – Traductor multi-idioma de Google. <http://translate.google.com>

Google API Translate Java – Librería que permite el uso del traductor de Google en cualquier aplicación Java. <http://code.google.com/p/google-api-translate-java>

Facebook – La red social más popular del mundo. <http://www.facebook.com>

Twitter – Red social y servicio de microblogging. <http://twitter.com>

MySpace – Sitio web de interacción social. <http://www.myspace.com>

Anexo A: Glosario

Traducciones

Agentes inteligentes: Es una entidad capaz de percibir su entorno, procesar tales percepciones y responder o actuar de manera racional, es decir, de manera correcta y tendiendo a maximizar un resultado esperado.

API (Application Programming Interface): En español, *Interfaz de Programación de Aplicaciones*. Una API representa una interfaz de comunicación entre componentes de software. Se trata del conjunto de llamadas a ciertas bibliotecas que ofrecen acceso a ciertos servicios desde los procesos y representa un método para conseguir abstracción en la programación.

Artificial Intelligence (AI): En español, *Inteligencia artificial (IA)*. Es una rama perteneciente a Ciencias de la Computación dedicada al desarrollo de agentes racionales no vivos. Se define la inteligencia artificial como aquella inteligencia exhibida por artefactos creados por humanos (de allí el término “artificial”). A menudo se aplica hipotéticamente a los computadores. El nombre también se usa para referirse al campo de la investigación científica que intenta acercarse a la creación de tales sistemas.

Clustering: En español, *Algoritmo de agrupamiento*. Es un procedimiento de agrupación de una serie de vectores de acuerdo con un criterio de cercanía. Esta cercanía se define en términos de una determinada función de distancia, como la euclídea, aunque existen otras más robustas o que permiten extenderla a variables discretas. Generalmente, los vectores de un mismo grupo (*cluster*) comparten propiedades comunes.

Data mining: En español, *Minería de datos*. Consiste en la extracción no trivial de información que reside de manera implícita en los datos. Dicha información era previamente desconocida y podrá resultar útil para algún proceso. En otras palabras, la minería de datos prepara, sondea y explora los datos para sacar la información oculta en ellos.

Dataset: En español, *Conjunto de datos*. Es una colección de datos que por lo general se presenta en forma de tabla. Cada columna representa una variable en particular. Cada fila corresponde a un determinado miembro del conjunto de datos en cuestión. Este conjunto puede incluir datos correspondientes a uno o más miembros, lo que corresponde al número de filas.

DBMS (Database Management System): En español, *Sistemas de Gestión de Base de Datos*. Son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

Folcsonomía: También llamado *folksonomía*. Es una indexación social, es decir, clasificación colaborativa por medio de etiquetas simples en un espacio de nombres llano, sin jerarquías ni relaciones de parentesco predeterminadas. Se trata de una práctica que se produce en entornos de

software social cuyos mejores exponentes son los sitios compartidos como Delicious y Flickr, entre otros.

Función de kernel: Es una generalización de la métrica de distancia. Mide la distancia entre dos vectores ya que los datos se proyectan en el espacio de dimensiones superiores.

Incoming hyperlink: En español, *link entrante*. Una página web A tiene un incoming hyperlink cuando existe otra página web B, que tiene un enlace hacia A. Por lo tanto, A puede tener muchos incoming hyperlinks provenientes de varias páginas webs (B, C, D, etc.). A partir de este concepto, se creó el algoritmo de PageRank utilizado en el buscador de Google, donde se asigna una popularidad a cada página web basándose en la cantidad y calidad de enlaces hacia ella (sumado a otros atributos).

Information retrieval: En español, *Recuperación de información*. Es la ciencia de la búsqueda de información en documentos, búsqueda de los mismos documentos, la búsqueda de metadatos que describan documentos, o también, la búsqueda en bases de datos. Todo esto ya sea a través de internet, intranet, para textos, imágenes, sonido o datos de otras características, de manera pertinente y relevante.

Logs: Es un registro oficial de eventos durante un rango de tiempo determinado. La mayoría de los logs son almacenados o desplegados en el formato estándar, el cual es un conjunto de caracteres para dispositivos comunes y aplicaciones. De esta forma cada log generado por un dispositivo en particular puede ser leído y desplegado en otro diferente.

Long tail effect: Expresión acuñada por Chris Anderson en un artículo de la revista Wired⁸⁵ de octubre de 2004⁸⁶ para describir determinados tipos de negocios y modelos económicos tales como Amazon.com⁸⁷ o Netflix⁸⁸. El término larga cola se utiliza normalmente en estadística en relación con distribuciones de riqueza o con el uso del vocabulario. En estas distribuciones una amplia frecuencia o gran frecuencia de transacciones es seguida por una baja frecuencia o baja amplitud de la población que disminuye gradualmente. En muchos casos, los acontecimientos de baja frecuencia o escasa amplitud pueden abarcar la mayor parte del gráfico.

Machine learning: En español, *Aprendizaje automático o Aprendizaje de máquinas*. Es una rama de la Inteligencia Artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras *aprender*. De forma más concreta, se trata de crear programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos. Es, por lo tanto, un proceso de inducción del conocimiento. El Aprendizaje Automático tiene una amplia gama de aplicaciones, incluyendo motores de búsqueda, diagnósticos médicos, detección de fraude en el uso de tarjetas de crédito, análisis del mercado de valores, clasificación de secuencias de ADN, reconocimiento del habla y del lenguaje escrito, juegos y robótica.

⁸⁵ Wired es una revista mensual americana de tecnología, <http://www.wired.com>

⁸⁶ <http://www.wired.com/wired/archive/12.10/tail.html>

⁸⁷ Compañía estadounidense de comercio electrónico, <http://www.amazon.com>

⁸⁸ Compañía de alquiler de DVD y discos Blu-ray en los Estados Unidos, <http://www.netflix.com>

Metadatos: Literalmente *sobre datos*, son datos que describen otros datos. En general, un grupo de metadatos se refiere a un grupo de datos, llamado *recurso*. El concepto de metadatos es análogo al uso de índices para localizar objetos en vez de datos. Por ejemplo, en una biblioteca se usan fichas que especifican autores, títulos, casas editoriales y lugares para buscar libros. Así, los metadatos ayudan a ubicar datos.

Microblogging: Es un servicio que permite a sus usuarios enviar y publicar mensajes breves (alrededor de 140 caracteres), generalmente de sólo texto. Las opciones para el envío de los mensajes varían desde sitios web, a través de SMS, mensajería instantánea o aplicaciones ad hoc.

Morfología: Es la rama de la lingüística que estudia la estructura interna de las palabras para delimitar, definir y clasificar sus unidades, las clases de palabras a las que da lugar (morfología flexiva) y la formación de nuevas palabras (morfología léxica).

Motor de búsqueda: Es un sistema informático que busca archivos almacenados en servidores web gracias a su “web spider”. Un ejemplo son los buscadores de Internet (algunos buscan sólo en la Web pero otros además lo hacen en noticias, servicios como Gopher, FTP, etc.) cuando se consulta sobre algún tema. Las búsquedas se hacen con palabras clave o con árboles jerárquicos por categorías; el resultado de la búsqueda es un listado de direcciones Web en los que se mencionan temas relacionados con la query realizada.

Open source: Es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas las cuales destacan en el llamado software libre.

PageRank: Es una marca registrada y patentada por Google el 9 de enero de 1999 que ampara una familia de algoritmos utilizados para asignar de forma numérica la relevancia de los documentos (o páginas web) indexados por un motor de búsqueda. Sus propiedades son muy discutidas por los expertos en optimización de motores de búsqueda. El sistema PageRank es utilizado por el popular motor de búsqueda Google para ayudarle a determinar la importancia o relevancia de una página. PageRank confía en la naturaleza democrática de la web utilizando su vasta estructura de enlaces como un indicador del valor de una página en concreto. Fue desarrollado por los fundadores de Google, Larry Page y Sergey Brin, en la Universidad de Stanford.

Parser: Un analizador sintáctico (en inglés *parser*) es uno de los módulos de un compilador que transforma su entrada en un árbol de derivación. Este convierte el texto de entrada en otras estructuras (comúnmente árboles), que son particularmente útiles para el posterior análisis y capturan la jerarquía implícita de la entrada. Un analizador léxico crea tokens de una secuencia de caracteres de entrada y son estos tokens los que son procesados por el analizador sintáctico para construir la estructura de datos, por ejemplo un árbol de análisis o árboles de sintaxis abstracta. [62]

Power law: Es un tipo especial de relación matemática entre dos cantidades. Aplicado a la estadística, si estas dos cantidades son la variable aleatoria y su frecuencia, en una distribución de ley de potencias, las frecuencias decrecen según un exponente cuando la variable aleatoria aumenta.

Query: En español, consulta realizada contra una base de datos. Se usa para obtener datos, modificarlos o bien borrarlos. Más específicamente, esta investigación se refiere a una web search query, que es una consulta que los usuarios ingresan en un motor de búsqueda para obtener la información que necesitan. Estas queries tienen la particularidad de no ser estructuradas y a menudo ambiguas.

RDBMS (Relational Data Base Management System): En español, *Sistemas de Gestión de Base de Datos Relacionales*. Es un DBMS en el que los datos se almacenan en forma de tablas, como así también la relación entre los mismos.

RSS: En inglés *Really Simple Syndication*. Es una familia de formatos de fuentes web codificados en XML. Se utiliza para suministrar a suscriptores de información actualizada frecuentemente. El formato permite distribuir contenido sin necesidad de un navegador, utilizando un software diseñado para leer estos contenidos RSS (agregador). Ha sido desarrollado específicamente para todo tipo de sitios que se actualicen con frecuencia (como diarios) y por medio del cual puedan compartir la información y usarla en otros sitios web o programas.

Redes sociales: Es una estructura social compuesta de personas (u organizaciones u otras entidades), las cuales están conectadas por uno o varios tipos de relaciones, tales como amistad, parentesco, intereses comunes, intercambios económicos, relaciones sexuales, o que comparten creencias, conocimiento o prestigio. Entre las más populares se encuentran Facebook, Twitter y MySpace.

Reworking: Comúnmente significa realizar nuevamente algún determinado trabajo, o bien llevar a cabo revisiones significativas a un trabajo ya finalizado.

Stemming: Es un método para reducir una palabra a su raíz o (en inglés) a un *stem* o lema. Hay algunos algoritmos de stemming que ayudan en sistemas de recuperación de información. Stemming aumenta el *recall* que es una medida sobre el número de documentos que se pueden encontrar con una consulta. Por ejemplo, una query sobre “bibliotecas” también encuentra documentos en los que solo aparezca “bibliotecario” porque ambas palabras comparten el mismo *stem* (“bibliotec”).

Sistema de recomendación: Es una técnica que trata de presentar al usuarios ítems de información (películas, música, libros, noticias, páginas web, etc.) sobre las que el mismo podría estar interesado. Para llevar a cabo su tarea, el perfil de usuario es contrastado con las características de los ítems, que pueden provenir del contenido del mismo (aproximación basada en el contenido, en inglés, *content-based approach*) o en el ambiente social del usuario (aproximación basada en la colaboración, conocido en inglés como *collaborative filtering*).

Social media: En español, *Medio social*. Son medios de comunicación social donde la información y en general el contenido es creado por los propios usuarios mediante el uso de las nuevas tecnologías, que permiten un fácil uso y acceso a partir de poderosas tecnologías de edición, publicación e intercambio.

Spell checker: En español, *Corrector ortográfico*. Es una aplicación de software que se utiliza para analizar textos con el fin de detectar y, de forma automática o manual, corregir faltas ortográficas.

Stop-word: En español, *Palabras vacías*, aunque sólo se lo conoce por su denominación en inglés. Es el nombre que reciben las palabras sin significado como artículos, pronombres, preposiciones, etc. que son filtradas antes o después del procesamiento de datos en lenguaje natural (texto).

Tag: En español, *Etiqueta*. Es una palabra clave no-jerárquica asignada a un ítem de información (como una imagen digital o un archivo informático). Este tipo de metadato describe el dato y permite recuperarlo a partir de una búsqueda o navegación.

Tag cloud: En español, *Nube de palabras*. Es una representación visual de las palabras que conforman un texto, en donde el tamaño de la fuente es mayor para las palabras que aparecen con mayor frecuencia. Uno de sus principales usos es la visualización de las etiquetas de un sitio web, de modo que los temas más frecuentes se muestren con mayor prominencia.

Tagging: Acción de colocar un *tag* a un recurso, como ser página web, imagen, video, etc.

Taxonomía: En su sentido más general, es la ciencia de la clasificación.

URL (Uniform Resource Locator): En español, *Localizador Uniforme de Recursos*. Una URL es una dirección que permite acceder a un archivo o recurso como ser páginas html, php, asp, o archivos gif, jpg, png, etc. Se trata de una cadena de caracteres que identifica cada recurso disponible en la WWW.

Web mining: Es una metodología de recuperación de la información que usa herramientas de la minería de datos para extraer información tanto del contenido de las páginas, de su estructura de relaciones (enlaces) y de los registros de navegación de los usuarios.

Web semántica: Es la “web de los datos”. Se basa en la idea de añadir metadatos semánticos y ontológicos a la World Wide Web. Esas informaciones adicionales (que describen el contenido, el significado y la relación de los datos) se deben proporcionar de manera formal, para que sea posible evaluarlas automáticamente por máquinas de procesamiento. El objetivo es mejorar Internet ampliando la interoperabilidad entre los sistemas informáticos usando *agentes inteligentes*.

Web spider: En español, “araña web”. Es un programa que inspecciona las páginas de la WWW de forma metódica y automatizada. El uso más frecuente que se les da consiste en crear una copia de todas las páginas web visitadas para su procesamiento posterior por un motor de búsqueda, que indexa las páginas proporcionando un sistema de búsqueda rápido. Los web spiders comienzan visitando una lista de URLs, identifica los enlaces en dichas páginas y los añade a la lista de URLs a visitar de manera recurrente de acuerdo a un determinado conjunto de reglas.

Anexo B: Morfología lingüística en IR

A lo largo de la investigación, el contenido textual de los documentos web ha sufrido algunas alteraciones sin perder ni variar nunca su significado, con el único objetivo de lograr mejores resultados al ser procesados por los distintos algoritmos de clasificación.

Las técnicas empleadas con las cuales se ha trabajado, se detallan a continuación:

Stemming

Es el proceso empleado para reducir una palabra a su raíz (también llamado *stem*). El stem no necesita ser idéntico a la raíz morfológica⁸⁹ de una palabra; es suficiente que palabras relacionadas tengan el mismo stem, incluso si este stem no es una raíz válida. El proceso de stemming es útil en motores de búsqueda para expandir las consultas o indexación, y en otros problemas de procesamiento del lenguaje natural. La razón de su implementación es que aumenta el *recall*, que es una medida sobre el número de documentos que se pueden encontrar con una consulta. Por ejemplo, una consulta sobre “bibliotecas” también encuentra documentos en los que solo aparezca “bibliotecario” porque el *stem* de las dos palabras es el mismo: “bibliotec”.

Los programas de stemming sean también conocidos como *algoritmos de stemming* o *stemmers*. La primera publicación data del año 1968 [63], logrando una remarcada importancia debido a que logró una gran influencia en trabajos posteriores de esta área.

Las aplicaciones más comunes de los algoritmos de stemming tienen que ver con la **recuperación e información**. Los stemmers son elementos básicos de los motores de búsqueda de hoy en día, ya que los usuarios que realizan la consulta “computadoras”, lógicamente también están interesados en los documentos que contengan la palabra “computadora” (sin “s” final). La eficiencia del stemming para los sistemas de consultas en inglés ha sido limitado en una primera etapa, y esto ha llevado a que los investigadores los consideraran como irrelevantes [64]. Un enfoque alternativo, basado en la búsqueda por *n-gramas* (uno de los distintos algoritmos de stemming que existen) en lugar de ítems, podría ser utilizado dando buenos resultados. También, investigaciones recientes han demostrado grandes beneficios para la recuperación de información en otros lenguajes [65, 66].

Los algoritmos de stemming pueden ser empleados en el uso de productos comerciales. Por ejemplo, muchas compañías han estado usando stemming desde finales de los 80, y han producido stemmers algorítmicos y léxicos en muchos idiomas [67].

⁸⁹ Es la unidad léxica primaria de una palabra, que contiene los aspectos más importantes del contenido semántico y no puede ser reducida a componentes más pequeños. Por ejemplo, la raíz en inglés del verbo *running* es *run*, y la raíz en español del adjetivo superlativo *amplísimo* es *ampl-*, ya que esas palabras son claramente derivadas de las raíces agregándole sufijos que no alteran la raíz de ninguna forma.

El **algoritmo de Porter** es el stemmer más reconocido. Fue desarrollado por Martin Porter en la Universidad de Crambridge en 1980 [68]. Se basa en la premisa de que los sufijos en el idioma inglés (alrededor de 1.200) están en su mayoría compuestos por una combinación de sufijos más pequeños y simples. Este algoritmo es un stemmer lineal. Básicamente, consta de cinco pasos donde se aplican determinadas reglas en cada uno de ellos. Dentro de cada paso, si una regla de sufijo coincide con una palabra, entonces las condiciones asociadas a la reglas son testeadas para determinar cuál sería el item resultante.

Una vez que se ejecutan las condiciones de una regla y esa regla es aceptada, se elimina el sufijo y se ejecuta el siguiente paso. Si la regla no es aceptada, entonces se testea la próxima regla, y continuando así hasta que haya alguna que sea aceptada (o bien se acaben). Ver figura B.1.

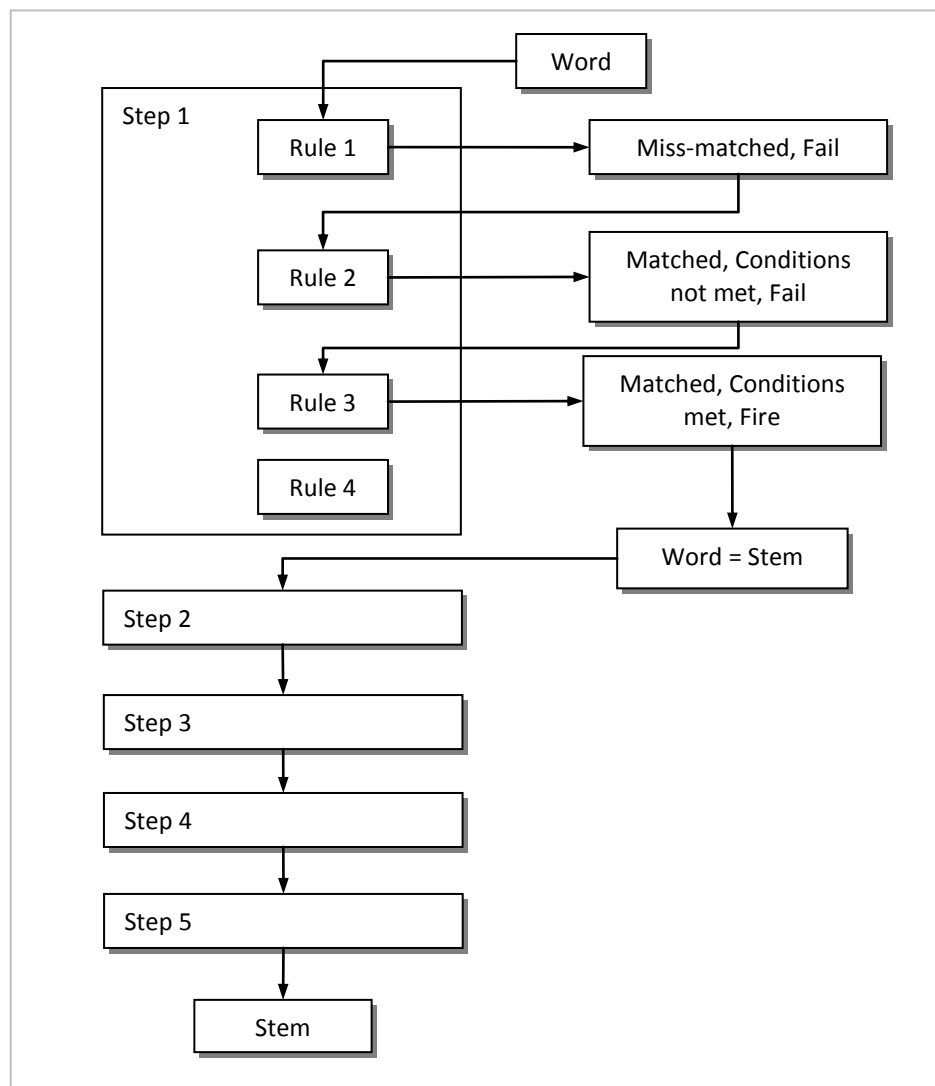


Figura B.1. Algoritmo de Porter.

Este algoritmo de stemmer es ampliamente utilizado, y existen muchas aplicaciones en donde emplearlo. Distintas implementaciones (en Java, C y PERL) pueden ser descargadas desde el sitio web creado por el propio Porter⁹⁰.

Stop words

Traducido en español como *palabras vacías*, es el nombre que reciben las palabras sin significado como artículos, pronombres, preposiciones, etc, que son filtradas antes o después del procesamiento de datos en lenguaje natural (texto) [69, 70]. A Hans Peter Luhn, uno de los pioneros en recuperación de información, se le atribuye el uso de este concepto. El manejo de *stop words* está controlado por la introducción humana y no automática de palabras.

No existe una lista definitiva de stop words que todas las herramientas de procesamiento de lenguajes naturales (PLN) procesen. A su vez, algunas herramientas de PLN evitan usar stop words para soportar búsquedas por frase. El uso de un algoritmo de stemming puede reducir parte de la base lógica o dependencia de una lista de stop words a filtrar.

Las stop words pueden causar problemas al emplear un motor de búsqueda para encontrar frases que las incluyen, especialmente en nombres como “La verdad”.

⁹⁰ <http://www.tartarus.org/~martin/PorterStemmer/>

Anexo C: Indicadores de evaluación de clasificadores

Los indicadores son muy importantes en la evaluación de la performance de los distintos clasificadores, ya que brindan una idea clara sobre el comportamiento de la colección de datos desde distintos puntos de vista. Estos valores o indicadores pueden utilizarse en modelos matemáticos para comprender y hasta predecir que podría suceder si se modifica alguna de las variables en juego. Es notable destacar que no se puede mejorar sin antes medir.

A continuación se mencionan los indicadores utilizados a lo largo del presente estudio.

Accuracy, en español *exactitud* o *precisión*, se refiere a cuán cerca del valor real se encuentra el valor medido. En términos estadísticos, la exactitud está relacionada con el sesgo de una estimación. Cuanto menor es el sesgo⁹¹ más exacto es una estimación. La exactitud de un resultado se expresa mediante el error absoluto que es la diferencia entre el valor experimental y el valor verdadero.

Formalmente, el accuracy se refiere al porcentaje de ejemplos correctamente clasificados (verdadero positivo⁹² más verdadero negativo⁹³) y es evaluado por la siguiente fórmula:

$$A_i = \frac{t}{n} \cdot 100$$

dónde t es el número de casos de ejemplo correctamente clasificados, y n es el total de números de ejemplos utilizados.

Precision y **recall** son dos métricas comúnmente utilizadas para evaluar la correctitud de algoritmos de reconocimiento de patrones. Pueden ser vistos como una versión extendida de *accuracy*. Cuando se usan *precision* y *recall*, el conjunto de posibles etiquetas para una instancia determinada es dividido en dos subconjuntos, uno de ellos es considerado “relevante” para los propósitos de la métrica. *Recall* se calcula como la fracción de instancias correctas entre todas las instancias que *realmente* pertenecen al conjunto relevante, mientras que *precision* es la fracción de instancias correctas entre aquellas en las que el algoritmo *crea* que pertenecen al subconjunto relevante.

⁹¹ Sesgo: el sesgo de un estimador es la diferencia entre su esperanza matemática y el valor del parámetro que estima. Un estimador cuyo sesgo es nulo se llama insesgado o centrado.

⁹² Verdadero positivo: Instancias etiquetadas correctamente como perteneciente a la clase en cuestión.

⁹³ Verdadero negativo: Similar al anterior, pero etiquetados incorrectamente como pertenecientes a la misma clase en cuestión.

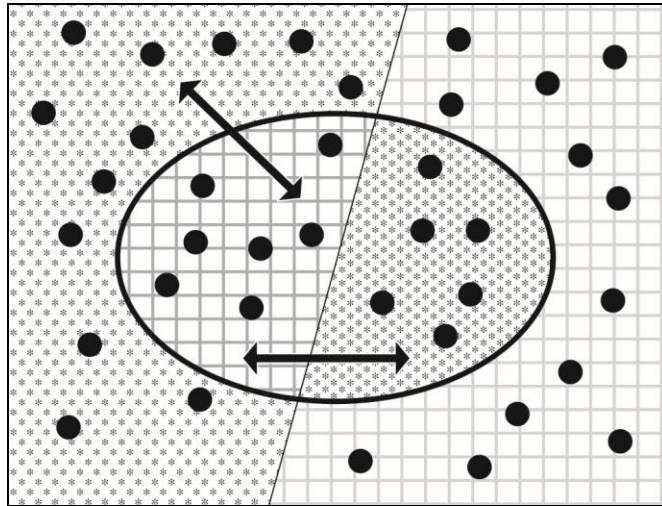


Figura C.1. Representación gráfica de Precisión y Recall.

Según la figura C.1, precisión y recall dependen de los resultados (óvalo) de una query y de su relación con todos los documentos relevantes (izquierda) y no relevantes (derecha). A mayor cantidad de resultados correctos (fondo cuadrículado), mejor. Precisión: flecha horizontal. Recall: flecha diagonal.

Precision puede ser considerada como una medida de exactitud o fidelidad, mientras que recall es una medida de integridad.

En el campo de la recuperación de información, precision y recall se definen como un conjunto de *documentos recuperados* (ej. el listado de documentos producidos por un motor de búsqueda web a partir de una query) y como un conjunto de *documentos relevantes* (ej. el listado de todos los documentos en internet que son relevantes para un determinado tópico).

Formalmente, precision y el recall se expresan como:

$$precision = \frac{t_p}{t_p + f_p} \quad recall = \frac{t_p}{t_p + f_n}$$

Donde t_p es verdadero positivo, f_p es falso positivo y f_n es falso negativo.

Los términos verdadero positivo, verdadero negativo, falso positivo⁹⁴ y falso negativo⁹⁵ son usados para comparar la clasificación de un documento en particular (la etiqueta de clase asignada al documento por el clasificador) con la clasificación correcta deseada (la clase a la que el documento realmente pertenece).

⁹⁴ Falso positivo: Instancias que han sido incorrectamente etiquetadas como pertenecientes a la clase en cuestión.

⁹⁵ Falso negativo: Instancias que no han sido etiquetadas como pertenecientes a la clase en cuestión y efectivamente no lo son.

El indicador **f-measure** (también conocido como F-score o F1) es una medida de precisión de una prueba. Tiene en cuenta tanto la precisión p como el recall r para realizar el cálculo: p es el número de resultados correctos dividido por el número de todos los resultados retornados, y r es el número de resultados correctos dividido por el número de resultados que se deberían haberse retornado. F-measure se puede interpretar como una medida ponderada de la precisión y el recall, donde ésta alcanza su mejor valor en 1 y su peor en 0.

Formalmente, f-measure se define como:

$$F = 2 \bullet \frac{precision \bullet recall}{precision + recall}$$

Bien sea una medida directa (la entregada por un aparato) o indirecta (utilizando una fórmula) existe un tratamiento de los errores de medida. Se pueden distinguir dos tipos de errores que se utilizan en los cálculos.

Por un lado, el **error absoluto** indica el grado de aproximación y da un indicio de la calidad de la medida. Formalmente se define como la diferencia entre el valor real de la medida y el valor tomado como exacto. Puede ser positivo o negativo, según si la medida es superior o inferior al valor real. Las unidades son las mismas que las de la medida.

$$e_a = v_r - v_m$$

Donde e_a es el error absoluto, v_r el valor real y v_m el valor de la muestra tomada.

Mientras que por otro lado, el **error relativo** indica la calidad de la medida. Se define como el cociente entre el error absoluto y el valor exacto. Si se multiplica por 100 se obtiene el % de error. Al igual que el error absoluto, puede ser positivo o negativo. No tiene unidades.

$$e_r = \frac{e_a}{v_r} \bullet 100$$

Donde e_r es el error relativo, e_a el error absoluto y v_r el valor real.

Por último, el **error relativo absoluto** es muy similar al *error relativo cuadrático*⁹⁶ ya que también es relativo a un indicador simple, que sería la media de los valores reales. En este caso, el error relativo absoluto se calcula como el error absoluto total normalizado, o sea dividiéndolo por el total del error absoluto del indicador simple.

Matemáticamente, el error relativo absoluto E_i de un clasificador individual i es evaluado por la siguiente ecuación:

⁹⁶ En inglés *relative squared error*, es relativo a lo que habría sido si un indicador simple hubiera sido usado. Más específicamente, un indicador simple es la media de los valores reales. De esta forma, el error relativo cuadrático toma el error total cuadrático y lo normaliza dividiéndolo por el error total cuadrático de un indicador simple.

$$E_i = \frac{\sum_{j=1}^n |P_{(ij)} - T_j|}{\sum_{j=1}^n |T_j - \bar{T}|}$$

donde $P_{(ij)}$ es el valor predicho por el programa individual i para el caso de ejemplo j (sobre un total de n ejemplos); T_j es el valor objetivo para el caso de ejemplo j ; y \bar{T} está dado por la siguiente fórmula:

$$\bar{T} = \frac{1}{n} \sum_{j=1}^n T_j$$

Anexo D: WordNet

Como se explicó vagamente en 4.7.2, WordNet es una enorme base de datos léxica del idioma inglés. Agrupa las palabras en conjuntos de sinónimos llamados *synsets*, proporcionando definiciones cortas y generales, y almacenando las relaciones semánticas entre estos conjuntos de sinónimos. El propósito del proyecto es doble: por un lado producir una combinación de diccionario y tesoro cuyo uso es más intuitivo, y ayudar al análisis automático de textos y a las aplicaciones de inteligencia artificial. La base de datos y las herramientas se han liberado bajo una licencia BSD y pueden ser descargadas y usadas libremente. Además la base de datos puede consultarse online.

A través de los años, muchas personas han contribuido al desarrollo de WordNet. Actualmente, el proyecto es llevado a cabo en el Departamento de Ciencias de la Computación de la Universidad de Princeton, y el equipo está compuesto por los siguientes miembros: George A. Miller, Christiane Fellbaum, Randee Teng, Helen Langone, Adam Ernst y Lavanya Jose.

Contenido de la base de datos

La última versión de WordNet es la 3.0. Como en el 2006, posee 147.278 términos estructurados en 117.659 *synsets*, logrando un total de 207.000 pares de palabras. En forma comprimida, son alrededor de 12 megabytes de tamaño.

WordNet establece cuatro categorías gramaticales: nombres, verbos, adjetivos y adverbios, puesto que siguen diferentes reglas. Cada *synset* contiene un grupo de palabras sinónimas o colocaciones (una colocación es una secuencia de palabras que juntas tienen un significado concreto, por ejemplo "car pool"); los diferentes sentidos de las palabras se encuentran en diferentes *synsets*. El significado de los *synsets* se clarifica con pequeñas glosas (definiciones y/o frases de ejemplo). Un ejemplo típico de *synset* con una pequeña glosa es:

`good, right, ripe -- (más conveniente o correcto para un propósito en particular; "a good time to plant tomatoes"; "the right time to act"; "the time is ripe for great sociological changes")`

La mayoría de los *synsets* están conectados con otros *synsets* a través de un número de relaciones semánticas. Uno de los más importantes es hipónimos⁹⁷, donde Y es un hipónimo de X si cada X es (una clase de) Y, ej. *canine* es un hipónimo de *dog*.

Organización de la estructura

Los sustantivos y verbos están organizados en jerarquías, definidos por hipónimos. Por ejemplo, el primer significado de la palabra *dog* podría tener la siguiente jerarquía hipónima; las palabras del primer nivel son sinónimos entre sí: algún significado de *dog* es sinónimo de algunos significados de

⁹⁷ Hipónimo, en inglés *hypernym*, es una palabra que posee todos los rasgos semánticos de otra más general.

domestic dog y *Canis familiaris*, y así. Cada conjunto de sinónimos (*synset*), tiene un índice único y comparte sus propiedades, como si se tratara de una definición del diccionario.

```

dog, domestic dog, Canis familiaris
=> canine, canid
=> carnivore
=> placental, placental mammal, eutherian ...
=> mammal
=> vertebrate, craniate
=> chordate
=> animal, animate being, beast, brute ...
=> ...

```

En el nivel superior, estas jerarquías son organizadas en tipos bases, 25 grupos primitivos para sustantivos, y 15 para verbos. Estos grupos forman *archivos lexicográficos* en un nivel de mantenimiento. Estos grupos primitivos están conectados a un nodo raíz abstracto que ha sido, desde hace un tiempo, usado por varias aplicaciones que emplean WordNet.

En el caso de los adjetivos, la organización es diferente. Existen dos polos con significados distintos, mientras que los “satélites” de sinónimos se conectan a cada uno de estos polos vía relaciones de sinónimos. Así, las jerarquías y el concepto involucrado con archivos lexicográficos, no se aplican aquí de la misma manera que lo hacen para sustantivos y verbos.

La red de nombres es mucho más profunda. Los verbos tienen una estructura más “espesa”, y los adjetivos se organizan en varios grupos distintos. Los adverbios son definidos en términos de los objetivos por los cuales son derivados, y por lo tanto heredan su estructura de los adjetivos.

Aplicaciones

WordNet ha sido usado en diferentes propósitos de sistemas de información, incluyendo *word sense disambiguation*⁹⁸, *information retrieval*⁹⁹, *automatic text classification*¹⁰⁰, *automatic text summarization*¹⁰¹, e incluso generación automática de palabras cruzadas.

Otro ejemplo destacado del uso de WordNet es determinar la similitud entre palabras. Varios algoritmos han sido propuestos, y estos consideran la distancia entre categorías conceptuales de palabras, como así también tienen presente la estructura jerárquica de la ontología de WordNet.

⁹⁸ Desambiguación del sentido de la palabra.

⁹⁹ Recuperación de información.

¹⁰⁰ Clasificación automática del texto.

¹⁰¹ Resumen de texto automático.

WordNet está conectado a varias bases de datos de la Web Semántica. WordNet es también comúnmente re-usado vía *mapping* entre categorías de WordNet (ej. *synsets*) y las categorías de otras ontologías. Más comúnmente, sólo las categorías del primer nivel de WordNet son mapeadas.