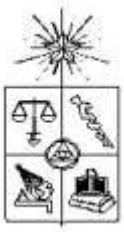


UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FISICAS Y MATEMATICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACION

DISEÑO EDITORIAL SOCIAL EN PORTALES DE INFORMACION UTILIZANDO TECNICAS DE MINERIA DE DATOS

MEMORIA PARA OPTAR AL TITULO DE INGENIERO CIVIL EN COMPUTACION

DANIEL ALEJANDRO ARTURO GOMEZ MARTINEZ

PROFESOR GUIA:

CARLOS HURTADO LARRAIN

MIEMBROS DE LA COMISION:

CLAUDIO GUTIERREZ GALLARDO
DIONISIO GONZALEZ GONZALEZ

SANTIAGO DE CHILE
AGOSTO 2007

RESUMEN DE LA MEMORIA
PARA OPTAR AL TITULO DE
INGENIERO CIVIL EN COMPUTACION
POR: DANIEL GOMEZ M.
FECHA: 20/08/2007
PROF. GUIA: Sr. CARLOS HURTADO L.

DISEÑO EDITORIAL SOCIAL EN PORTALES DE INFORMACION UTILIZANDO TECNICAS DE MINERIA DE DATOS

La sindicación de contenidos en la Web permite la generación y distribución de contenidos a través de canales de información tales como blogs, medios de prensa o comunidades digitales. En la Web actual existen millones de estos canales, los cuales producen un flujo continuo de información actualizada: artículos de texto, fotografías, videos, etc. Los canales constantemente publican resúmenes de contenido en la forma de archivos de metadatos RSS o Atom, que son recolectados por agregadores de internet, o directamente por los consumidores finales.

Un problema fundamental de estos sistemas es organizar y facilitar a los usuarios la digestión del enorme flujo de información que se puede recibir. Uno de los enfoques utilizados es el de organizar los contenidos en la forma de portadas, tratando de representar de la mejor manera los contenidos recolectados a través de portales de información. Sin embargo, la creación de dichas portadas es un problema no trivial, el cual plantea una serie de desafíos interesantes.

La detección de grupos o segmentos de contenido similar en la Web es la técnica elegida para abordar el problema planteado. En el presente proyecto se propone el desarrollo de un sistema de segmentación de artículos RSS, utilizando técnicas de clustering de documentos de manera de agrupar y reconocer los tópicos más relevantes y visualizar los resultados de aplicar el sistema sobre la Web chilena, a través del portal *orbitando.com*.

El desarrollo del proyecto fue dividido en tres etapas. En primer lugar se abordó el problema de la transformación de artículos en vectores según el modelo de espacio vectorial, considerando las características particulares de los artículos RSS. A continuación se abordó el problema de la segmentación a través de las técnicas de clustering aplicadas a una parte de la colección dentro de una cierta ventana de tiempo. Finalmente, se implementó un mecanismo de clustering en línea, que permitiría la posterior operación del sistema en el tiempo, manteniendo con vida los tópicos relevantes y detectando la aparición de nuevos tópicos emergentes.

El resultado de esta implementación fue el prototipo de un sistema que permite reconocer los principales tópicos tratados en la Web sindicada y su posterior operación mediante la agregación en línea de nuevos artículos. Los resultados obtenidos cumplieron plenamente con lo esperado, y permiten dar un primer paso en el estudio de aplicaciones que aprovechan técnicas clásicas de la minería de datos, aplicadas al caso de los artículos sindicados en la Web.

Agradecimientos

Quiero agradecer, en primer lugar, a mi profesor guía, Carlos Hurtado, por su colaboración, por su paciencia, y por haber creído en este proyecto y haberme apoyado durante estos meses de trabajo. También doy mis agradecimientos al Centro de Investigación de la Web (CIW), por apoyar la realización de esta memoria a través del financiamiento otorgado.

Pero especialmente, quiero agradecer a mi madre... por todo el apoyo que me brindó durante los años de carrera. Su ayuda y permanente colaboración fueron un aporte invaluable en todo este tiempo...
y aprovecho este medio para dar las gracias, por todo lo recibido.

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Presentación

...o como entender, en pocas palabras, que me motivó a hacer esto.

1.1. Introducción

El periodismo es una disciplina que se ha visto históricamente influenciada por los cambios que han afectado a la sociedad. Cada vez que se ha presentado un cambio social, económico o tecnológico, ha ocurrido una transformación en esta disciplina[17]. Y así como ocurrió esto con el telégrafo, la radio o la televisión, hoy se está viviendo una nueva transformación gracias a la masificación de la web, la cual ha dado pie al llamado *periodismo participativo*.

En los últimos años, la Web ha experimentado una serie de cambios en la forma en que es usada por la comunidad. Estos cambios se refieren principalmente a la manera en que interactúan quienes publican información y contenidos, y quienes la consumen (la audiencia). Es la llamada Web 2.0, la cual se basa en el principio de descentralización, de diversificación de las fuentes de información, donde los contenidos son producidos por los mismos usuarios y donde los sitios web tradicionales han dado paso a aplicaciones web destinadas a los usuarios. Ahora la comunidad participa de lo que lee, de lo que ve, de lo que se informa. Internet pasó de ser la gran biblioteca, a ser la gran conversación [30].

Esta nueva forma de publicación y difusión de la información se apoya principalmente en la llamada *sindicación* de contenidos. La sindicación es una forma de difusión donde los contenidos generados por un emisor son distribuídos por otros, de manera de expandir el alcance del medio original. En el caso de contenidos web se refiere al acto de recopilar, de manera automática, contenidos de diversos medios o fuentes. El concepto es el de delegar en la comunidad la difusión de aquellos contenidos. En este caso, cada usuario termina siendo su propia fuente de noticias (esto es lo que ocurre cuando un usuario decide a que canal suscribirse).

En este escenario, han surgido portales de sindicación, los cuales intentan recoger de alguna manera aquellos contenidos que aparecen en la red. Estos portales recopilan artículos provenientes de diversas fuentes, en la forma de resúmenes de noticias, artículos de *blogs*[1], comentarios, imágenes, etc. Dichos contenidos son organizados de diversas maneras, de forma de presentar de la mejor manera posible el gran flujo de información, agrupándolos por tema, mostrando las últimas entradas, o tratando de descubrir y presentar aquellos contenidos de mayor interés.

Figura 1.1: **Emisión**: Caracterizada por el control del medio de comunicación. Toda noticia es filtrada a través del medio antes de alcanzar la audiencia. Fuente: *Nosotros, el medio*[5]

Figura 1.2: **Red Social**: Los participantes son pares y tienen la habilidad para cambiar sus roles. Las noticias con frecuencia no son filtradas por un mediador antes de alcanzar su audiencia. Fuente: *Nosotros, el medio*[5]

[1]Blog, o web-log, es un tipo de publicación en internet, donde una o varias personas publican artículos en una especie de bitácora, permitiendo la inclusión de comentarios de parte de sus lectores.

El propósito del presente trabajo es aprovechar esta nueva forma de difusión y tratar de indagar en aquel *flujo de artículos* provenientes de canales *RSS*[2], tanto fuentes de noticias como blogs, mediante el uso de técnicas de la minería de datos. El uso de la técnica de *clustering*[3] permite realizar la segmentación del espacio de artículos, obteniendo un conjunto de grupos que representan los diversos tópicos detectados en la colección. Una vez realizada esta tarea inicial, el desafío es el de poder mantener el sistema en operación, implementando un mecanismo de *clustering en línea*, que permita la llegada de nuevos artículos, preservando la popularidad de los tópicos principales, así como permitiendo la creación de nuevos temas.

El resultado final es la definición de una portada de titulares, creada de manera colaborativa, al recoger los temas más tratados en el ecosistema de la "web sindicada". El desafió es el de alinear la tarea del diseño editorial, con los principios de la Web 2.0.

[2]RSS (*Really Simple Sindication*) es el principal formato utilizado para la generación y distribución de contenidos sindicados.

[3]El término *clustering* es una palabra de origen anglosajón: la acción de formar clusters o grupos. Dado que no es una palabra del idoma español, no será acentuada en este documento.

1.2. Justificación

Esta corriente ha tenido un explosivo crecimiento en los últimos años, provocando una proliferación de fuentes de información. Existe una gran cantidad de contenidos, noticias, artículos, y con el crecimiento de la cantidad de fuentes, el flujo de nuevos contenidos que produce la red crea un escenario donde al usuario común cada día le cuesta más aprovechar la información disponible.

Figura 1.3: Estado de la blogósfera: número de weblogs recopilados por el sitio *technorati.com*. Fuente: *The state of blogosphere*. David Sifry. [31]

Todo esto genera la necesidad de mejorar de alguna manera la forma en que ese gran flujo de información es presentado al usuario. Los consumidores de información necesitan disponer de nuevas formas de visualizar los contenidos, de nuevas formas de digerir y comprender aquel flujo de noticias, de artículos, de videos o fotografías.

En este sentido, han aparecido en la web diversas soluciones en la forma de buscadores especializados en noticias o blogs, sitios web que recogen contenidos sindicados, o sitios agregadores que permiten personalizar los contenidos recogidos. Especial interés presentan los sitios que buscan resumir contenidos, presentando al usuario una portada de contenidos emulando la portada de un medio escrito, a partir de los contenidos generados dinámicamente por la comunidad. Por *portadas*

Figura 1.4: Portada del sitio web chileno, *orbitando.com*

sociales se pueden entender aquellas portadas de sitios web surgidas a partir de las preferencias de los usuarios, buscando presentar en una sola página un resumen de interés para un dominio (o comunidad) específico, en un momento específico. El concepto de social viene de su conformación automática en base a lo que la comunidad está produciendo o consumiendo.

El caso de estudio a analizar, corresponde a una de dichas portadas sociales: *orbitando.com*, un sitio web que reúne una gran cantidad de contenidos sindicados, correspondiente al conjunto de blogs y canales de noticias de Chile. Este sitio realiza una recolección de artículos RSS relacionados con el país, tomando artículos de blogs, noticias de medios online, y canales que sindican resultados de búsquedas relacionadas con Chile, tratando de ser una ventana a la blogósfera chilena[4].

Se desearía tener la capacidad de percibir, de alguna manera, cuales son los principales temas tratados en la blogósfera. Se quisiera, más precisamente, disponer de un meta-editor periodístico, que logre identificar cuales son los titulares con las últimas noticias, o que muestre cuales son los tópicos de moda en la web, en el contexto de un idioma o región (para el caso del estudio, el dominio es Chile). Pero ese meta-editor, debiera ser un agente acorde a la filosofía de la Web 2.0,

[4]Por *blogósfera chilena*, se entiende el conjunto de blogs, medios de comunicación, y más en general, a todo contenido sindicable, producido en Chile, o que se refiere a Chile.

es decir, que esté basado en una forma de periodismo participativo, realizando un *diseño editorial social*[5].

El modelo inspirador, es el de un editor de un medio de prensa escrito, el cual recibe un conjunto de artículos y noticias, y cada día debe ordenar los temas, elegir los temas principales, evitar redundancias en lo que se publica, categorizar y ordenar las notas. Lo que se quiere hacer en este caso, es algo similar, pero basándose en toda la información disponible bajo el modelo de información sindicada.

[5]Diseño editorial se refiere a la tarea de elaboración de una portada, es lo que realizan diariamente los medios de prensa escritos. Social, se refiere al hecho de ser una portada generada automáticamente por la comunidad de usuarios, en base a los temas más populares.

1.3. Plan de Trabajo

El desarrollo del proyecto fue dividido en un conjunto de partes, las cuales fueron desarrolladas de manera secuencial. En primer lugar, se procedió al estudio del problema que motivó el desarrollo de este proyecto: el fenómeno de la web 2.0, el periodismo participativo, y la sindicación de contenidos, consultando una serie de referencias e información en línea.

Una vez definido el problema, la siguiente etapa consistió en el estudio de las tecnologías involucradas: *recuperación de información*, y los modelos de representación de documentos, y *minería de datos*, y las técnicas del clustering de documentos. Enfasis especial se dio al tema del clustering incremental, tarea que fue necesario evaluar para revisar las alternativas disponibles. En este punto fue cuando se tomaron las decisiones de las herramientas a utilizar.

La siguiente etapa consistió en realizar una clasificación manual de una muestra del conjunto de artículos de la base de datos de orbitando.com, con el fin de contar con un conjunto de referencia para las pruebas futuras, además de poder conocer el comportamiento de los artículos y los tópicos formados en la colección.

A partir de este punto, se procedió con el desarrollo del sistema. En primer lugar, se abordó el proceso de indexación de documentos, después, con el clustering estacionario del conjunto de artículos, y finalmente, se abordó la problemática del clustering en línea.

La última etapa del proyecto consistió en realizar las pruebas que permitieran validar los resultados y las decisiones tomadas en la etapa de desarrollo. Con esto, se llegó a un prototipo del sistema, el cual en el futuro sería implementado en el sistema de orbitando.com.

1.4. Objetivos

Objetivo General

Desarrollar un sistema de segmentación de artículos RSS, utilizando técnicas de clustering de documentos de manera de agrupar y reconocer los tópicos más relevantes de la colección, y visualizar los resultados de aplicar el sistema sobre la web sindicada chilena.

Objetivos Específicos

- Estudiar el estado del arte de las tecnologías involucradas.
- Generar una buena representación vectorial de los artículos RSS.
- Producir una segmentación del espacio de artículos, que logre diferenciar los diversos tópicos de la colección.
- Desarrollar un método de clustering en línea que permita reconocer la aparición de nuevos temas, o la desaparición de tópicos obsoletos.
- Desarrollar un prototipo completo del sistema, con la intención de implementar a futuro la aplicación en el sitio *orbitando.com*.

Capítulo 2

Antecedentes

...o lo que ud. necesita para comprender esta memoria.

2.1. Internet

La web comenzó como una forma de publicar contenidos en páginas web, uniendo contenido con algunas reglas de formato básicas, que junto al uso del hipertexto permitió construir la primera parte de la historia de la WWW. Es la época del crecimiento de la cantidad de contenidos disponible. La época en la que aparecieron los buscadores de contenido, como un primer intento de permitir al usuario entrar a esta gran base de datos.

Hacia el año 2000, la web experimentó un crecimiento que permitió su masificación a nivel mundial. Aparecieron los contenidos generados dinámicamente, lo que provocó una explosión en la cantidad de recursos en la red. Con esto surgió el problema de crear métodos para poder aprovechar de manera eficiente su gran masa de contenidos. La *minería web* apareció como la disciplina a cargo de investigar esta área. Era la unión de otras áreas tales como la recuperación de información y la misma minería de datos utilizada para desentrañar los secretos de grandes bases de datos. Los resultados: los primeros sitios web que comenzaron a aprovechar la información implícita en la web, a partir del comportamiente de los usuarios de la red: sus formas de navegar, la manera en que enlazaban las páginas, estudiando lo que publican. Ejemplos de esto son los clásicos buscadores: *Yahoo!*, *Google*, entre muchos otros.

La siguiente etapa de esta breve historia, es el paso natural en este concepto de aprovechar la información proveniente de los usuarios. El modelo inicial de generación centralizada de los contenidos, dio paso al nuevo modelo basado en contenidos dinámicos y el desarrollo de comunidades. Surge el concepto de *Web 2.0*. Aparecen en escena sitios como *youtube*, *flickr*, *del.icio.us*, y presentan el novedoso modelo de contenidos construidos en base a lo que los mismos usuarios producen y publican en la red: videos, fotografías, bookmarks, entre otros.

Asociado al mismo concepto, los blogs se convierten en el medio a través del cual cada usuario de internet tendría acceso a hacer público lo que quisiera, a comunicar libremente noticias de su interés, sus opiniones o lo que estime necesario, en la forma de una bitácora diseñada para permitir a casi cualquier persona publicar contenidos fácilmente. Al mismo tiempo, esos contenidos estarían sujetos a críticas y comentarios, estableciendo una verdadera conversación en torno a los recursos publicados. El resultado: la masificación del acto de publicar y comunicar información. El fenómeno de los blogs ha constituído por sí mismo prácticamente un nuevo medio de comunicación.

Por otro lado, los medios tradicionales dieron paso a los medios digitales. Primero fueron la versión online de los tradicionales medios escritos, posteriormente, aparecerían los medios de internet, sitios web que sólo tienen vida en la red. También se observa una evolución en la forma en que son publicadas las noticias: desde publicar un texto estático con imágenes, hasta contenidos que se transforman en auténticos debates online, permitiendo a los mismos lectores opinar, comentar, calificar cada contenido. Asi también se observan aquellos medios que dan cabida a que los lectores sean la propia fuente de información, surgiendo así los periódicos ciudadanos (ej: *ohmynews*).

Un factor común a estas tendencias lo constituye la sindicación de contenidos, tema presentado en el capítulo anterior. Las diversas formas de difusión de contenidos presentadas (artículos de blogs, noticias de medios, contenidos multimediales) tienen en común el concepto de constituir unidades de información pequeñas, típicamente provenientes de alguna fuente que permanentemente genera nuevos contenidos. En este escenario surge la sindicación como el medio a través del cual estas informaciones podrían llegar masivamente a la audiencia. La sindicación se basa en el uso del estándar RSS[1]. El RSS básicamente consiste en una especificación para un formato de documentos electrónicos escritos según una sintaxis XML, que permite la fácil intercomunicación entre distintas fuentes de información. Un ejemplo de documento RSS se presenta en la figura 2.1.

Cada fuente de información es reconocido como un canal (*feed*), y cada canal posee un archivo RSS, al cual se van agregando los nuevos artículos (*items*). Por otro lado, quienes reciben los contenidos pueden ser clientes RSS en los computadores de los usuarios, o bien, sitios que se dedican a recoger contenidos de la web. Tal es el caso de *technorati*, u *orbitando.com*, en Chile. El proceso a través del cual se realiza la captura de nuevos contenidos es conocido como *polling*, y consiste en realizar una comparación periódica de una versión local del archivo RSS de cada canal, y la versión en línea. Cuando éstas difieren, es porque ha ocurrido la generación de un nuevo contenido, y éste es recogido por el sistema sindicador, en la forma de un *nuevo artículo*.

[1]En realidad, RSS es el principal formato para difusión de contenidos sindicados, mas no el único. Otro estándares utilizados son *ATOM* o *RDF*. De aquí en adelante, la referencia a RSS se refiere en forma genérica a los artículos sindicados.

```
<?xml version="1.0" encoding="UTF-8"?>
    <rss xmlns dc="http://purl.org/dc/elements/1.1/" version="2.0">
        <channel>
            <title>La Tercera - Noticias del dia</title>
            <link>http://www.latercera.cl/lt3/canal/noticiasdeldia/noticia.html</link>
            <description>Ultimas informaciones publicadas por La Tercera edicion online</description>
            <item>
                <title>Lagos Weber destaca llamado de su padre a "no degradar la politica y a mantener la
                altura de miras"</title>
                <link>http://www.latercera.cl/medio/articulo/0,0,3255_5664_265436531,00.html</link>
                <description>El vocero de gobierno, Ricardo Lagos Weber, destaco las palabras de su padre,
                el ex Presidente Ricardo Lagos Escobar, quien ayer rompio su silencio frente a las criticas
                que ha recibido por su gestion como jefe de Estado.</description>
                <pubDate>Fri, 27 Apr 2007 15:19:00 GMT</pubDate>
            </item>
            <item>
                <title>Superavit fiscal llega a 2,4% del PIB en primer trimestre de 2007</title>
                <link>http://www.latercera.cl/medio/articulo/0,0,3255_5676_265433803,00.html</link>
                <description>El Fisco anoto un superavit de 2,4% del Producto Interno Bruto (PIB) en el
                primer trimestre de este año, segun informo el director de Presupuesto, Alberto Arenas, en
                una cifra que resulto superior al 2,1% registrado en el mismo periodo del año pasado.
                </description>
                <pubDate>Fri, 27 Apr 2007 15:19:00 GMT</pubDate>
            </item>
        </channel>
    </rss>
```

Figura 2.1: Ejemplo del código fuente de un archivo RSS, con dos ítems.

El escenario actual presenta una gran cantidad de fuentes de información que publican periódicamente contenidos, junto a la creciente cantidad de recursos disponibles en repositorios de comunidades on-line. Aquí aparecen las dos principales formas de acceso a la información: la búsqueda de información, y el *browsing*. Ambas buscan satisfacer la necesidad de visualización de los contenidos. En el primer caso, se cuentan los buscadores que han surgido en la forma de buscadores especializados en blogs o noticias: *technorati*, *topix*, *blogpulse*, *googlenews*. A partir de la recopilación de miles o millones de artículos, se construyen grandes bases de datos donde se puede consultar por contenidos específicos.

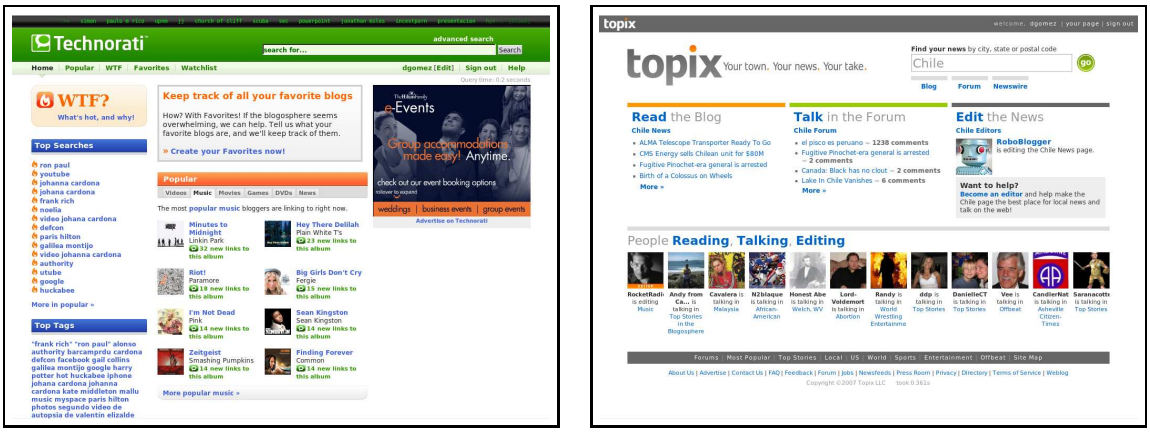


Figura 2.2: Ejemplos de buscadores de noticias/blogs.

El segundo enfoque corresponde a presentar diversas formas para facilitar al usuario la navegación por los contenidos. El *tagging* y las *tag-clouds*[2] son una alternativa muy utilizada en un intento por resumir los contenidos disponibles. De esta manera, se pueden conocer cuales son los términos más populares para un dominio en particular. Ejemplos de esto se puede encontrar en *flickr* o *del.icio.us*. El propósito es el de resumir de alguna manera los contenidos, para facilitar al usuario la navegación por la información.

Otro enfoque, busca construir una portada ideal, que muestre lo que los usuarios desean encontrar diariamente. Esta portada puede ser construida, por ejemplo, en base a sistemas de votaciones o recomendaciones. Es el caso de sitios como *digg*, *meneame*, *blogmemes*. Otros sitios simplemente realizan estadísticas en base a los clicks, buscando identificar contenidos de mayor interés. El problema común a estos casos es el de la creación de una portada, construida en base a las preferencias de los usuarios, o bien, en base a lo que los usuarios publican. Es el caso de las "portadas sociales", mencionadas en la introducción: sitios web que elaboran su portada de manera colaborativa, según la actividad de los mismos usuarios de la web.

Figura 2.3: Ejemplos de portadas sociales en internet.

[2]El tagging se refiere a la acción de etiquetar contenidos: fotografías, videos, o artículos. Posteriormente, estas etiquetas o tags pueden ser utilizados para realizar búsquedas, o para resumir la información de una colección. Por su parte, los tagclouds o nubes de tags, son un formato para presentar el conjunto de tags más populares para un dominio específico.

2.2. Indexación de documentos

La recuperación de información [3] es la disciplina encargada de la extracción de información de bases de datos textuales, ya sean bases de datos documentales, o la extracción de conocimiento de la internet. Esta disciplina ha alcanzado un gran desarrollo en la última década, motivado principalmente por sus aplicaciones en la web, y la necesidad de los usuarios de poder extraer información de interés.

Un primer problema a enfrentar en la búsqueda de información, se refiere a la característica del contenido de los documentos a analizar: documentos basados en la representación escrita del lenguaje natural. Es aquí donde surge la necesidad de traducir de alguna manera este contenido, a alguna forma en la que las máquinas puedan realizar análisis. En este sentido, la *indexación de documentos* permite transformar documentos de texto en una representación estructurada, factible de analizar mediante los métodos de la minería de datos.

El modelo de espacio vectorial (VSM) [22] es una de las primeras propuestas realizadas para la representación de documentos y es, hasta el día de hoy, la más utilizada. Este modelo busca representar documentos en forma de vectores, transformando un documento de texto en un vector que refleje la frecuencia de las palabras que lo componen. Al aplicar este principio sobre una colección de documentos, se obtiene un conjunto de vectores los cuales formarán el llamado espacio vectorial [3], formado por los vectores representantes de los documentos de la colección, y cuyas dimensiones serán los términos encontrados.

Cada documento de la colección es transformado en un vector representante, cuyos valores corresponden al peso asociado a cada palabra presente. Dichos pesos corresponden a un valor que busca representar la importancia del término en el documento. De esta manera, se tiene:

$$D_i \rightarrow d_{i1}, d_{i2}, ..., d_{ik}$$

donde $d_{ij} = w(D_i, t_j)$ representa el peso del j-ésimo término en el documento i.

La función de peso $w(D_i, i_j)$ busca representar la relación entre un documento y cada término que lo compone. Para esta labor existen diversas funciones posibles de utilizar, que dependerán del escenario en el cual se aplicará el proceso. Suponiendo: f_{ij} la frecuencia del término-j en el documento-i, $|D_i|$ la cantidad de palabras que forman el documento-i, n_j el número de documentos que poseen al término-j, N el número total de documentos, se tienen las siguientes funciones de peso:

[3] Un espacio vectorial es una representación matemática de un conjunto de elementos (vectores), los cuales admiten las operaciones de suma y multiplicación.

Binaria: el valor asignado es un valor binario, e indica la existencia o ausencia de el término j en el documento.

$$w(D_i, t_j) = \begin{cases} 1 \text{ si } f_{ij} > 0 \\ 0 \sim \end{cases}$$

Frecuencia del término: corresponde a la frecuencia absoluta del término dentro del documento.

$$w(D_i, t_j) = f_{ij}$$

TF, Frecuencia normalizada del término: similar a la anterior, sólo que el valor contabilizado es normalizado por el tamaño del documento.

$$w(D_i, t_j) = TF = \frac{f_{ij}}{|D_i|}$$

TF-IDF: La frecuencia normalizada de cada término, ponderado por la frecuencia inversa del término en la colección.

$$w(D_i, t_j) = TF * IDF = \frac{f_{ij}}{|D_i|} * -log(\frac{n_j}{N})$$

La idea intuitiva de esta última función, es la de premiar la aparición frecuente de un término en un documento, y castigar el hecho de que ese término aparezca demasiado en el resto de la colección, lo que indicaría que el término no ayuda a discriminar unos documentos de otros.

El resultado de aplicar esta transformación sobre una colección de documentos, será un conjunto de vectores pertenecientes a un espacio vectorial, cuyas dimensiones corresponderán a las palabras encontradas en todos los documentos. Es en este punto cuando aparece el gran problema que enfrenta este modelo: la alta dimensionalidad del espacio (del orden de miles de palabras para una colección de apenas unos mil documentos).

La *Ley de Heaps*[15] muestra empíricamente el fenómeno del crecimiento del vocabulario, para una colección creciente de documentos. A medida que aumenta el número de documentos considerados, crece la cantidad de palabras distintas, es decir, crece la dimensionalidad del espacio vectorial. Por otro lado, la *Ley de Zipf*[28], también conocida como Ley de Potencia, muestra que la distribución de las palabras distintas en una colección de documentos sigue un comportamiento descrito por: ranking del término-t $*$ frecuencia del término-t $= cte.$. Esto se traduce en una distribución altamente desequilibrada, donde pocas palabras poseen una muy alta frecuencia (son muy utilizadas), y muchas palabras son usadas muy pocas veces. Sin embargo, un resultado muy útil indica que entre las palabras encontradas, existe un cierto rango de frecuencias en el cual se encuentran las palabras que mejor describen el contenido de los documentos de la colección[18].

Otro fenómeno corresponde a la aparición de distintas formas sintácticas para palabras que representan un mismo concepto. Es el caso de las formas verbales del español, o de las distorsiones

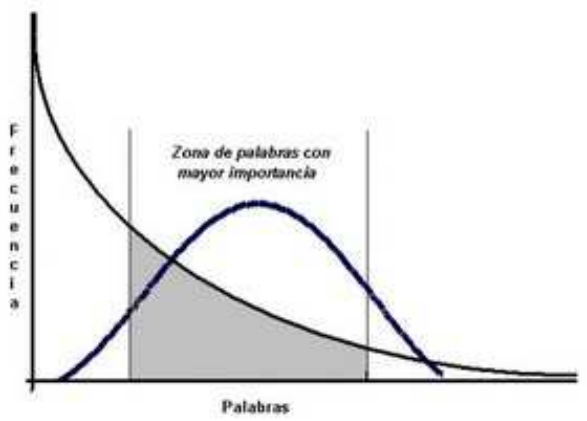


Figura 2.4: Distribución de las palabras en una colección de documentos.

por género o número. Sería deseable disponer de un mecanismo que permita distinguir las distintas formas que puede presentar un concepto del lenguaje.

En base a estos hechos, es posible plantear mecanismos de post-procesado de las palabras encontradas, con el fin de obtener un conjunto reducido que preserve la información contenida en la colección. Las principales acciones que se pueden realizar en este sentido, son:

Eliminación de palabras comunes (stopwords): En cada idioma siempre existirá un conjunto de palabras de uso común, tales como artículos, pronombres o adverbios[19]. Estas son palabras de poca importancia, ya que no aportan mayor significado. Si se construye una lista de estas palabras, se podrán filtrar de la lista de palabras producida por el proceso de indexación. Este proceso busca eliminar la cabeza de la distribución de palabras, donde se ubican todas aquellas palabras de uso común que no aportan significado discriminatorio a los documentos.

Reducción a la raíz (stemming): Para solucionar el problema de la multiplicidad de formas para un mismo concepto (las diferencias en género, número o forma verbal), se utiliza un método conocido como *stemming*, el cual busca reducir a su raíz las palabras encontradas. Básicamente consiste en un procedimiento de normalización de palabras, eliminando generalmente las terminaciones de éstas. Una forma de implementar este proceso es mediante algoritmos de stemming (por ejemplo, el *algoritmo de Porter*[20][33]). También existen otras formas directas de simplificación, consistentes en el uso de diccionarios.

Poda de la lista de palabras.: Una vez construida la lista definitiva de términos, se observa que éstas se distribuyen siguiendo la Ley de Zipf. Esto motiva la última reducción posible: podar la lista de palabras, eliminando aquellas que tienen una aparición menor a un cierto valor. Esto corresponde a eliminar la cola de la distribución mostrada en la figura 2.4. Eliminando las palabras con una frecuencia menor que el umbral de corte, se puede reducir drásticamente el tamaño del vocabulario.

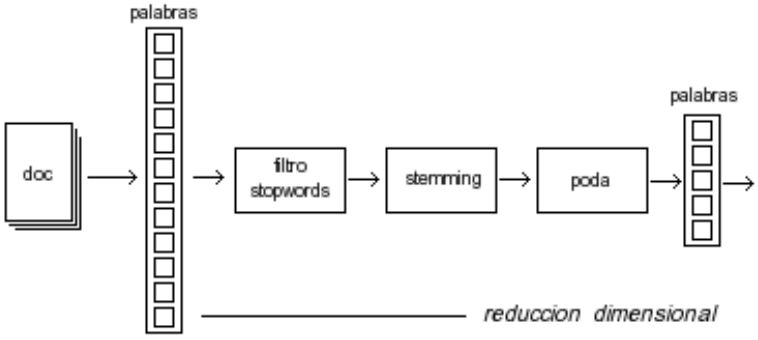


Figura 2.5: Esquema del proceso de reducción dimensional del conjunto de palabras.

2.3. Técnicas de clustering

El clustering de documentos es el proceso utilizado para dividir en grupos una colección de documentos, permitiendo evidenciar relaciones que de otra manera no serían evidentes. El análisis de clustering puede ser visto como un proceso de aprendizaje no supervisado, que a diferencia de la clasificación, no posee categorías (etiquetas) predefinidas.

A partir de un conjunto de n elementos, se desea particionar la colección en un conjunto de k subconjuntos $C_1, C_2, ..., C_k$. El objetivo es que los elementos de cada grupo sean similares a los demás elementos de su grupo, y al mismo tiempo, sean diferentes a los elementos de los demás subconjuntos.

Existe una diversidad de algoritmos que permiten realizar la tarea de clustering, y su elección dependerá de cada caso. En algunos casos se busca maximizar la calidad del resultado, en otros, es necesario considerar los temas de eficiencia, tanto tiempo como espacio. Según la manera en que se forman los grupos, estos algoritmos pueden ser clasificados en: *jerárquicos*, *de particionamiento*, *basados en densidad*, entre otros [12][4].

Algoritmos de particionamiento: Estos realizan la tarea de dividir el conjunto de datos en clases, asignando una pertenencia (o probabilidad de pertenencia) a cada uno de los elementos del espacio. Esta asignación se irá perfenccionando a medida que itera el algoritmo. Ejemplos de este caso son los algoritmos de *K-means*, *K-medianas*, *isodata*, *E-M*, entre otros.

El algoritmo de *K-means* supone una cantidad dada de k-clusters, y buscará localizar las k-medias de los clusters. Estas medias corresponderán a los *centroides* de cada agrupación, teniendo:

$$centroide_j = \frac{\sum_{i=1}^{n} \vec{x}^i}{n}$$

para el cluster j, compuesto de n elementos $\vec{x}_i$.

El algoritmo comienza con definición (típicamente aleatoria) de k medias, y la asignación de todas las instancias a su media más cercana. En el siguiente paso, nuevamente se reubicarán las k medias, esta vez buscando acercarse al centro de cada uno de los grupos recién definidos, y luego, se volverá a asignar cada vector del conjunto a la media más cercana en el espacio. Lo que se busca, es minimizar una función objetivo, definida por:

$$I = \sum_{r=1}^{k} \sum_{d_i \in S_r} \|d_i - C_r\|^2$$

El algoritmo se detendrá en el momento en que se haya alcanzado suficiente estabilidad en la posición de las medias de los clusters, o en otras palabras, cuando el error alcance su mínimo. Es interesante notar que este mínimo alcanzado sólo es un mínimo local, y su valor dependerá de las condiciones iniciales: la asignación inicial de medias. Es por ello que generalmente el algoritmo es ejecutado varias veces, y entre esas soluciones se elige la mejor, lo cual aproxima bastante el resultado global deseado.

Algoritmos jerárquicos[10]: Estos algoritmos se basan en una la agrupación jerárquica de los elementos, donde su estructura es construida a través de un proceso iterativo, que genera un árbol (*dendograma*) de los elementos relacionados según su grado de cercanía. Esta construcción puede comenzar con un único cluster que contiene a todos los elementos (*algoritmos divisivos*), y en cada iteración los clusters existentes son divididos, generando una nueva agrupación. O bien, se puede comenzar con tantos clusters como elementos (*algoritmos aglomerativos*), y cada paso consiste en ir eligiendo pares de clusters que se unirán. El algoritmo finalmente se detendrá según algún criterio de parada, por ejemplo, al alcanzar el número de clusters deseado.

El algoritmo de HAC (*Hierarchical Agglmerative Clustering*) corresponde a un caso de representación de los datos mediante un dendograma, el cual se irá construyendo a medida que avanza el algoritmo. Para un conjunto de n instancias, se tendrá n clusters, cada uno asociado a un elemento. A continuación, se elegirá unir los dos clusters más cercanos, según alguna regla de distancia inter-clusters. Ejemplos de esta regla pueden ser la distancia entre las medias de los clustesr, o bien, la distancia entre los bordes de cada grupo. De esta manera, el par de clusters elegidos se unirá para formar uno solo en el nivel superior del árbol. Finalmente, el algoritmo se detendrá cuando se

satisfaga el criterio de parada.

Los algoritmos jerárquicos tienden a tener mejores resultados que el caso de los algoritmos de particionamiento, ya que realizan la agrupación siguiendo un mecanismo natural de creación. Sin embargo, presentan el problema de su elevado costo de computación.

Algoritmo basados en densidad: Un tercer grupo de algoritmos corresponde a aquellos basados en densidad. Tienen la ventaja de que sus costos son lineales respecto a la cantidad de objetos [12], y puede encontrar clusters de formas arbitrarias, además que presentan un buen manejo del ruido. DBSCAN es un ejemplo de esta categoría. El algoritmo encuentra todos los puntos que estan "conectados por densidad", es decir, que para un radio dado, superan cierto umbral de frecuencia. El algoritmo aplica este criterio de forma tal de cubrir a todos los elementos de la colección: algunos formarán grupos y otros, simplemente no se asociarán con ninguno (ruido).

Algoritmos basados en grafos: Otra clase de algoritmos son aquellos que se basan en la representación de un grafo de similaridad, donde cada nodo representa un elemento/documento, y los pesos de los arcos modelan la similaridad entre pares de elementos. Luego de eliminar algunos arcos del grafo inicial, el problema se reduce a encontrar componentes conexas. El incoveniente es su costo temporal, del orden de n^2.

Otros esquemas de clustering son, por ejemplo, aquellos basados en modelos, donde se buscan los modelos para cada cluster que mejor ajusten los datos. O también, aquellos basados en redes neuronales, caso de los mapas autoorganizativos de Kohonen.

La operación de clustering se basa en la noción de distancia o similaridad entre vectores. El clustering permite utilizar diversas funciones de cercanía, tales como la distancia euclidiana, u otras, como la distancia Manhattan. Sin embargo, para el caso de clustering de documentos, los resultados demuestran que es más adecuado el uso de la llamada *distancia coseno*, la cual mide el coseno del ángulo formado entre los vectores [27]. De esta manera se tiene:

$$d(\vec{v_1}, \vec{v_2}) = cos\theta = \frac{\vec{v_1} * \vec{v_2}}{\|\vec{v_1}\|\|\vec{v_2}\|}$$

que medirá la similaridad entre v_1 y v_2.

Una vez definida la noción de distancia, se ha completado la definición del espacio vectorial de documentos. Cada documento está representado por un vecetor, y la distancia entre vectores corresponde a la distancia coseno. Este resultado se puede apreciar en la figura 2.6.

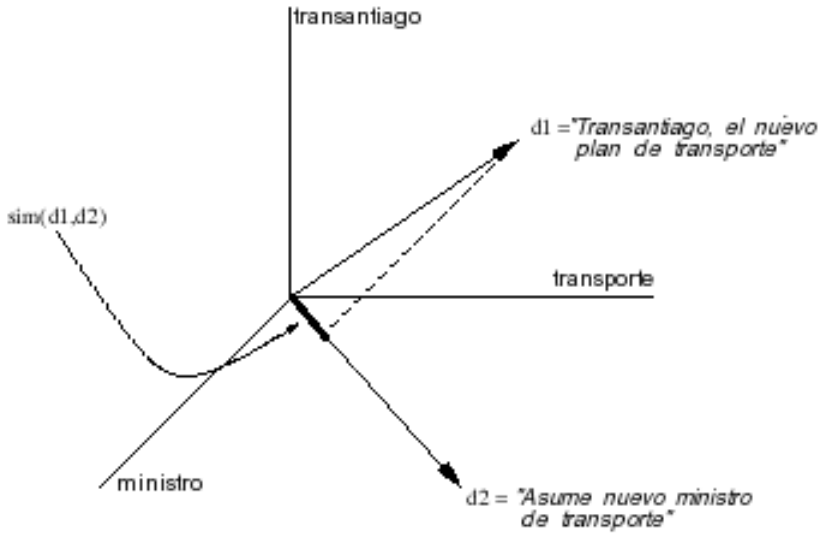


Figura 2.6: Representación de dos documentos en el espacio vectorial y la medida de su similaridad.

Clustering incremental

Lo expuesto hasta ahora, funciona bien para realizar todos los estudios necesarios sobre una colección bien definida de documentos. Sin embargo, en muchos escenarios surge la necesidad de actualizar la agrupación realizada, a partir de nuevos documentos que se van agregando a la colección, por ejemplo, registro de llamadas entrantes de un callcenter, o el rastreo de posts de un newsgroup. Estos documentos pueden ser parte de un flujo constante de nuevas entradas, lo que motiva el estudio de los llamados *algoritmos de clustering incremental*.

Este tema es un área relativamente poco explorada hasta el día de hoy. Los primeros resultados al respecto aparecen con [9], que presenta el algoritmo de *COBWEB*[11], un algoritmo de clustering conceptual el cual es naturalmente incremental. Posteriormente, en los 90's, surgen propuestas para actualizar resultados de datamining sobre bases de datos que cambian en el tiempo [8]. En [8], se propone una variación a un algoritmo basado en densidad: *Incremental DBSCAN*[7]. La mayoría de estos planteamientos se basan en una estructuración jerárquica de las agrupaciones.

Un segundo grupo de algoritmos incrementales se basa en la representación plana de los clusters[14]. *Single-pass* plantea la idea simple de procesar cada nuevo documento, comparándolo con cada uno de los clusters existentes, se elige el más cercano y, si dicha distancia es menor que cierto valor umbral, se asigna a éste. En caso contrario, el documento formará un nuevo cluster, de tamaño 1. Ajustando el umbral, se pueden obtener clusters de diferentes grados de granularidad [25]. Una versión generalizada de este algoritmo es *GenInc*, presentada en [13]. Otro enfoque es el propuesto por *K-nearest neighbor clustering*, el cual calcula la similaridad del nuevo documento

a cada uno de los documentos de la colección, eligiendo los primeros k documentos más cercanos. El cluster elegido será aquel que contenga a la mayor cantidad de documentos, entre los k seleccionados. Por último, en [6] se presenta una versión incremental de *k-means*, donde se estudia el problema de minimizar la suma de los cuadrados de las distancias a las medias, para el caso de un flujo entrante de datos.

La aplicación del clustering incremental a flujos de datos provenientes de internet, tiene su foco de atención en el proyecto TDT: *Topic Detection and Tracking* [1]. TDT es una iniciativa patrocinada por DARPA [4], donde participan diversos grupos de investigación[5]. Su principal objetivo es investigar el hallazgo de nuevos eventos o tópicos, y su seguimiento en el tiempo, en flujos continuos de documentos, por ejemplo, artículos de newsgroups. El primer estudio formal se realizó sobre una colección de 15863 artículos, provenientes de la agencia Reuters y CNN, en el período de Julio 1994 a Junio 1995.

Las investigaciones realizadas en el proyecto TDT son desarrolladas de manera independiente en cada uno de los grupos de investigación. En particular, la tarea de detección en linea de nuevos eventos es abordada de distinta manera en cada grupo [1] [21]. En CMU[6] se utiliza una regla basada en umbrales para decidir si una nueva historia es agregada a un cluster existente o si genera un nuevo grupo. Existe un primer umbral (como en single-pass), que determina si agregar o no un nuevo documento a un cluster ya encontrado, y un segundo umbral, que determina si el nuevo vector está lo suficientemente lejano como para ser considerado un nuevo cluster, o bien, está cerca de un cluster, pero no lo suficiente para ser parte de él. Además, se ha estado investigando la incorporación de la variable temporal al momento de medir las distancias, creando una función de similaridad que depende linealmente del tiempo. En el caso de UMass[7] se utiliza un umbral variable, donde cada nuevo documento genera un vector-consulta, y la similaridad entre esa query y el documento original entrega una cota para el umbral. Además, se utiliza una componente temporal para favorecer la cercanía entre documentos más cercanos en el tiempo, buscando favorecer la aparición repentina de eventos. De hecho, se centra en la búsqueda de "desastres". El grupo Dragon[8] se preocupa más de reconocer tópicos de discusión en la colección, basándose en modelos de reconocimiento de lenguaje natural.

[4]DARPA: Defense Advanced Research Projects Agency, una agencia del departamento de defensa de los Estados Unidos, encargada del desarrollo de nuevas tecnologías.

[5]Los participantes del proyecto TDT son parte de la universidad de Carnegie Mellon, la universidad de Massachusset, y la empresa Dragon Systems.

[6]Carnegie Mellon University

[7]University of Massachusset Amherst

[8]Dragon Systems

Capítulo 3

Estudio del dominio

...o como darse cuenta de lo tortuoso de clasificar documentos a mano.

Una vez estudiadas las técnicas involucradas en el desarrollo del proyecto, la siguiente tarea consistió en realizar lo que en minería de datos se conoce como la fase de *data understanding*, es decir, el estudio de los datos a analizar. Dicho estudio consistió en construir un conjunto de muestra clasificado manualmente, determinando el conjunto de tópicos que surgen en la colección muestreada. El objetivo es el de poder conocer el comportamiento de los tópicos, y obtener estadísticas que sirvan de referencia a la hora de tomar decisiones. Además, este conjunto será utilizado como conjunto de prueba, con el fin de poder comparar los resultados de los experimentos realizados.

En primer lugar, se presenta el escenario sobre el cual se trabajó: *orbitando.com*, y se muestra parte del modelo de datos en el cual viven los artículos, además de algunas estadísticas básicas. En la segunda parte, se muestra el estudio realizado sobre los datos: la clasificación manual de 2000 artículos, correspondientes a los primeros tres meses del año 2007.

3.1. orbitando.com

Orbitando.com es un sitio web desarrollado a principios del año 2006, el cual nació como un simple recolector de contenidos sindicados. Estos contenidos tenían la particularidad de ser contenidos creados en Chile, o bien, que hacían referencia a Chile: noticias, posts de blog, podcast, fotografías, videos. Inicialmente, se agregaron canales de manera explícita (principalmente los resultados de búsquedas predefinidas en sitios de noticias), y después, se permitió a los usuarios agregar sus propios canales (por ejemplo, blogs). De esta manera, *orbitando* fue acumulando canales, y por lo tanto, acumulando una gran cantidad de artículos, los cuales, al cabo de un año, forman una base de datos con más de dos millones de artículos.

Esta base de datos es el punto de partida del estudio del presente proyecto. Cada canal al cual está suscrito el sistema, se encuentra almacenado en una tabla llamada CHANNELS, que contiene los datos de cada uno de los *feeds*(canales) que alimentan al sistema. Con respecto a los artículos, cada nuevo artículo llegado al servidor es almacenado en la forma de un registro ITEM. Este ítem tiene asociados un conjunto de atributos, correspondientes a las partes que lo componen: titular, descripción, fecha de publicación, fecha de recepción, además de la referencia a su canal de origen, entre otros datos; este registro se crea al momento de llegar cada nuevo artículo. A continuación, el sistema realiza algunas labores de extracción de información, o pre-procesado, tales como la extracción de términos, y la clasificación del artículo en categorías (que en el caso general, se obtiene simplemente a partir del canal de origen). El resultado de este paso, es un conjunto de metadatos, que son almacenados en la tabla ITEM_METADATA. Esta tabla será finalmente el "universo" del cual se obtendrán los artículos a ser procesados. Un breve esquema de la base de datos es el siguiente:

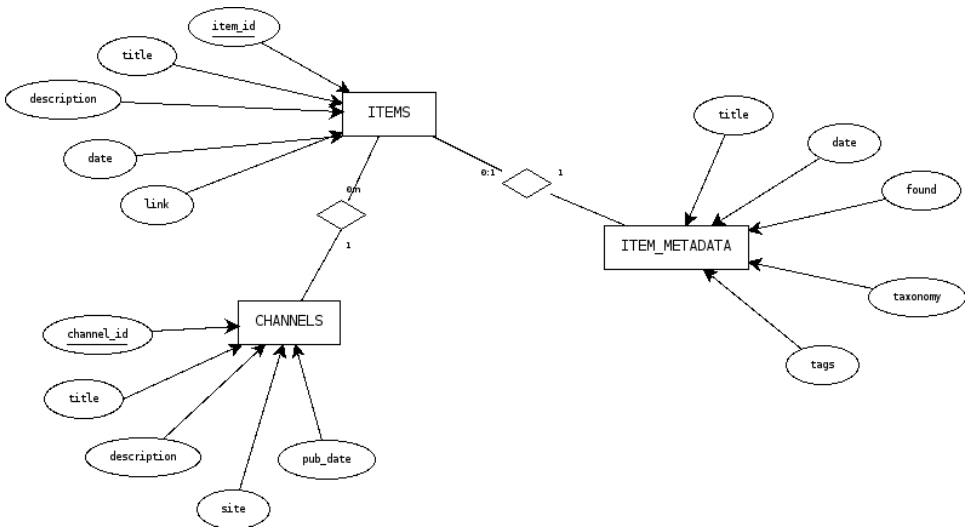


Figura 3.1: Breve esquema de la base de datos de *orbitando.com*.

La base de datos original cuenta con una cantidad que asciende a cerca de 2.000.000 de artículos, hasta el mes de mayo de 2007. Esta es la cifra que arroja el conteo simple de entradas en la tabla ITEMS, la cual almacena los artículos entrantes. Sin embargo, se observó que en esta tabla existe una gran cantidad de artículos repetidos, debido a errores producidos en una de sus principales fuentes: *google news*. Para efectos de este estudio, la cantidad de artículos disponibles fueron las almacenadas en la tabla ITEM_METADATA, la cual posee 420mil registros. A su vez, se decidió tomar una muestra de la base de datos, recogiendo los artículos correspondientes a solo tres meses: enero, febrero y marzo de 2007. Esta muestra sería la utilizada para realizar el desarrollo del proyecto, con lo que finalmente, el universo de artículos quedó reducido a 154861, correspondiente a los artículos válidos de los tres meses mencionados. Al graficar el histograma de frecuencias, se puede apreciar su distribución en el tiempo, evidenciando una leve tendencia creciente, producto de la constante adición de nuevos canales al sistema.

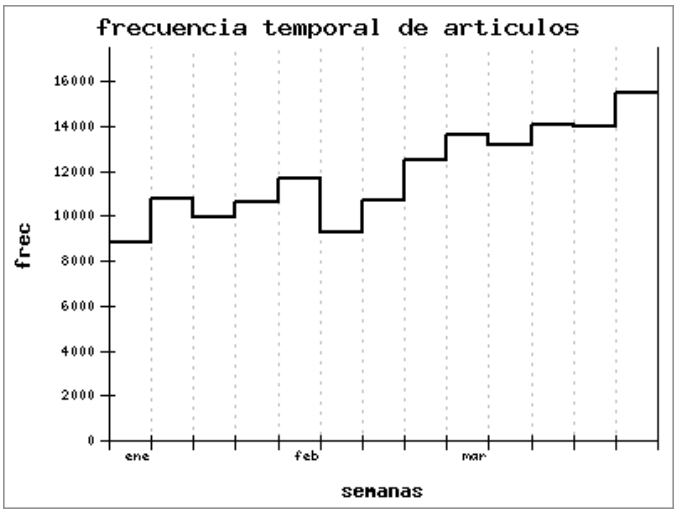


Figura 3.2: Gráfico que muestra la distribución de artículos en el período a considerar para el estudio.

Los canales que alimentan al sistema, en la actualidad ascienden a más de 2000 feeds. La principal fuente de artículos corresponde al conjunto de medios de prensa-online, no sólo aquellos que publican contenidos sindicados, sino casi cualquier medio de prensa en internet que publique informaciones sobre Chile, gracias a los resultados recogidos por sitios como *googlenews*, *Topix.net*, o *MSN Search News*, cuyas búsquedas pueden ser recogidas en formato RSS. Otras fuentes importantes de artículos son algunos medios de prensa nacional, tales como *La Tercera online*, y *Emol.com*, además de las búsquedas que arroja *Flickr* asociadas al tag *Chile*. El siguiente es el listado de los canales más populares en la colección (es decir, los canales que aportan con la mayor cantidad de artículos).[1]

[1]Estas cifras fueron obtenidas sobre el conjunto de 154861 artículos seleccionados.

canal	artículos asociados
Flickr : chile - Everyone's Tagged Photos	36462
La Tercera : Noticias del día	7955
Topix : Search for çhile"	6674
Google Noticias : tag chile	3633
MSN Search News : Live Search News: chile	2984
LA TERCERA	2958
Google News Chile - Chile	2479
Yahoo pipes: Chile News	2277
EIN News: Chile News	1785
Google News : chile	1721

Cuadro 3.1: Listado de los canales más populares de orbitando.com

Como se aprecia en la tabla 3.1, el canal que tiene asociado una mayor cantidad de entradas, es la búsqueda del tag *chile* en las fotografías de *Flickr*, seguido por las "noticias del día" en el diario *La Tercera*, y los resultados de la búsqueda del término "chile" en *Topix.net*. Por otro lado, si se relaja la condición de pertenencia a un canal, se pueden totalizar los artículos según el sitio de origen, de una manera un poco más categorizada. De esta manera, se obtiene algunos otros resultados (solo a modo de referencia):

sindicación de búsquedas	
news.google	13090
yahoo.com	2820
topix.net	6921
einnews.com	1801

contenidos multimediales	
flickr.com	36489
fotolog.com	1373
youtube.com	978
video.google.com	313
podcaster.cl	315
podcasts radio zero	277

medios nacionales	
emol.com	2502
tercera.cl	10961
nacion.cl	1014
elmostrador.cl	404

medios regionales	
el rancahuaso (Rancagua)	1116
el morrocotudo (Arica)	1098
el observatodo (Coquimbo)	856
el aMaule (Talca)	887
gran valparaiso (Valparaiso)	295
elsur.cl (Concepción)	1930

blogs	
blogspot	21626
atinachile	555
blogger	182

Cuadro 3.2: Principales fuentes de artículos, agrupadas por categoría.

Lo mostrado en estas estadísticas permite tener una visión del contenido disponible en el sitio web. Se observa que la mayoría de los artículos corresponden a noticias sindicadas de medios de comunicación (medios extranjeros o de circulación nacional), ya sea directamente o a través de un buscador. Pero también hacen un importante aporte los artículos provenientes de los llamados medios ciudadanos, los que en Chile representan a diversas ciudades del país. Por otro lado, los blogs marcan una presencia cercana al 15 % del total de artículos. Estos artículos son los que permiten

“medir” las publicaciones que típicamente son opiniones respecto a las noticias, eventos o temas de actualidad tratados por los medios noticiosos.

A partir de la información presentada, se pudo determinar más precisamente el dominio de los datos que finalmente serían utilizados en los análisis del sistema. Primero que todo, se determinó no considerar a los artículos provenientes de *Flickr*, debido a que en el caso general, se trata sólo de fotografías, sin un contexto muy definido . En el mismo sentido, se decidió filtrar los artículos provenientes de *fotolog.com*, ya que la naturaleza de sus artículos se centra en torno a una fotografía, muchas veces sin más información que la propia imagen. Finalmente, se decidió excluir aquellos artículos catalogados como noticias en idioma inglés (existe una categoría *english* que etiqueta artículos provenientes de medios de habla inglesa), ya que el procesamiento de palabras (stemming, filtrado de stopwords) se basa en propiedades exclusivas del idioma español, por lo que textos en idioma inglés terminarían representando ruido para el sistema.

3.2. Clasificación del conjunto de prueba

El objetivo de esta parte del proyecto es el de estudiar primero la colección de artículos disponibles, y en segundo lugar, realizar una clasificación manual de artículos, con el fin de poder tener una visión de los resultados que se desean producir con el sistema desarrollado. Para ello, se implementó un sistema simple que permitiera realizar una visualización y categorización de artículos, en base a tópicos.

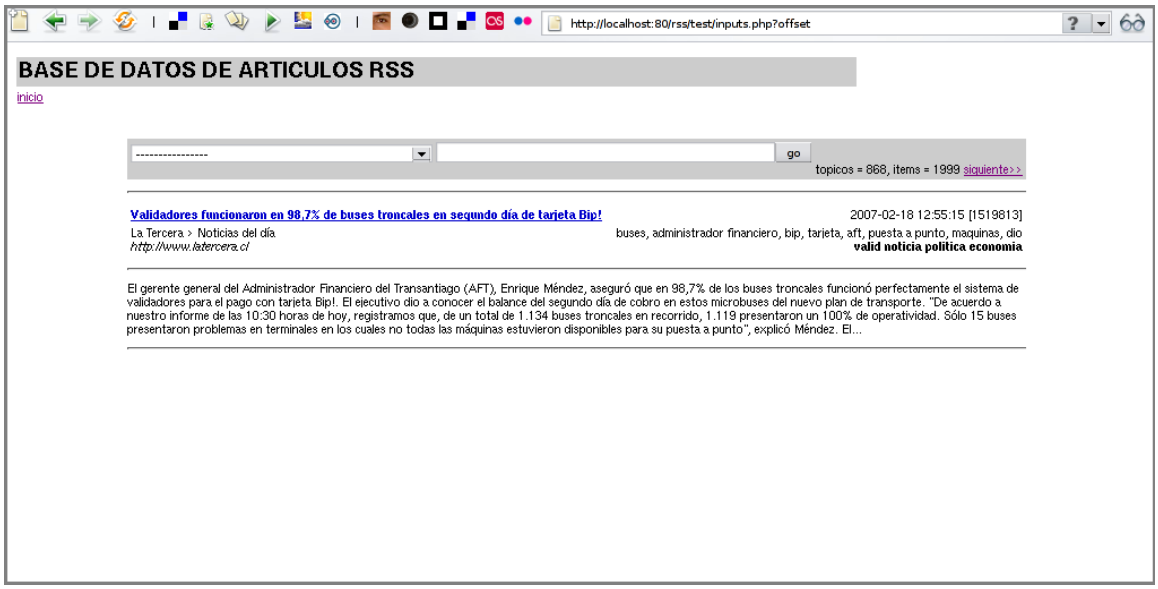


Figura 3.3: Pantalla de catalogación de artículos, del sistema de clasificación manual.

El procedimiento utilizado fue el siguiente: la pantalla de catalogación selecciona al azar[2] un artículo de la colección, lo muestra en pantalla, y permite realizar una de dos tareas: asignarlo a un tópico existente, o darle nombre a un nuevo tópico y asociar el artículo a éste. Esta labor fue realizada para un total de 2000 artículos, lo que permitió construir un conjunto de artículos categorizados, y generar una colección de tópicos. El criterio utilizado fue el de considerar un *nuevo tópico* a un tema que represente algo que no pueda ser parte de los tópicos ya creados. Esto incluye noticias o eventos ("*sudamericano sub 20*", "*festival de viña del mar*"), y temas de discusión permanente ("*Pinochet*", "*torneo de apertura de fútbol*"). La labor de clasificación fue realizada en varias iteraciones, es decir, en una primera revisión se catalogaron los artículos de manera libre, y *a posteriori*, se realizó una exhaustiva revisión de los tópicos creados y sus relaciones, con el fin de eliminar redundancias, y refinar el resultado de la clasificación realizada.

El resultado de la clasificación, es la segmentación del conjunto muestreado en un conjunto de 792 tópicos, de los cuales se muestran los primeros 48 en la tabla 3.3.

[2]La selección de artículos fue diseñada de manera de conseguir una distribución uniforme de los documentos extraídos en el período de tiempo del estudio. Para ello, se cuenta con una tabla temporal con todos los id válidos (después de filtrar) de la cual se selecciona cada vez, un identificador aleatorio.

tópico	frec.
transantiago	52
sudamericano de futbol sub 20	37
michell bachelet	27
copa davis chile-rusia	24
resultados club u.de chile	22
conflicto limitrofe con peru	21
fernando gonzalez	20
chiledeportes	18
colo colo	18
copa america de futbol	18
gabriela mistral	17
torneo de apertura de futbol	16
augusto pinochet	15
partido futbol chile-brasil	15
conflicto belico en irak	14
contrataciones club u.de chile	14
universidad de chile	14
alberto fujimori	13
club universidad catolica	13
festival de vinna	13
educacion	12
andre agassi	11
energia	11
festival de cine	11
literatura	11
escolares	10
futbol nacional	10
mundial junior de tenis de mesa	10
pelicula	10
sudamericano de futbol sub 17	10
vacaciones	10
acidente automovilistico	9
blogs	9
festival de cine de cartagena	9
futbol femenino	9
futbol sudamericano	9
gabriel garcia marquez	9
periodismo	9
video juegos	9
atp vinna del mar	8
cine latinoamericano	8
farandula	8
hugo chavez	8
pildora del dia despues	8
poesia	8
protestas	8
tecnologia	8
trafico de drogas	8

Cuadro 3.3: Listado de los primeros 48 tópicos encontrados en la colección.

A partir de esta clasificación, se obtuvo una primera aproximación a los resultados que se desea obtener con el sistema en operación: para una colección de 2000 artículos, los primeros tópicos agrupan una cantidad en el rango de 20 a 50 documentos. Por otro lado, la muestra utilizada representa aproximadamente al 2 % del total, con lo que los resultados de este estudio se pueden multiplicar por 50 para obtener una aproximación bastante cercana a los valores reales. De esta manera, se concluye que, para un conjunto de 100.000 documentos observados, los primeros tópicos debieran concentrar una cantidad de entre 1000 a 2500 documentos.[3]

Al graficar la distribución de la frecuencia de los tópicos, se aprecia la formación de una ley de potencia que describe la manera en que éstos se concentran. Esto refleja algo esperado: los primeros tópicos (en especial, los dos primeros) concentran una gran cantidad de artículos, mientras que existen muchos otros tópicos que poseen muy pocos documentos asociados (solo uno o dos).

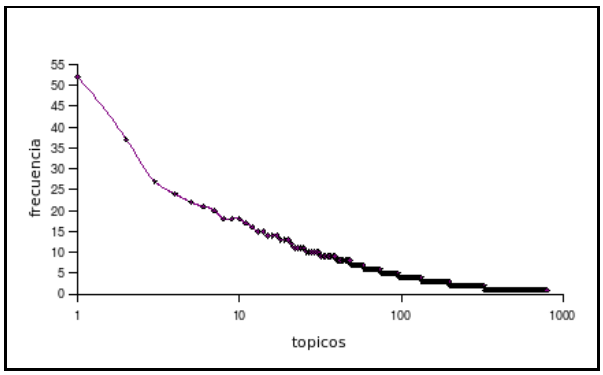


Figura 3.4: Distribución de los tópicos encontrados en la muestra.

Un resultado interesante se obtiene, si se totalizan las frecuencias en el tiempo, de los artículos de cada tópico, agrupándolos por semana. Se obtiene así, un histograma que muestra la evolución del tópico en el tiempo, para el período de tres meses del estudio. Esto permite una primera visión de la dinámica de los tópicos.

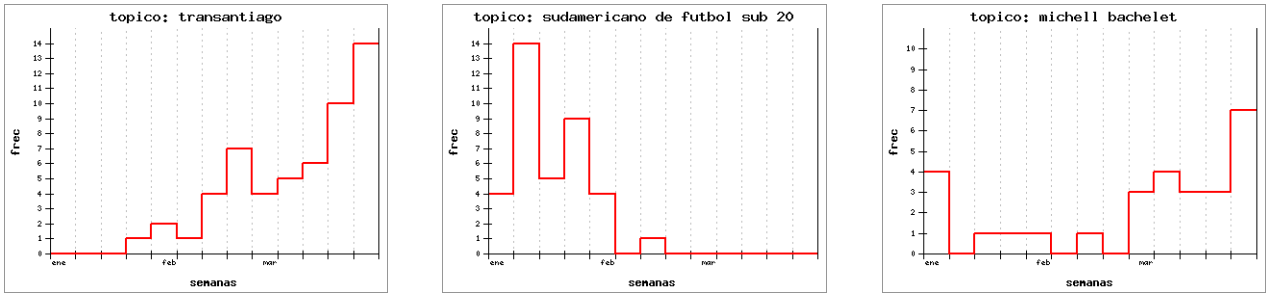


Figura 3.5: Gráficos de frecuencias para los tres primeros tópicos de la colección.

A modo de ejemplo, la siguiente es la pantalla con el listado de artículos relacionados al tema *transantiago*, en la interfaz creada para la administración de los artículos:

[3]Dado que el sitio web *orbitando.com* es un sitio en constante expansión, esta estimación representa en la práctica una cota inferior. Dada la tasa de crecimiento estimada del sitio, al cabo de un año esta cifra podría llegar a duplicarse.

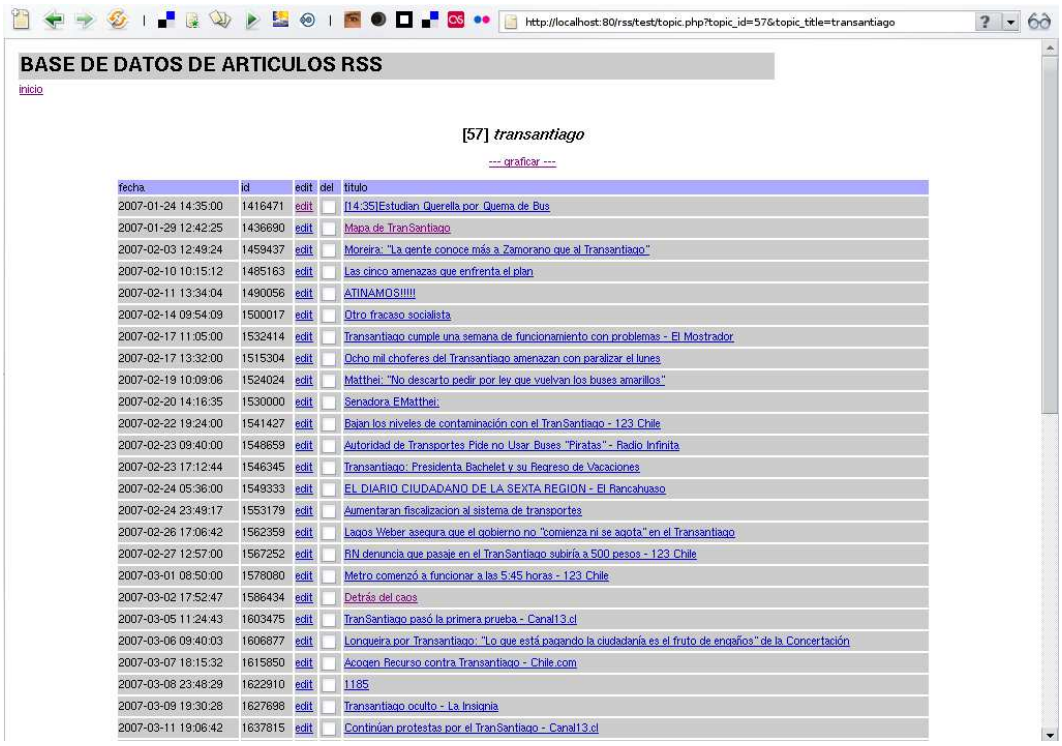


Figura 3.6: Listado de artículos pertenecientes al tópico *transantiago*.

Otro resultado interesante, es obtener el ranking de los top-10 tópicos para cada uno de los meses: enero, febrero y marzo.

enero 2007	
tópico	frec
sudamericano de futbol sub 20	36
fernando gonzalez	11
contrataciones club u.de chile	9
gabriela mistral	9
copa davis chile-rusia	8
anno nuevo 2007	7
augusto pinochet	7
michell bachelet	7
pildora del dia despues	7
alberto fujimori	6

febrero 2007	
tópico	frec
copa davis chile-rusia	15
transantiago	15
copa america de futbol	13
festival de vinna	13
conflicto limitrofe con peru	11
resultados club u.de chile	11
torneo de apertura de futbol	9
fernando gonzalez	7
club universidad catolica	6
colo colo	6

marzo 2007	
tópico	frec
transantiago	35
michell bachelet	17
partido futbol chile-brasil	15
andre agassi	9
chiledeportes	9
colo colo	9
educacion	9
resultados club u.de chile	9
sudamericano de futbol sub 17	9
gabriel garcia marquez	8

Cuadro 3.4: Listado de los top-10 tópicos para cada mes del estudio.

Con esto se comprueban en gran parte los resultados obtenidos en este proceso: cada mes presenta los temas más tratados en cada uno de los períodos. Este es un primer acercamiento a los resultados que se busca obtener en el desarrollo del proyecto.

Estadísticas de palabras

Otros resultados interesantes se obtienen si se realiza un conteo de las palabras encontradas en la muestra. Esto permite tener una idea del comportamiento de las palabras de la colección, y su distribución. El siguiente listado fue realizado al ejecutar un proceso de generación de lista de palabras, sin utilizar ninguna de las técnicas de filtrado mencionadas en el capítulo 2. Sólo se consideraron palabras con un largo mínimo de 3 caracteres. El resultado arrojó un total de 27799 palabras, para el total de 2000 documentos muestreados.

Se muestra a continuación, la lista de las palabras más frecuentes encontradas en el conjunto de referencia. La frecuencia mostrada indica el número de documentos, sobre el total de 2000, que contiene cada palabra.

palabra	frecuencia
hace	1176
que	1149
chile	1106
del	984
los	825
por	797
con	760
para	700
las	627
una	601
relacionados	567
articulos	567
como	425
mas	353
este	339
esta	326
radio	319
com	271
quot	255
hoy	253
sus	246
pero	240
fue	220
minutos	219
entre	217
universidad	204
sobre	202
nacion	191
años	183
ser	175
desde	173
sin	172
todo	170
futbol	169
nuevo	168
cuando	162
ante	160
dos	158
tiene	157
solo	156
mercurio	153
peru	151

palabra	frecuencia
nacional	149
donde	149
son	145
argentina	145
prensa	144
hasta	143
santiago	141
muy	141
canal	140
segunda	139
diario	138
tenis	137
http	135
mundo	133
uno	132
año	128
online	127
tambien	121
parte	120
hay	120
dia	120
america	120
style	119
gobierno	119
ahora	118
vida	117
vez	117
todos	117
tiempo	117
primera	117
gran	117
pais	116
href	114
deportes	114
durante	111
contra	111
src	109
cooperativa	109
bien	108
otros	107
horas	107
porque	106

palabra	frecuencia
minutosel	105
han	105
cine	105
nos	104
asi	102
quien	100
img	100
latina	99
brasil	99
border	99
mexico	96
alt	96
ver	95
tanto	95
primer	95
hacer	93
tras	92
otro	92
nueva	92
mejor	92
luego	91
literatura	91
esto	91
ademas	91
puede	90
personas	90
mar	90
dijo	90
blogger	90
estan	88
sera	87
menos	87
tan	86
minutosla	86
jpg	86
grupo	86
eso	86
cursor	86
margin	85
tres	84
pasado	84
mientras	84

Cuadro 3.5: Listado de las palabras más frecuentes encontradas en la colección.

Las palabras encontradas, muestran la existencia de muchas palabras que se sabe no aportan información respecto a la discriminación que se desea realizar. Esto revela la necesidad de utilizar, en primer lugar, la técnica de eliminación de *stopwords*, además del uso de los algoritmos de *stemming* para reducción a la raíz. Aquí también es importante mostrar la distribución de las palabras, que nuevamente responde a una ley de potencia. Este resultado permite justificar la posterior poda de la cola de la distribución, con lo que se consigue una notable reducción del conjunto de palabras consideradas en el conjunto, lo cual redunda en una buena reducción dimensional.

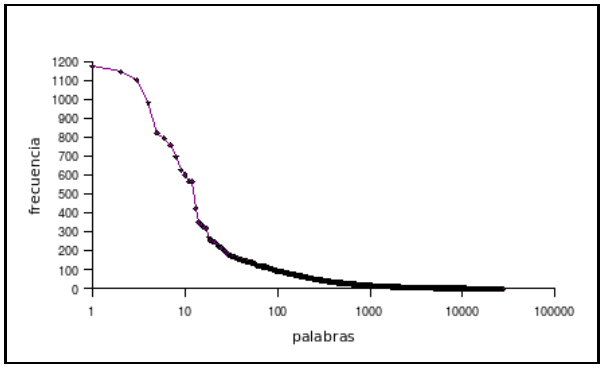


Figura 3.7: Distribución de las palabras de la muestra.

Todos estos resultados entregan una primera idea del comportamiento de la información contenida en la colección del estudio, la distribución de las palabras del vocabulario y principalmente, la distribución de los tópicos en el sistema. Con todos estos resultados, el proceso continúa con la creación e implementación del sistema, con el fin de obtener resultados que se aproximen de la mejor manera a los resultados obtenidos en esta fase.

Capítulo 4

Visión general del sistema

...o veamos que es lo que queremos hacer.

Este proyecto plantea el desarrollo de un prototipo para el sistema de clasificación automática de artículos RSS. Esta aplicación sería implementado en el sitio web *orbitando.com*, integrada al resto del sistema que mantiene el sitio web, comunicándose con las demás componentes a través del protocolo REST[1], implementando llamadas a través de *web services*. Este sistema entonces, basaría su funcionamiento en la invocación de llamadas http, tanto para la agregación de artículos, como para las solitudes de consultas por parte del servidor web.

Para ello, el presente proyecto plantea el desarrollo de un prototipo que permita implementar las funcionalidades del sistema, desarrollando los distintos casos de uso, y principalmente, el ajuste de los parámetros que definen el comportamiento de la aplicación. En este capítulo se presenta el escenario en el cual fue desarrollada la aplicación, y se presenta el diseño del sistema, desde el punto de vista de la aplicación.

[1] Mecanismo de comunicación entre componentes de software, utilizando XML y HTTP.

4.1. Contexto de la aplicación

El desarrollo de este proyecto se planteó como la creación y desarrollo de un prototipo de un sistema que sería implementado finalmente en operación, en el contexto del sistema de orbitando.com. Esto planteaba una serie de restricciones y requerimientos respecto del sistema en desarrollo

El sistema detrás de orbitando.com presenta al servidor web (un servidor de aplicaciones *Tomcat*) como el punto de inicio de un completo sistema de control de artículos (ítems), el cual permite acceder a los contenidos almacenados en la base de datos, procesar nuevos artículos, realizar búsquedas, llevar estadísticas, entre otras tareas. El caso de consulta del sitio web, es atendido por el servidor web, el cual presenta al usuario una portada construída por un conjunto de componentes pre-procesadas: *tag-clouds*, artículos más vistos, ranking blogs. Por otro lado, la llegada de ítems al sistema, mediante el proceso de *polling*, se traduce en la siguiente secuencia de acciones:

- se agrega ítem a tabla ITEMS
- se realiza una extracción de términos
- se calcula su categoría, en base a la fuente de origen
- se determina si es un artículo repetido o nuevo
- la meta-información es almacenada en la tabla ITEM_METADATA

En este escenario, el sistema de detección de tópicos correspondería a un proceso desarrollado a continuación de los pasos anteriores, ya que parte de la información almacenada en la tabla ITEM_METADATA es utilizada para discriminar los artículos que pueden ser procesados.

Cada una de las componentes del sistema vive en la forma de un servicio web (*web service*), y la comunicación entre las partes se realiza utilizando el protocolo REST. Esto implica que, en última instancia, la comunicación desde y hacia el sistema de clasificación de tópicos, debe ser hecha mediante XML.

Una vez conocido el detalle de la implementación futura de la aplicación, se pueden plantear los casos de uso identificados. Estos corresponden a dos casos de procesamiento: la invocación del clustering batch y la llegada de nuevos artículos, y dos casos de consulta: consulta de los primeros tópicos detectados en la colección, y la consulta de artículos relacionados a un cierto tópico. El primer caso, es una acción que sólo puede ser invocada por el administrador del sistema (un actor humano), y se ejecuta de manera extraodinaria, al principio de la operación, o posteriormente si se quisiese reiniciar el sistema . En cambio, el agregar artículos es una operación en constante ejecución, ya que cada nuevo artículo (válido) que sea incorporado a la base de datos debiera ser

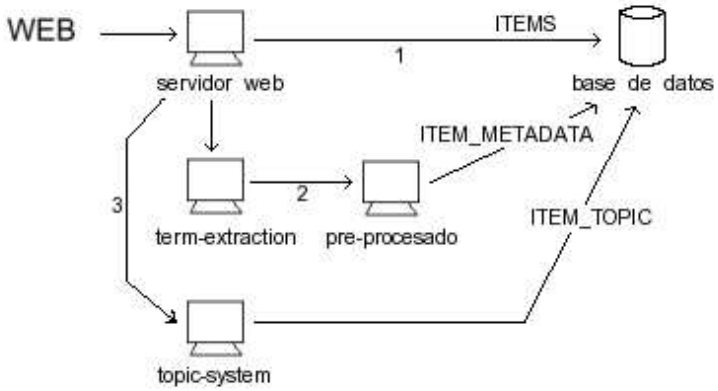


Figura 4.1: Arquitectura del funcionamiento de orbitando.com

procesado por el sistema. Por su parte, los casos de consulta son también requeridos por el actor que representa a orbitando.com. Esto, porque la generación de contenidos de la portada del sitio se realiza de forma batch, es decir, cada cierto tiempo (horas) se genera una nueva versión de la portada principal, por lo que no existe el caso en que el usuario final requiera directamente la información del sistema de generación de tópicos.

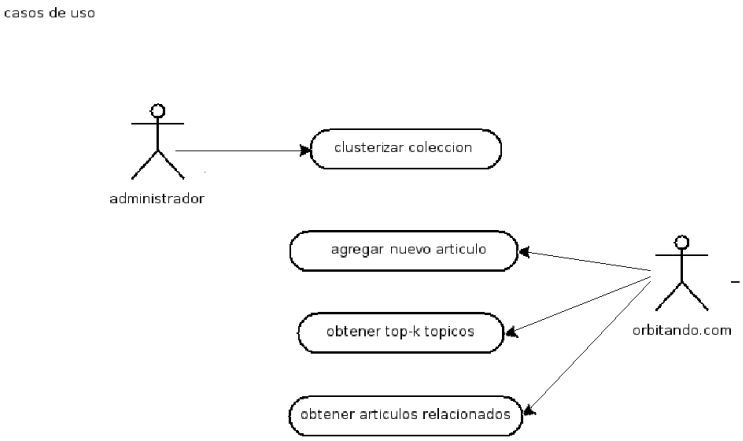


Figura 4.2: Diagrama de casos de uso del sistema.

4.2. Prototipo del sistema

Una vez definida la situación del sistema en operación, se pudo concebir el diseño de lo que sería la aplicación. Esta fue desarrollada utilizando el enfoque de desarrollo orientado a objetos y

el lenguaje Java, permitiendo desarrollar un sistema que fuera modificable, escalable, y sobretodo, modular. Esto era un requerimiento auto-impuesto debido a que se quería tener un sistema que permitiera ir probando diversas componentes de manera transparente.

El diseño final comienza por una componente principal, la que representa el corazón del sistema. Esta componente es el punto de partida de cada uno de los casos de uso, al mismo tiempo que realiza la correcta invocación de las funcionalidades de cada uno de los módulos que implementan las operaciones. También, esta componente es el lugar donde se mantiene la persistencia en memoria de la información necesaria en los procesos del sistema.

En el nivel superior, el módulo central fue encapsulado por otras componentes que cumplen el rol de interfaz del sistema, permitiendo su operación real por parte de los clientes: orbitando.com, administrador del sistema, desarrollador. El resultado es una interfaz que permite acceder al sistema en la forma de una aplicación java, desde consola, para realizar el conjunto de pruebas durante la etapa de desarrollo. Y una segunda interfaz, que representaría el intermediario entre el sistema, y sus clientes, en el contexto de un web service. Es decir, esta componente es la que traduce llamadas a los métodos de la componente central, en código XML.

En cuanto a los procesos de la aplicación, estos fueron divididos en tres módulos: el módulo de indexación de documentos, el módulo de clustering batch, y el módulo de clustering incremental. La característica principal de este enfoque del diseño, es que la componente principal puede invocar cada una de las funcionalidades de los módulos, sin importar la manera en que cada uno fue implementado. Esta abstracción permitió realizar un desarrollo modular, fundamental a la hora de decidir la implementación de cada componente.

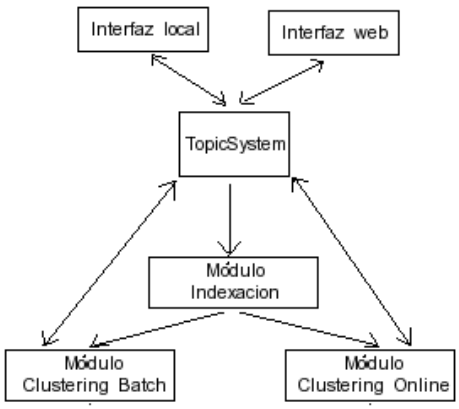


Figura 4.3: Arquitectura lógica del sistema.

El objetivo final del presente proyecto, es el de construir el prototipo de la aplicación, con datos reales, de manera de poder generar la información requerida por el sistema: los tópicos principales de la colección.

Capítulo 5

Módulo: indexación de documentos

...o como hacer que el computador entienda lo que nosotros entendemos.

5.1. Presentación y evaluación del problema

Una vez evaluados todos los antecedentes del proyecto, se presenta a continuación cada uno de los módulos que componen el sistema. En primer lugar, se presenta el desarrollo del módulo de conversión de documentos a vectores, basado en el modelo de espacio vectorial.

El objetivo de esta etapa es, a partir de una colección de documentos correspondientes a un conjunto de artículos de la base de datos del sistema, generar una colección de vectores relacionados bajo un espacio vectorial definido por el conjunto de palabras encontradas en dicha colección. A partir de este enunciado, se pueden mencionar los objetivos del desarrollo de esta componente:

- Se desea que los vectores (y en particular, los pesos asignados a sus componentes), sean un buen reflejo de la *semántica* de cada documento.

- Se necesita generar una buena colección de palabras, que sea representativa de la información de los artículos.
- La dimensionalidad del espacio (la lista de palabras utilizadas) debe ser la menor posible.
- Se desea obtener el conjunto de clusters que representen los tópicos más populares, según lo obtenido en el estudio presentado en el capítulo 3.

Teniendo definido el objetivo de esta etapa, se procedió a la evaluación de las alternativas disponibles para llevar a cabo esta tarea. En la primera etapa de desarrollo, las posibilidades se podían reducir a dos escenarios: desarrollo completo del módulo, o la utilización de una librería o programa existente.

El primer caso implicaba la implementación de todos los procedimientos estudiados, con el fin de generar la representación vectorial de la colección. Esta posibilidad podía ser viable, en la medida que se intentara producir una componente de software lo más adaptada posible al caso del proyecto, principalmente buscando optimizar las variables en juego: tiempo y espacio. Sin embargo, este tema no resulta ser una variable crítica en este caso, ya que el principal costo de operación es cargado por la componente de *clustering batch*. Por su parte, la segunda alternativa condujo a la evaluación de algunas herramientas existentes, siempre bajo el paradigma de software *open source*, y teniendo en cuenta que las nuevas componentes serían desarrolladas utilizando *Java*. Bajo este escenario, se evaluaron las siguientes alternativas:

Lucene

La herramienta, parte del proyecto *Jakarta*, es una librería ampliamente utilizada en el desarrollo de proyectos web, ya que permite implementar todo un sistema de búsqueda utilizando exclusivamente el lenguaje Java. Un uso típico es la implementación de un buscador para un sitio web corporativo.

Otro factor a tener en cuenta, era el hecho de que *orbitando.com* ya implementa un buscador basado en Lucene: Solr. Esto -aparentemente- presentaba la ventaja de disponer de un espacio vectorial ya construido, incluyendo el uso de índices invertidos[1]. Pero, esta aparente ventaja en realidad significaba atarse a un modelo de representación vectorial que quizás no era el más apropiado para el caso en estudio: las posibilidades de manejar la representación de documentos estaría limitada. Además, el uso de índices invertidos no era algo realmente necesario, ya que el sistema nunca estuvo enfocado a realizar búsquedas.

Weka

Weka[23] es una colección de algoritmos de aprendizaje de máquinas, desarrollado en Java,

[1]Por *índice invertido* se entiende un modelo de representación de documentos, en el cual el acceso a los vectores se realiza a través de las palabras que lo definen. Es decir, para cada término se almacena el conjunto de vectores (documentos) que lo contienen.

para su aplicación a problemas de minería de datos. Este software fue desarrollado en la universidad de Waikato, y es un sistema open-source, el cual está disponible en web para su descarga y desarrollo.

Esta librería puede ser utilizada en una aplicación gráfica, tipo *standalone*, o bien, puede ser usada como una librería incorporada a los programas desarrollados. Con respecto a su utilidad para el proyecto, si bien posee mecanismos de indexación de documentos, éstos son algo limitados. Además, el desarrollo de esta herramienta está enfocado principalmente a estudios de investigación, desfavoreciendo temas de eficiencia.

WVTool

La librería WVTool[34][24], es una librería desarrollada en Java, de creación reciente. Presenta todo un *framework* para el desarrollo de la indexación de documentos. Su principal ventaja: su diseño permite realizar modificaciones fácilmente, lo cual lo hace altamente configurable, utilizando la herencia en Java, y el uso de interfaces.

5.2. WVTool

WVTool, "the Word Vector Tool", es una librería Java desarrollada para el estudio estadístico de modelos de lenguaje, e integración con webservices o bases de datos. Permite realizar la tranformación de documentos de texto en su representación vectorial,

WVTool es una librería compuesta por un conjunto de clases, cuyo objetivo es realizar la transformación de documentos a vectores, según el modelo de espacio vectorial. WVTool fue desarrollado pensando siempre en su uso bajo escenarios muy distintos, permitiendo manipular sus propiedades mediante la adecuada elección de la clase Java que implementa cada una de las funcionalidades. Su implementación se basa en un esquema basado en interfaces Java, las cuales modelan las diversas funciones o acciones del sistema.

Las distintas clases que conforman la librería estan distribuidas en un conjunto de paquetes de clases que agrupan diversas funcionalidades. En un primer nivel, se tienen los siguientes directorios:

- config: el conjunto de clases utilizadas para configurar parámetros (*WVTConfiguration*, *WVTConfigurationRule*)
- crawler: *WVToolCrawler*, ...
- generic: este *package* contiene al conjunto de clases de operación del proceso de filtrado de documentos.
- main: contiene las clases principales de la librería: *WVTool*, *WVTWordVector*, *WVTInputList*, etc.
- util: algunas clases utilitarias: *WVToolException*, *TokenEnumeration*, etc.
- wordlist: contiene las dos principales clases del modelo: *WVTWordList* y *WVTWord*.

Para invocar la librería, se debe instanciar a la clase WVTool, la cual permite la invocación de los métodos principales: `createWordList(..)` y `createVectors(..)`. La clases WVTool no almacena los datos de operación, sino que inicializa o carga los objetos necesarios, pero es responsabilidad de la aplicación cliente la persistencia en memoria de los datos. A modo de ejemplo, la invocación desde un programa en Java sería:

```
WVTool wvt = new WVTool(false);

WVTConfiguration config = new WVTConfiguration();
WVTFileInputList list   = new WVTFileInputList(2);
                 list.addEntry(new WVTDocumentInfo("datasample", "txt", "", "spanish"));

WVTWordList wordList = wvt.createWordList(list, config);
WordVectorWriter wvw = new WordVectorWriter(outFile, true);

wvt.createVectors(list, config, wordList);
```

`createWordList( list, config )` permite construir la lista de palabras que será utilizada en la transformación a vectores. `createVectors( list, config, wordList )` genera la representación vectorial de los documentos dados en list, al mismo tiempo que los va guardando a disco.

Un esquema de la estructura de la librería WVTool, y sus componentes (clases y *packages*), es el siguiente:

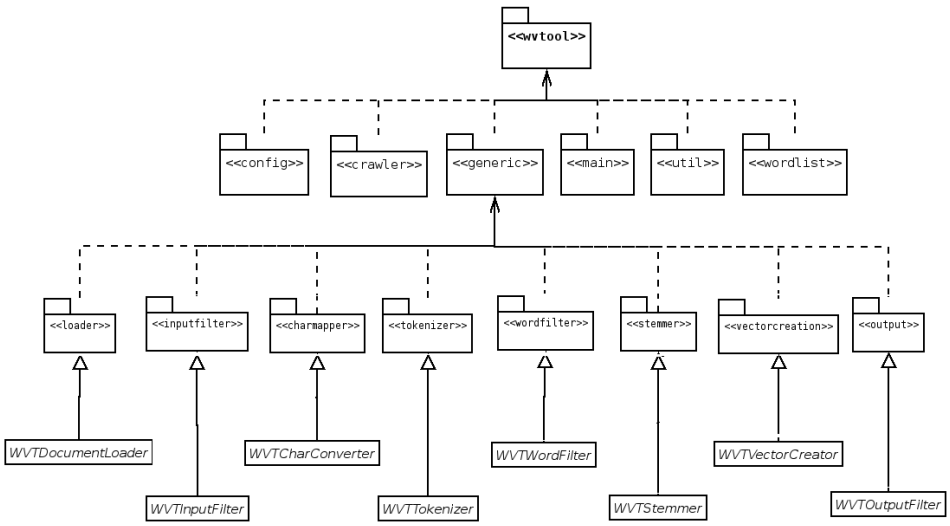


Figura 5.1: Resumen de la estructura de clases de la librería WVTool.

El funcionamiento de WVTool, se basa en el uso de un conjunto de filtros, los cuales forman una cadena para el procesamiento de los documentos de entrada. Por ejemplo, comenzando con

un archivo de texto, el sistema se encarga de ejecutar las diversas etapas para llegar finalmente al resultado del vector con los pesos asignados. Los filtros utilizados por WVTool son los siguientes:

- *Loader*: Realiza la carga de la fuente de datos. Para cada documento a leer, este objeto almacenaría la ruta de un archivo, o la URL de una web.
- *InputFilter*: Representa el primer filtro aplicado sobre la entrada. Permite obtener texto puro a parit de entradas de diferente naturaleza (PDF, XML, HTML).
- *CharMapper*: Realiza conversiones sobre los caracteres de la entrada, por ejemplo, la eliminación de acentos.
- *Tokenizer*: Separa el stream de entrada, en un conjunto de palabras, o *tokens*.
- *WordFilter*: Ya con las palabras separadas, realiza un primer filtrado directo, típicamente utilizando una lista de stopwords.
- *Stemmer*: Realiza el proceso de reducción a la raíz, *lematización* o *stemming*.
- *VectorCreation*: Dado el conjunto de palabras definitivo, obtiene la representación vectorial del documento.
- *Output*: Establece la salida donde se almacenará el vector recién creado.

Estos filtros son ejecutados en secuencia, comenzando por el *Loader*, hasta el *Stemmer*, que retorna un conjunto de palabras válidas. Con este conjunto, y la lista de palabras en uso, se invoca al método de creación de *VectorCreation*, y se construye el vector representante del documento. Finalmente, este vector se le entrega al objeto *Output*, para ser escrito a disco.

5.3. Solución

La solución implementada, consistió en la creación de una interfaz que modela la componente de transformación de documentos a vectores. Un objeto *String2Vector* representa la instancia que ejecuta esta componente, y presenta la intefaz de los métodos principales:

load()
Realiza la carga de los documentos a analizar a partir de la fuente (en este caso, la base de datos), y crea la lista de palabras.

createVSpace()

Ejecuta la creación de los vectores, calculando los pesos en función de las frecuencias de las palabras.

addVector()

A partir de la lista de palabras existente, calcula la representación vectorial de un nuevo artículo.

updateWordList()

Actualiza la lista de palabras, para el subconjunto de documentos referenciados en el *buffer* de los últimos artículos agregados al sistema. El uso de este método es opcional, y depende de la implementación de clustering en línea que sea utilizada.

loadWordList()

Este método sólo es utilizado para poder cargar la lista de palabras nuevamente a memoria, en caso de caída del sistema.

El diseño fue planificado de forma de poder abstraer la implementación de los distintos casos de uso, de su invocación en cada uno de los módulos. En este caso, el uso de la clase *WVTString2Vector* representa la implementación de los métodos mencionados, para el caso de la librería WVTool.

En resumen, la entrada de este módulo consiste en un texto que especifica la fuente de los datos a extraer: una muestra aleatoria de tamaño n, o un período de tiempo, los cuales son leídos desde la base de datos del sistema. Por otro lado, la salida de esta componente es un archivo de texto con la información de los vectores creados. Estos vectores se encuentran en su forma expandida, donde la mayoría de sus componentes son valores nulos. Este archivo (o matriz) es lo que será utilizado como entrada al siguiente módulo, el módulo de clusteirng batch.

La implementación del módulo, utilizando WordVectorTool, se basa en la implementación de la clase *WVTString2Vector*, junto a la definición (o redefinición) de cada uno de los filtros anteriormente presentados.

A continuación se presenta cada uno de los problemas enfrentados, y la solución planteada para cada caso.

5.3.1. Loader: carga desde la BD

La carga de documentos se debía realizar desde la base de datos del sistema, sin embargo, la librería no proveía de una clase Loader que permitiera cargar datos desde una consulta a una base de datos. Esto motivó la creación de las clase *DBLoader* y *DBInputList*, las cuales permiten leer los datos -los documentos- desde la base de datos, a partir de una consulta inicial que es expandida por la *inputList*.

Por ejemplo, si DBInputList es cargada con la query `SELECT title, description FROM ITEMS WHERE date >= "2007-01-01" AND date <"2007-02-01"`, esta consulta es expandida en un conjunto de consultas, donde cada una apunta a un documento específico en la base de datos. Cada una de dichas consultas son almacenadas en un objeto *WVTDocumentInfo*, generando una lista de objetos WVTDocumentInfo. Finalmente, cada documento es leído por la clase BDLoader, generando un *stream* de datos que es entregado al siguiente eslabón en la cadena de filtros.

5.3.2. InputFilter: transformación a texto plano, y útil.

Un primer problema con respecto a los datos leídos, es el hecho de que el contenido viene, indistintamente, en formato de texto plano, o en formato html. Esto hace necesario normalizar la entrada, eliminando todos los tags html, es decir, todo contenido entre $<$ y $>$. Para esta labor, WVTool posee la clase *SimpleTagIgnoringReader*, que realiza esta tarea.

Una vez solucionado el problema de los documentos html, aparecieron un conjunto de problemas derivados de la naturaleza de los documentos en estudio. Lamentablemente, a pesar de tratarde de documentos en formato RSS (lo cual haría creer que responden a una estructura rígida de separación de los contenidos), existen muchos problemas de forma que ensucian de una u otra manera el contenido que se obtiene. Entre otros fenómenos, se observa:
• Los artículos provenientes de *google news*, la principal fuente de artículos, generalmente incluyen referencias (links) a otros artículos, en la forma de “artículos relacionados”. Y este texto representa cerca de un tercio del total del contenido del artículo.
• Cerca de la mitad de los artículos de la colección contienen en el título el nombre de la fuente. Esto provoca una distorsión de los contenidos, ya que al momento de agruparlos, se producen asociaciones indeseables según la fuente de origen.

La solución a ambos problemas fue la creación e implementación de una clase que cumpliría el rol de filtro a este nivel:
emphRSSInputFilter, la cual permite filtrar todos los contenidos de los enlaces (textos entre las etiquetas $<a$ ...$>$ y $$). Además, se creó un mecanismo de filtrado de nombres de fuentes, el cual lee un conjunto de términos correspondientes a nombres de medios (periódicos, medios

digitales, agencias de noticias) y realiza un filtrado de dichos nombres en el texto en proceso[2]. Con esto se consigue eliminar los dos problemas mencionados.

5.3.3. CharMapper: eliminación de acentos

Un segundo problema con el *stream* de entrada, es el tema de los acentos, junto a la aparición de algunos caracteres provenientes del uso de *html*. En el primer caso, se decidió normalizar todos los acentos (tildes), llevándolos a su forma no acentuada. Esto se suele hacer en los buscadores web, debido a que aún existen muchas fuentes que intencionalmente omiten los acentos. El segundo caso, se refiere a la aparición de caracteres, tales como: ` `, `&aacute;`, `&ntilde;`, `&quote;` (caracteres especiales del html), los cuales fueron convertidos a su forma final. Ambas tareas fueron solucionadas creando una clase *RSSCharMapper*, que implementa las operaciones señaladas.

5.3.4. WordFilter: filtrado de stopwords

El filtrado de palabras, o *stopwords*, es la primera operación que se realiza sobre el conjunto de palabras obtenido de cada documento. Consiste en eliminar palabras, a partir de una lista dada.

El propósito inicial de este paso, es el de eliminar aquellas palabras de uso frecuente en el lenguaje, tales como artículos (*él*, *la*, ...), pronombres (*yo*, *tú*, ...), adverbios (*ahora*, *después*, *todavía*, ...). Sin embargo, esta idea se puede extender con el fin de cubrir otros conjuntos de palabras, según los requerimientos de la aplicación.

En esta implementación se utilizó la clase *StopWordFilterFile*, que utiliza un archivo de texto como base para eliminar palabras no deseadas. La lista se lee de un archivo `stopwords.txt`, el cual contiene las palabras a filtrar. Este archivo fue construído especialmente para este proyecto, buscando eliminar todas aquellas palabras que no contribuyesen a discriminar los documentos. Su estructura es la siguiente:

- artículos
- pronombres
- adjetivos (determinativos)

[2]Esta tarea es necesario realizarla de manera independiente al proceso de filtrado de stopwords, ya que dicho proceso sólo es capaz de eliminar palabras (*tokens*), y no términos compuestos, como es el caso de la mayoría de los nombres de medios. Si se filtrase los nombres de medios utilizando palabras de sus nombres, por ejemplo, *tercera* o *cooperativa*, se perdería información, ya que se eliminarían palabras útiles en ciertos contextos, y necesarias al momento de discriminar contenidos.

• adverbios
• verbos de uso básico: estar, haber, ser, tener

• stopwords en inglés Se agrega una lista básica de stopwords de habla inglesa, con el propósito de poder enfrentar la llegada de palabras de documentos en inglés. En caso contrario, se corre el riesgo de que se produzca *ruido* por palabras que siempre aparecerían, pero con relativa baja frecuencia.
about, *after*, *again*, *all*, *almost*, ...

• palabras provenientes del html Empíricamente se observó que a pesar del filtrado de stopwords del español, y a pesar del filtrado de tags html, existen muchos casos donde no se filtran palabras relacionadas a la computación o los documentos html. Esto hizo necesario considerar el filtrado explícito de un conjunto de palabras relacionadas con el html, principalmente.
html, *span*, *alt*, *div*, *quote*, *border*, ...

• palabras relativas al contexto Finalmente, se tiene un conjunto de palabras que aún siguen apareciendo. Estas son las palabras que tienen que ver directamente con el contexto de la aplicación: los artículos sindicados, y la publicación de noticias. *articulos*, *relacionados*, *online*, *news*, *minutos*, ...

Un resultado interesante que se observó, es el hecho de que una buena lista de palabras, puede llegar a reemplazar el efecto del término IDF en la función de pesos, ya que ambos procesos buscan el sancionar la frecuencia masiva de palabras en la colección.

5.3.5. Stemming: lematización en español

Una de las características que motivó la elección de WVTool, fue el hecho de disponer de una implementación de un algoritmo de stemming para el idioma español. WVTool implementa un conjunto de algoritmos de stemming, y además, permite el uso de programas externos para realizar esta labor. Es el caso de los algoritmos de stemming del proyecto *Snowball*[32], entre los cuales se cuenta precisamente con una versión para español.

5.3.6. VectorCreation: la función de pesos

Un último tema a considerar, es el relativo a la función de peso utilizada en la transformación a vectores. Según lo expuesto en el capítulo 2, el método más utilizado es la función *TF-IDF*, el cual busca premiar aquellas palabras muy frecuentes en un documento, y castigar aquellas palabras muy frecuentes en toda la colección.

TF-IDF es una metodología desarrollada para su uso principalmente en bases de datos documentales, donde cada documento posee un tamaño mediano a grande. A su vez, los documentos pueden ser libros o publicaciones de caracter científico/técnico, o incluso, ser de índole narrativa. Por su parte, los artículos RSS presentan algunas particularidades:

- Poseen un tamaño relativamente pequeño.
- En su mayoría, son artículos de índole noticioso.

En base a estos dos puntos, es posible identificar un fenómeno que tiene origen en el periodismo clásico: el uso de la *pirámide invertida*[3] permite describir de una buena manera la forma en que las informaciones son comunicadas a través de cada artículo. Esto, debido a que el tamaño es siempre acotado (el objetivo de un artículo RSS es ser un resumen respecto de alguna información o publicación), y es necesario comunicar de la manera más eficiente la esencia de la información.

Dado lo anterior, y en base a la función TF-IDF, se propuso reemplazar la componente del término TF, basado en frecuencias, por una componente llamada PF, cuyo valor tuviera relación con la posición del término dentro del documento, ponderando la frecuencia encontrada. La idea es que este peso sea proporcional a la posición, premiando las apariciones en las primeras posiciones, y quitando importancia a las apariciones en los últimos lugares. Lo que se hizo fue dividir el documento en tres zonas: título, cabecera del cuerpo, y resto del cuerpo.

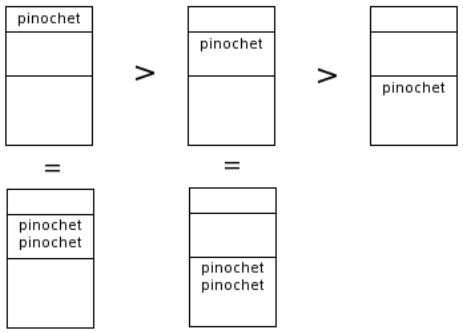


Figura 5.2: Esquema que muestra la primera idea sobre la función de pesos basada en posición. Lo que se busca es definir una relación de orden para las palabras del documento, combinando posición con frecuencia.

[3]La pirámide invertida es una estructura que norma la manera de escribir una noticia, cuando los medios de transmisión imponen un formato especialmente acotado. En el titular se debe comunicar lo esencial de la información, seguido por el *lead* o entradilla, donde se comunica todo lo básico para entender el contexto de la información. Por último, el cuerpo agrega todos los detalles necesarios a la información publicada. Este modelo surgió en los tiempos del telégrafo, con el fin de transmitir de manera resumida y eficiente las informaciones.

El resultado de este planteamiento, es la función PF-IDF, que se define como:

$$f(D_j, t_i) = PF * IDF = (\frac{f_{ij}}{|D_j|} * pos) * (-log(\frac{f_j}{|N|}))$$

donde pos corresponde a una puntuación que depende de la aparición de la palabra en cada uno de los bloques del documento, f_{ij} a la frecuencia del término en el documento, f_j es el número de documentos con el término f_j, y N es el conjunto de todos los documentos considerados. De esta manera, se tiene un puntaje por aparecer en la primera zona (en el título), un puntaje por aparecer en el segundo segmento, y otro, menor, por aparecer en la última parte. Los valores asignados a cada segmento fueron determinados mediante experimentos realizados en la aplicación.

Capítulo 6

Módulo: clustering batch

...o como hacer que el computador vea lo que nosotros no vemos.

6.1. Problema

La segunda componente a ser abordada, corresponde al proceso de *clustering batch*, es decir, al proceso de segmentación del espacio vectorial creado por la componente de indexación de documentos, a partir de un conjunto de artículos de la base de datos.

Los antecedentes técnicos de este problema fueron abordados en el capítulo 2, donde se presentaron los fundamentos de las técnicas de clustering y se presentaron algunos de los algoritmos más utilizados. Para el caso del presente proyecto, el problema consiste en clasificar una colección del orden de los 200.000 artículos, correspondientes a la ventana de tiempo de 3 meses con la que se trabajó. Para ello, se necesita utilizar alguna herramienta que permita realizar de manera eficiente el clustering de la colección, con el propósito de descubrir los principales tópicos presentes en el conjunto.

Los objetivos de esta parte del proyecto, son los que surgen al plantear la tarea principal: segmentar una parte de la base de datos de artículos. En este sentido, se requiere:

- Obtener clusters (tópicos) que cumplan con el enunciado del proyecto, es decir, que sean representativos de los tópicos reales encontrados en la colección.
- Ejecutar una clusterización que funcione en tiempos razonables.
- Que los resultados entregados por el módulo, sean rescatables por el sistema, de forma de poder compartir los resultados con el resto del sistema.

Entre las alternativas disponibles, nuevamente estaba la posibilidad de implementar todos los procesos del módulo, sin embargo, en este caso el tema de la eficiencia era crítico, por lo que era recomendable externalizar dicha tarea. En este sentido, la decisión condujo a elegir el software Cluto[29][16], una herramienta ampliamente utilizada en minería de datos, la cual permite realizar eficientemente el clustering de grandes colecciones de documentos.

6.2. Cluto

Cluto es un programa desarrollado en C++, que permite su operación a través de un programa gráfico para visualización de los resultados, y también, su ejecución a través de un programa invocado por línea de comandos (*vcluster*). Incluso puede ser utilizado como librería incrustada en programas. Además, permite realizar análisis completos sobre la clusterización creada, entregando estadísticas sobre cada uno de los clusters que forma.

Le elección de Cluto impuso a su vez, la necesidad de comunicación de los datos y resultados entre los módulos del sistema. En este caso, la entrada a Cluto consiste en un archivo de texto, conteniendo la matriz de datos a analizar. Por su parte, Cluto también entrega sus resultados en archivos de texto plano, en este caso, con la asociación de vectores con sus clusters.

Concretamente, el software de segmentación requiere que le sea entregado un archivo de texto, conteniendo el conjunto de vectores a procesar. Cada vector puede estar en uno de dos formatos posibles: *sparse* y *non-sparse*. Cuando un vector contiene una gran cantidad de ceros, y sólo unos pocos valores no nulos, se dice que está en la forma *sparse* si en lugar de almacenar el valor de cada componente, sólo se almacenan los valores de las componentes no-nulas, junto con el índice a la posición dentro de las coordenadas.

Por otro lado, Cluto permite la configuración de un gran número de parámetros que guían el

procesamiento de los datos. Entre estos, se pueden mencionar:

- *Método de particionado*:
  La decisión entre **rb**, repeated bisections, o **direct**, direct k-way. Además, están disponibles el método aglomerativo y el método basado en grafos.
- *Función de similaridad*:
  Los valores posibles son: **cos**, **corr**, **dist**, **jacc**.
- *Función objetivo*:
  Corresponde a la función que es optimizada por el algoritmo. Existe un conjunto de funciones posibles de utilizar[26], entre las que se cuentan:
  i1, **i2**, **e1**, **g1**, **g1p**, **h1**, **h2**,
- *Criterio de bisección*:
  Determina el criterio usado en la bisección, al utilizar *rb*.
- *rowmodel*, *colmodel*:
  Realiza la normalización de las filas o las columnas, según algún criterio estadístico.
- *Número de intentos*:
  El número de intentos que se realizan, buscando aproximarse al óptimo global.
- *Número de iteraciones*: Número de iteraciones de refinamieneto del algoritmo.
- otros parámetros:
  Además, existen otros parámetros que no fueron considerados en los experimentos, tales como: *nnbrs*, *grmodel*, *edgeprune*, *agglofrom*, *agglocrfun*, entre otros, los cuales son usados en otros escenarios (por ejemplo, en el método de grafos).

El siguiente es un ejemplo de la salida de Cluto, al ser invocado mediante el programa vcluster, en línea de comandos:

6.3. Solución

La solución implementada corresponde al desarrollo del módulo de clustering batch. Este módulo fue desarrollado de manera análoga al módulo de indexación de documentos, es decir, se creó una interfaz Java, a través de la cual el sistema interactúa con él.

Se creó la interfaz *Vector2Cluster*, la cual modela los procesos genéricos de este módulo. Los métodos que posee son:

```
*******************************************************************************
vcluster (CLUTO 2.1.1) Copyright 2001-03, Regents of the University of Minnesota

Matrix Information ------------------------------------------------------------
  Name: /tmp/ts0926.mat, #Rows: 100, #Columns: 284, #NonZeros: 28400

Options -----------------------------------------------------------------------
  CLMethod=RB, CRfun=I1, SimFun=Cosine, #Clusters: 10
  RowModel=None, ColModel=None, GrModel=SY-DIR, NNbrs=40
  Colprune=1.00, EdgePrune=-1.00, VtxPrune=-1.00, MinComponent=5
  CSType=Best, AggloFrom=0, AggloCRFun=I1, NTrials=10, NIter=10

Solution ----------------------------------------------------------------------

------------------------------------------------------------------------
10-way clustering: [I1=1.89e+01] [100 of 100]
------------------------------------------------------------------------
cid  Size  ISim  ISdev   ESim  ESdev  |
------------------------------------------------------------------------
  0     3 +0.548 +0.024 +0.009 +0.003 |
  1     4 +0.469 +0.052 +0.026 +0.005 |
  2     5 +0.392 +0.045 +0.024 +0.006 |
  3     8 +0.245 +0.034 +0.025 +0.007 |
  4     9 +0.219 +0.032 +0.021 +0.006 |
  5     9 +0.203 +0.029 +0.017 +0.010 |
  6    10 +0.204 +0.033 +0.025 +0.007 |
  7    10 +0.179 +0.028 +0.027 +0.011 |
  8    21 +0.114 +0.031 +0.025 +0.017 |
  9    21 +0.071 +0.033 +0.018 +0.015 |
------------------------------------------------------------------------
-------------------------------------------------------------------------------
10-way clustering solution - Descriptive & Discriminating Features...
-------------------------------------------------------------------------------
```

Figura 6.1: Salida del programa Cluto, para un ejemplo de 100 documentos y 10 clusters.

load()
Inicializa los parámetros del módulo, para preparar la carga de datos.

run()
Ejecutael proceso de clustering y almacena los resultados según la forma utilizada por el programa o algoritmo usado.

saveClusters()
Almacena en memoria y disco, el resultado del proceso, obteniendo los resultados a partir de las salidas entregadas por el programa de clustering.

La implementación del proceso de clustering se realizó creando la clase *ClutoVector2Cluster*, la cual es la implementación de la interfaz *Vector2Cluster*, para el caso de Cluto. Esta clase implementa cada uno de los métodos presentados, permitiendo la operación del proceso. En primer lugar, load() sólo establece la ruta en disco de donde es leída la matriz de vectores de entrada. El método run() invoca al programa `vcluster`, estableciendo el conjunto de parámetros que fijan los datos mencionados anteriormente. Finalmente, el método saveClusters() se encarga de recoger el resultado entregado por Cluto, en un archivo de texto, y realiza la carga en memoria, y posteriormente, en la

base de datos.

A continuación, se presentan cada uno de los temas que tuvieron que ser revisados con detalle:

6.3.1. Función objetivo

La implementación de los algoritmos de clustering, en Cluto, se basan en la idea de optimizar una función objetivo, con lo cual se combinan los vectores y los clusters, de manera de maximizar o minimizar el valor deseado. Esa medida a optimizar es alguna medida de calidad de la segmentación realizada, por ejemplo, la medida de similaridad dentro de cada cluster, o la medida de distancia entre cada cluster, buscando maximizar la diferencia entre unos y otros grupos.

Las medidas de similaridad pueden ser agrupadas en tres familias: las que miden similaridad interna, las que miden calidad externa (distancias inter-clusters), y las que se basan en la representación de grafos. Incluso existen funciones que combinan los enfoques de similaridad interna y externa. Para el presente proyecto, se rucurrió a la medida de similaridad interna, ya que esta es una primera condición de los grupos: que los elementos dentro de él sean semejantes unos a otros. La opción de las funciones híbridas fue descartada de un principio, ya que su costo de procesamiento es elevado.

Entre las funciones de similiaridad interna, se dispone de dos funciones de similaridad que pueden ser utilizadas. Estas funciones se definen como:

$$I_1 = \sum_{i=1}^{k} \frac{1}{n_i} \Big(\sum_{u,vinS_i} sim(u,v) \Big)$$

$$I_2 = \sum_{i=1}^{k} \sqrt{\sum_{u,vinS_i} sim(u,v)}$$

Mediante algunas simplificaciones algebraricas[26], estas funciones pueden ser re-escritas como:

$$I_1 = \sum_{i=1}^{k} \frac{||D_i||^2}{n_i}$$

$$I_2 = \sum_{i=1}^{k} ||D_i||$$

Este resultado permite apreciar una de sus primeras diferencias: I_1 considera en su resultado la

cantidad de elementos en cada cluster, además de que el término $||D_i||$ hace un aporte cuadrático a la ecuación, donde $||D_i||$ corresponde a la raíz cuadrada de las similaridades entre todos los pares de documentos del cluster. Esto se traduce en el siguiente comportamiento: el uso de I_2 provoca clusters con una calidad (similaridad interna) más homogénea, versus I_1, donde se suele observar la formación de un cluster sumidero, a cambio de mejorar la calidad del resto de los grupos. Este cluster sumidero recogerá aquellos elementos que habitan la periferia de cada uno de los demás clusters, provocando un efecto "limpieza".

Por lo tanto, el comportamiento mostrado por I_1 se asemeja de mejor manera al comportamiento buscado, pero al mismo tiempo, implica considerar el hecho de la formación de un cluster de gran tamaño, pero de muy baja calidad. Esto se soluciona trivialmente considerando una poda de los clusters formados, eliminando aquellos clusters de menor calidad. Esto es directo en Cluto, ya que el resultado arrojado por el programa está ordenado por el grado de similaridad interna de cada grupo, por lo que basta con eliminar la cola del conjunto de clusters.

En [26] existe una completa explicación de la naturaleza de cada una de estas funciones, además, se presentan diversos experimentos que buscan comparar sus resultados.

6.3.2. Número de clusters

El número de clusters, es el último parámetro que falta determinar. En el caso general, este valor se desconoce y se debiera determinar empíricamente mediante un conjunto de experimentos, probando diferentes valores de k, y buscando el k óptimo. Esto quiere decir, se desea utilizar el mínimo número de clusters posible, a partir del cual no se produce una mejora considerable en la calidad, al aumentar dicho valor.

Por otro lado, la realización de la experiencia expuesta en el capítulo 3 permitió obtener una cantidad de tópicos que podría ser tomada como la cantidad "natural" de tópicos que se forman en la colección, para un tamaño dado de la muestra (2000). Esto da una primera aproximación al valor deseado.

Este resultado, junto a la definición de la función objetivo (I_1), permitieron definir el comportamiento del valor de k para los experimentos. Al asumir el hecho de que un conjunto de documentos queda contenido en el llamado "grupo sumidero", se puede definir el valor de k que permita enviar aquellas malas agrupaciones a dicho conjunto. Este conjunto contendría finalmente a la mayoría de aquellos tópicos que aparecieron con cardinalidad uno en la muestra de referencia.

Capítulo 7

Módulo: clustering on-line

...o como conseguir que el sistema sobreviva al paso del tiempo.

7.1. Problema

Lo expuesto hasta ahora, corresponde a la implementación de un sistema clásico de minería de datos, donde el objetivo que se persigue es la segmentación de una colección estática de documentos de una base de datos. Sin embargo, el presente sistema presenta la particularidad de que su operación principal consiste en la segmentación de un flujo constante de documentos, los artículos sindicados, los cuales deben ser agregados al sistema, de forma de mantener y preservar la información de la segmentación producida en la etapa anterior.

Esta componente del sistema, fue el último actor en aparecer en escena. En un principio, este problema era considerado casi un detalle dentro del desarrollo del proyecto. Sin embargo, fue surgiendo como una componente de gran importancia. Además, al estudiar el tema del clustering incremental (parte de lo expuesto en el capítulo 2) se pudo comprobar el pobre desarrollo de esta área, en comparación con las técnicas clásicas de clustering “estático”. Debido a estas razones, el desarrollo de esta componente no fue todo lo óptima que pudo haber sido, ya que su desarrollo no era la componente principal del proyecto, al mismo tiempo que los recursos disponibles (tiempo) eran limitados.

7.2. Solución

La solución implementada fue desarrollada de la misma manera que en los otros dos casos antes expuestos: se desarrolló una interfaz Java que define la funcionalidad mínima de la componente, con un conjunto de métodos que definen su comportamiento, mientras que la implementación fue realizada a través de una clase específica.

La interfaz de clustering on-line (*ClusteringOnline*) presenta los siguientes métodos:

addVector(vector, con):
Este es el método que realiza la agregación de un nuevo vector al sistema. El resultado de esta operación es el identificador del cluster al cual fue asociado el elemento (esto es usado principalmente con fines de *debug*).

save(con):
Este método actualiza la resultados de la clusterización en la base de datos.

Lo presentado corresponde a la descripción del módulo de clustering incremental, sin embargo, el proceso de clustering en línea pasa por la ejecución de al menos dos etapas: la transformación del nuevo documento a su representación vectorial, y en segundo lugar, la ejecución del clustering incremental, con el vector recién creado.

Creación del vector:
La transformación del documento en su representación vectorial, es realizada por el módulo *String2Vector*. Para ello, se invoca el método *addVector(...)* que recibe el artículo que se desea transformar. Por su parte, la lista de palabras (el objeto *wordList*) reside en el espacio de memoria del módulo de transformación de vectores. El procedimiento es similar el realizado en la primera etapa, es decir, se invoca a un método *createVector* en WVTool, el cual produce un objeto *WVTWordVector*. Esta transformación es realizada invocando al conjunto de filtros de WVTool.

Clustering incremental:
La solución implementada corresponde a una versión del algoritmo de *Single-pass*. Este algoritmo plantea el procedimiento de agregar cada nuevo vector al cluster más cercano, siempre y cuando su distancia sea menor a un cierto umbral. En caso contrario, se crea un nuevo cluster.

El método *addVector(...)*, de la clase *SimpleBufferClusteringOnline*, recibe el vector creado en el paso anterior, y busca en el conjunto de clusters al cluster cuyo centroide es el más cercano al nuevo

vector. A continuación se toma esa distancia, y se compara con un valor de umbral[1] que indica si el nuevo vector está lo suficientemente cerca del cluster encontrado, o no, en cuyo caso se crea un nuevo cluster con un único elemento. El nuevo cluster es agregado al sistema y se continúa con el proceso.

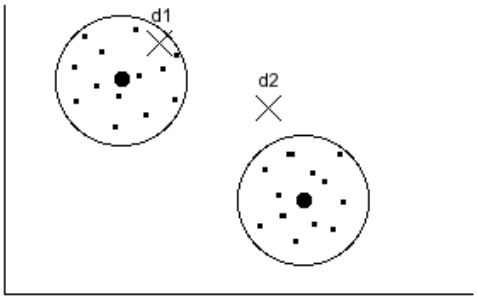


Figura 7.1: Esquema del proceso de asignación de documentos a clusters. En este ejemplo, d_1 es asignado al cluster de la izquierda del gráfico, mientras que d_2 no alcanza la distancia mínima necesaria respecto de su cluster más cercano, por lo que provocará el nacimiento de un nuevo grupo, de tamaño 1.

El desarrollo de este módulo planteó un conjunto de problemas. Estos problemas tienen relación con la naturaleza del proyecto, y significaron tener en consideración un conjunto de factores.

Problema de la actualización de palabras... la *wordlist* variable

Un primer problema que surge en este proceso, es el hecho de que los nuevos documentos -eventualmente- pueden contener palabras que no existen en la lista de palabras del sistema. Un primer enfoque es simplemente no considerar a esas nuevas palabras, y siempre calcular los vectores basado en la lista de palabras existente. Suponiendo que el clustering batch fue realizado sobre un conjunto lo suficientemente grande de artículos, se podría argumentar que la lista de palabras es una buena muestra de las palabras en la colección, y no sería necesario agregar nuevas palabras. Pero, este argumento no es válido en el caso de este proyecto, ya que uno de los propósitos principales, es la capacidad de detectar la aparición de nuevos tópicos, sean eventos noticiosos, o temas de discusión. El considerar una lista estática de palabras impediría la aparición de nuevos tópicos basados en palabras que no existían previamente.

La siguiente alternativa, consistiría en agregar las palabras encontradas en cada vector, en el momento de calcular su representación vectorial. La implementación permite realizar esto directamente. Sin embargo, en este caso aparece un nuevo problema: la agregación indiscriminada de palabras. En el proceso de clustering batch, una vez generada la lista de palabras, se realiza una poda sobre el conjunto, eliminando la cola de la distribución consistente en todas aquellas palabras de frecuencia menor a un cierto umbral (2, 3 o más). Pero en este caso, se ha perdido la *visión de conjunto*, ya que al agregar cada nuevo documento no se conoce la frecuencia futura de esas palabras, y por supuesto, esa palabra tendrá una frecuencia = 1. El resultado de este método es

[1]Este valor fue determinado empíricamente, mediante diversos experimentos. Su determinación no es directa, ya que su valor depende de parámetros establecidos en la etapa de indexación de documentos.

el crecimiento indiscriminado de la lista de palabras, y por lo tanto, de la dimensión del espacio vectorial del sistema.

La solución realizada consistió en la implementación de un *buffer* de documentos. Este buffer permitió disponer de una *visión de conjunto*, con el fin de poder conocer de alguna manera la frecuencia de las nuevas palabras agregadas al sistema. Este enfoque implica definir una ventana de tiempo, que corresponde al tiempo de vida del buffer. Cada cierto tiempo (por ejemplo, cada día), se recurre al conjunto de documentos del buffer para realizar una actualización de la lista de palabras. Dicha actualización es realizada por el método *updateWordList(...)* que agrega las nuevas palabras encontradas a la *wordList*.

Problema del ruido... eliminación de tópicos

Un segundo problema, es el derivado del procedimiento de clustering incremental. Como fue mencionado, al agregar nuevos documentos, si estos no superan un cierto umbral de cercanía, se crea un nuevo cluster. El problema de este enfoque, es que se agregan nuevos clusters sin mayor control, lo que provoca la aparición de *ruido*, en la forma de tópicos de un solo documento.

La solución implementada consiste en un procedimiento de *recolección de basura*, que permite desechar aquellos clusters de frecuencia muy baja. Estos clusters corresponden a aquellos *tópicos* que no alcanzaron la masa crítica necesaria para nacer. El procedimiento consiste en realizar periódicamente una limpieza de rutina, de esta manera se establece un cierto umbral de "frecuencia de partida" mínimo para la creación de nuevos tópicos.

Purgado de artícilos

Un tercer problema, que es abordado en esta etapa, es el de mantener actualizado el conjunto de artículos que forman los distintos tópicos. Para ello, el sistema contempla un sistema de eliminación de artículos (vencimiento), de una cierta cantidad de días (por ejemplo, 30). Con esto, a medida que se procesan los nuevos documentos, periódicamente se está revisando qué documentos han caducado. Este proceso implica a su vez, tener que ir eliminando adecuadamente los artículos de sus tópicos, lo que implica un recalculo del vector centroide en cada caso.

De esta manera, el proceso del clustering incremental es abordado con este conjunto de acciones "de mantención", que se preocupan de permitir la correctitud y actualización de los datos.

Capítulo 8

Experimentos y resultados

...o veamos lo que obtuvimos finalmente.

En el presente capítulo, se presentan los resultados de la ejecución del sistema. Primero se presenta la implementación final del sistema y el programa creado para realizar las pruebas. A continuación se muestran los resultados de aplicar el sistema para evaluar algunos de los parámetros utilizados en la aplicación. Finalmente, se presentan resultados de ejecutar el sistema tanto en su forma batch, como en su forma online, agregando un flujo continuo de artículos que simula la operación real del sistema, para un período de tiempo.

8.1. Métrica de evaluación de resultados

Para realizar la evaluación de la calidad del proceso de clustering, se definió una medida de calidad que intentó medir la calidad del resultado, recurriendo a los conceptos de la recuperación de información: se definió una medida inspirada en las medida de *precision* y *recall*, que permiten evaluar la calidad de lo recuperado, y la capacidad del sistema para recuperar los resultados relevantes, respectivamente. Concretamente, se utilizó el conjunto de referencia construido previamente como el conjunto de tópicos relevantes que se espera el sistema sea capaz de recuperar.

El procedimiento consistió en etiquetar cada uno de los grupos detectados, identificándolos como alguno de los tópicos de referencia, en base a un conteo de mayoría. Es decir, se contabiliza la pertenencia de cada uno de los artículos de un tópico, con respecto a la clasificación manual, lo que

da como resultado una etiqueta de tópico para cada cluster encontrado. En base a esto se puede reducir el problema a comparar ambos conjuntos.

Se decidió realizar el muestreo sobre el conjunto de los primeros 10 tópicos de cada conjunto. De esta manera, la precisión se midió revisando la calidad de cada uno de los tópicos detectados en el conjunto de los primeros 10 resultados. Por su parte, la cobertura (recall) se midió revisando en que posición eran recuperados cada uno de los primeros 10 tópicos de referencia. Así, para cada uno de los tópicos de cada conjunto (los recuperados y los relevantes), se tiene:

- Si el tópico revisado es parte de los primeros 10 del conjunto opuesto, se contabiliza como un acierto (1 pto.).
- Si el tópico no es parte de los primeros 10, se suma un puntaje igual a $10/posici'on$. Si el tópico encontrado es, por ejemplo, el número 11, se asigna un puntaje igual a 10/11 ¡1, descendiendo a medida que la posición desciende.

En base a esta razonamiento, se definió una función de calidad, Q de la siguiente manera:

$$Q = \frac{\sum_{i=1}^{n} Res * q(topico_i) + \sum_{i=1}^{n} REF * q(topico_i)}{(nRes + nREF)}$$

donde, $nRes$ corresponde a la cantidad de tópicos recuperados a considerar (los top-10), y $nREF$ corresponde a la cantidad de tópicos del conjunto de referencia a buscar (los top-10). Por su parte, $q(topico)$ corresponde a la puntuación definida anteriormente, con valores entre 0 y 1. La suma total de esta evaluación suma, en el caso óptimo, $nRes + nREF = 20$, por lo cual el resultado es normalizado para obtener, finalmente, valores entre 0 y 1.

8.2. Procedimiento

Para la ejecución de las pruebas, y la evaluación de los resultados, se implementó un programa que permitiera invocar al sistema desde consola, en Linux. Este programa (la clase *Laboratory*), permitió ejecutar tanto el clustering batch como el clustering en-línea, entregándole diversos parámetros, tanto al momento de su invocación, como a través de un archivo de configuración. Este archivo de configuración permite modificar directamente cada uno de los valores (parámetros) utilizados por la aplicación, sin necesidad de recompilar al sistema. En función de esos valores, el programa en tiempo de ejecución determina valores a utilizar, o decide la clase que implementa alguna de las interfaces de los procesos.

```
[9:56 turing:~/memoria/src/version1.5]> ./run.sh 1000 200 all
[main] TopicSystem - EXPERIMENTO # 0956
[main] TopicSystem - loadClustering()
[main] TopicSystem - clusteringBatch(sample:1000)
[main] TopicSystem - inicializando la base de datos
[main] TopicSystem - creando espacio vectorial...
[main] WVTString2Vector - WVTString2Vector()
[main] WVTString2Vector - load(sample:1000)
[main] WVTString2Vector - createWordList(list, config)
[main] DBInputList - leyendo la query de entrada
[main] WVTString2Vector - createVSpace(1)
[main] DBInputList - leyendo la query de entrada
[main] TopicSystem - agrupando vectores...
[main] ClutoVector2Clusters - run()
[main] ClutoVector2Clusters - saveClusters(con)
[main] TopicSystem - TIEMPOS:
[main] TopicSystem -    createWordList : 37.668s
[main] TopicSystem -    createVSpace   : 37.968s
[main] TopicSystem -    runClustering  : 97.674s
[main] TopicSystem -    saveClusters   : 12.859s
[main] TopicSystem - writeClusters()
[main] Laboratory - sistema segmentado... DONE
[main] Laboratory - calidad = 0.35984847
[main] Laboratory - tiempos:
[main] Laboratory -     TOTAL : 194.702s
```

Figura 8.1: Salida a pantalla del programa principal, desde la consola Linux.

El resultado, a nivel cuantitivo, se obtiene a través de la medida de calidad definida anteriormente, la cual es presentada en pantalla utilizando la librería *log4j*. Además se muestran otras informaciones producidas con fines de desarrollo y depuración, incluyendo tiempos de ejecución. Un ejemplo de la salida del programa, para un conjunto de 1000 artículos y 100 clusters, se muestra en la figura 8.1.

Para visualizar los resultados, se implementó un mecanismo basado en la caracterización de los clusters según sus centroides. Cada cluster detectado posee un centroide que lo identifica, y a su vez, cada centroide puede ser visualizado eligiendo las tres componentes de mayor peso, y mostrando la etiqueta que identifica a la palabra asociada a cada componente. Por ejemplo, el vector:

$$\vec{D} = \{rusi(0{,}286), davis(0{,}249), cop(0{,}235)\}$$

representa al tópico “Copa Davis” (cuarto tópico del conjunto de referencia). Cada palabra mostrada corresponde a la raíz de las palabras originales, y se muestra también el valor de cada componente. Con este procedimiento, se obtiene una fácil visualización de cada uno de los tópicos recuperados, lo que permite detectar por simple inspección visual el resultado encontrado.

Finalmente, los resultados de la aplicación pueden ser visualizados directamente en la aplicación java (a través de la consola Linux), o bien, pueden ser visualizados mediante una interfaz web realizada en php (la misma utilizada para la clasificación manual de artículos), que recoge los resul-

tados desde la base de datos. Además, el caso de uso de recuperación de los artículos relacionados también puede ser ejecutado tanto en la aplicación java, como a través de la interfaz web.

Para el mismo experimento, la pantalla web de resultados arroja:

#	id -	topico	#	...
1	167	ecuador (0.157) --- venezuel (0.141) --- colombi (0.134)	11	...
2	92	col (0.571) --- carac (0.107) --- futbol (0.073)	11	...
3	173	polit (0.131) --- ley (0.081) --- public (0.068)	11	...
4	165	bachelet (0.264) --- michell (0.153) --- president (0.152)	10	...
5	177	consum (0.129) --- agu (0.065) --- energi (0.063)	9	...
6	164	chiledeport (0.277) --- deport (0.132) --- fond (0.094)	9	...
7	7	puel (0.288) --- bolson (0.281) --- hit (0.281)	8	...
8	134	paraguay (0.243) --- sub (0.229) --- argentin (0.184)	8	...
9	160	transport (0.261) --- bus (0.166) --- ministeri (0.111)	8	...
10	117	rusi (0.286) --- davis (0.249) --- cop (0.235)	8	...
11	178	fin (0.185) --- seman (0.154) --- libr (0.091)	8	...
12	106	mistral (0.297) --- gabriel (0.288) --- nobel (0.196)	8	...
13	139	metr (0.355) --- estacion (0.185) --- pasajer (0.074)	8	...
14	142	cin (0.289) --- latinoamerican (0.163) --- premi (0.145)	8	...
15	169	pas (0.108) --- a€o (0.075) --- cos (0.074)	8	...
16	174	hombr (0.177) --- altim (0.108) --- lenguaj (0.093)	7	...
17	129	davis (0.286) --- cop (0.229) --- andreev (0.165)	7	...
18	176	seri (0.212) --- escritori (0.117) --- doctor (0.102)	7	...
19	107	tenis (0.278) --- mes (0.276) --- mundial (0.189)	7	...
20	172	protest (0.19) --- deten (0.105) --- aric (0.088)	7	...
21	116	fujimori (0.391) --- albert (0.163) --- peru (0.158)	7	...
22	162	internet (0.163) --- informacion (0.144) --- electron (0.091)	7	...
23	168	segund (0.234) --- elimin (0.172) --- feed (0.082)	7	...
24	154	cin (0.278) --- teatr (0.161) --- cineast (0.146)	7	...
25	179	santiag (0.137) --- podra (0.122) --- luis (0.123)	7	...

Figura 8.2: Pantalla de resultados para el experimento #0956

8.3. Experimentos

Una de las principales características del sistema desarrollado, es la gran cantidad de parámetros que debieron ser fijados. Algunos parámetros son valores numéricos, mientras que otros discriminan sobre distintas opciones de implementación. Cada uno de los módulos posee un conjunto de parámetros que permiten ajustar su funcionamiento. Para la etapa de la indexación de documentos, se deciden cosas como el punto de corte de la lista de palabras (la poda), o la función de pesos utilizada. Para el clustering batch, se establecen diversos parámetros del programa Cluto. Y en el clustering online, se definen los parámetros de las ventanas utilizadas, asi como el valor umbral del algoritmo de clustering incremental.

Del conjunto de parámetros, algunos se fijaron utilizando sus valores por defecto (en el caso de Cluto), o bien, se realizaron pruebas mínimas que permitieron detectar trivialmente el mejor caso. Lo que se presenta a continuación es el conjunto de experimentos que validan tres de los parámetros críticos del sistema: la función de pesos elegida, la función objetivo utilizada en el clustering batch, y el número de clusters utilizado para crear la segmentación base.

Evaluación de la función de pesos La función de pesos utilizada en la etapa de indexación de documentos, es el primer parámetro analizado en este capítulo. En el desarrollo del capítulo 5 se propuso la implementación de una función de peso, $PF-IDF$, basada en un conteo de frecuencias, pero ponderado por la posición de cada término dentro del documento.

Asumiendo independencia de esta decisión respecto a los demás parámetros, se procedió a fijar algunos valores, y se realizó el experimento utilizando la clásica función TF-IDF, versus tres casos de PF-IDF, usando distintos juegos de ponderaciones para las tres partes del artículo. La muestra utilizada fue de 1000 documentos, utilizando 100 clusters. Los valores presentados corresponden al promedio de tres ejecuciones para cada caso. El resultado fue:

función	tf-idf	pf-idf{2, 1, 1}	pf-idf{3, 2, 1}	pf-idf{6, 3, 2}
calidad	0.67	0.62	0.60	0.56

Los resultados muestran que los mejores valores de calidad se obtienen en PF-IDF, para la ponderación dada por $\{2, 1, 1\}$. Este resultado confirma en parte lo propuesto en el capítulo 5, respecto a que la función TF-IDF perdería validez para el caso de los artículos RSS. Sin embargo, los resultados cuatitativos no son lo suficientemente evidentes en este sentido. Una posible mejora podría conseguirse probando otras combinaciones de las ponderaciones.

Evaluación de la función objetivo En este caso, se propone analizar el comportamiento del sistema, para las dos funciones planteadas en el capítulo 6: I_1, e I_2. Ambas corresponden a medidas de similaridad interna, lo que asegura la calidad del resultado interno de las agrupaciones, es decir, la precisión dentro de cada cluster es alta.

El experimento realizado correspondió a la ejecución del sistema, para cada una de las dos funciones, utilizando una muestra de 1000 documentos, y alternando el número de clusters. Esto, debido a que los resultados de la función elegida posiblemente sean distintos dependiendo de número de clusters utilizado. De esta manera, para $k = 50$, $k = 100$ y $k = 200$, se procedió a probar ambas funciones:

función/calidad	$k = 50$	$k = 100$	$k = 200$	promedio
I_1	0.63	0.57	0.71	0.64
I_2	0.58	0.69	0.72	0.66

El resultado comprueba lo que se había mencionado: no existe un claro dominio de alguna de las funciones, para todos los casos. Una posible explicación puede ser que en general el comportamiento de I_2 es mejor si se considera el total de los resultados (se distribuye de una manera más parecida a lo que se espera), sin embargo, I_1 se comporta mejor en la detección de los primeros grupos, lo cual es precisamente el resultado buscado. Además, una mejora que se puede implementar fácilmente

consiste en podar los últimos clusters detectados, sabiendo a priori que pueden ser grupos de baja calidad.

Evaluación del número de clusters El número de clusters a utilizar, es uno de los valores importantes del sistema. La determinación de su valor pasó por varias etapas, hasta llegar a los valores encontrados finalmente. En un primer momento, se postulaba que la cantidad de clusters a utilizar debiera ser cercana al número de clusters encontrado en la clasificación manual (del orden del 35 % del total de documentos). Sin embargo, los resultados prácticos mostraron que el valor de K debía ser mucho menor. Esto, motivado por la forma en que funcionan los algoritmos utilizados en la etapa de clustering: muchos de los artículos catalogados en tópicos de uno o dos elementos, son agrupados en grupos de baja calidad, que cumplen una función de sumidero para artículos que no forman agrupaciones naturalmente.

Otra idea que se tenía previamente, es que el valor de K óptimo sería un valor proporcional al número de documentos considerados. Es decir, correspondería a un porcentaje del número de documentos procesados. Los resultados preliminares sugirieron utilizar en las pruebas un valor de $k = 10\,\%$ del número de documentos. Sin embargo, esto podría no ser necesariamente cierto, ya que al aumentar el número de documentos considerados, aparecerían fenómenos relacionados con la naturaleza del lenguaje: a medida que se consideran más documentos, se reduciría la probabilidad de que aparezcan nuevos tópicos.

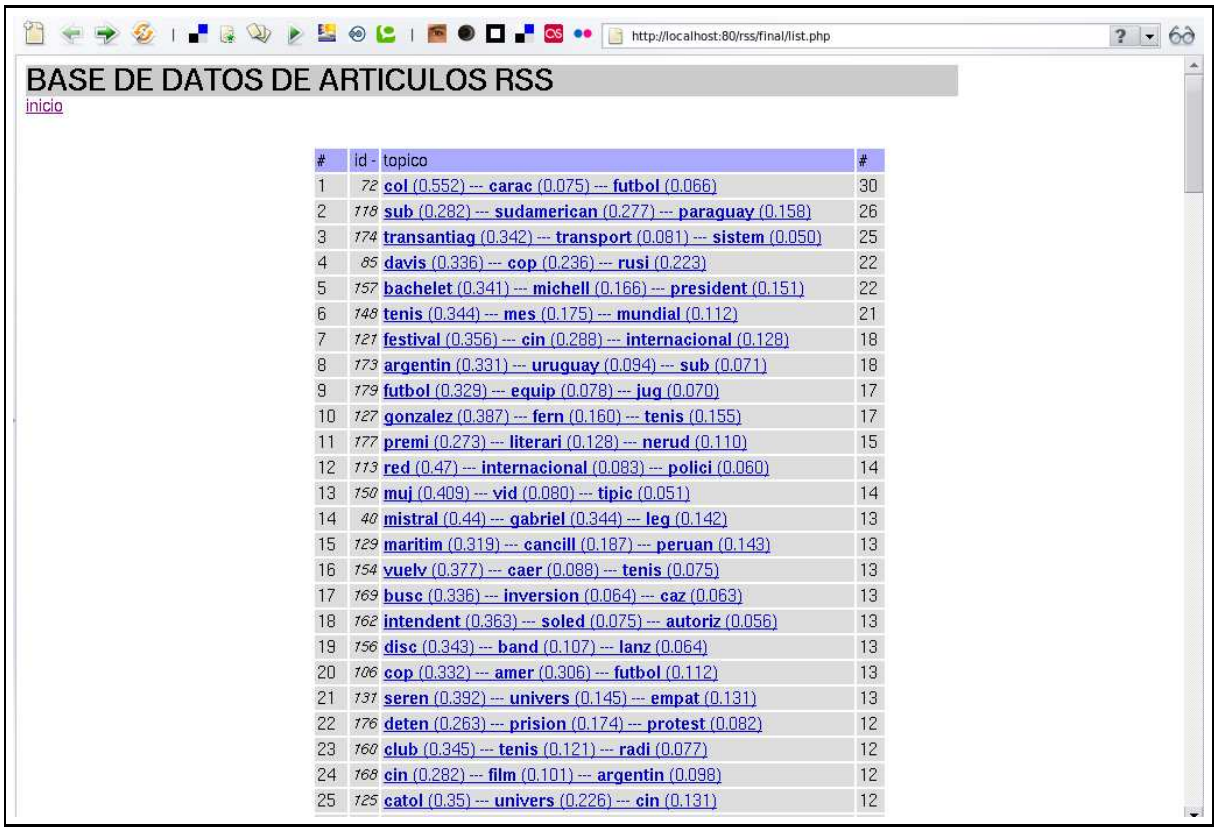
El probar el sistema, para un conjunto de 1000 documentos, y para diferentes valores de k (y promediando diversas instancias del experimento), se obtuvieron los siguientes resultados:

calidad q	$k = 50(5\,\%)$	$k = 100(10\,\%)$	$k = 200(20\,\%)$	$k = 400(40\,\%)$
...	0.59	0.67	0.69	0.54

Los resultados apoyan en parte lo propuesto: el valor ideal de k, para una muesrtra de 1000 documentos, se ubica entre el 10 al 20 % del total de artículos. De aquí en adelante, el resto de los experimentos se realizó utilizando un valor de $k = 10\,\%$. Lo que queda pendiente, es poder establecer el comportamienteo de k para grandes conjuntos de datos.

8.4. Resultados

A continuación, se presenta la ejecución del sistema desarrollado, para el caso batch, sobre el total de artículos usados en la construcción del conjunto de referencia. Esto corresponde a los tres primeros meses del año 2007. Se utilizó un total de 200 clusters (un 10 % del total de documentos), podando a su vez el 10 % de los tópicos de peor calidad.



#	id -	topico	#
1	72	col (0.552) --- carac (0.075) --- futbol (0.066)	30
2	118	sub (0.282) --- sudamerican (0.277) --- paraguay (0.158)	26
3	174	transantiag (0.342) --- transport (0.081) --- sistem (0.050)	25
4	85	davis (0.336) --- cop (0.236) --- rusi (0.223)	22
5	157	bachelet (0.341) --- michell (0.166) --- president (0.151)	22
6	148	tenis (0.344) --- mes (0.175) --- mundial (0.112)	21
7	121	festival (0.356) --- cin (0.288) --- internacional (0.128)	18
8	173	argentin (0.331) --- uruguay (0.094) --- sub (0.071)	18
9	179	futbol (0.329) --- equip (0.078) --- jug (0.070)	17
10	127	gonzalez (0.387) --- fern (0.160) --- tenis (0.155)	17
11	177	premi (0.273) --- literari (0.128) --- nerud (0.110)	15
12	113	red (0.47) --- internacional (0.083) --- polici (0.060)	14
13	150	muj (0.409) --- vid (0.080) --- tipic (0.051)	14
14	40	mistral (0.44) --- gabriel (0.344) --- leg (0.142)	13
15	129	maritim (0.319) --- cancill (0.187) --- peruan (0.143)	13
16	154	vuelv (0.377) --- caer (0.088) --- tenis (0.075)	13
17	169	busc (0.336) --- inversion (0.064) --- caz (0.063)	13
18	162	intendent (0.363) --- soled (0.075) --- autoriz (0.056)	13
19	156	disc (0.343) --- band (0.107) --- lanz (0.064)	13
20	106	cop (0.332) --- amer (0.306) --- futbol (0.112)	13
21	131	seren (0.392) --- univers (0.145) --- empat (0.131)	13
22	176	deten (0.263) --- prision (0.174) --- protest (0.082)	12
23	160	club (0.345) --- tenis (0.121) --- radi (0.077)	12
24	168	cin (0.282) --- film (0.101) --- argentin (0.098)	12
25	125	catol (0.35) --- univers (0.226) --- cin (0.131)	12

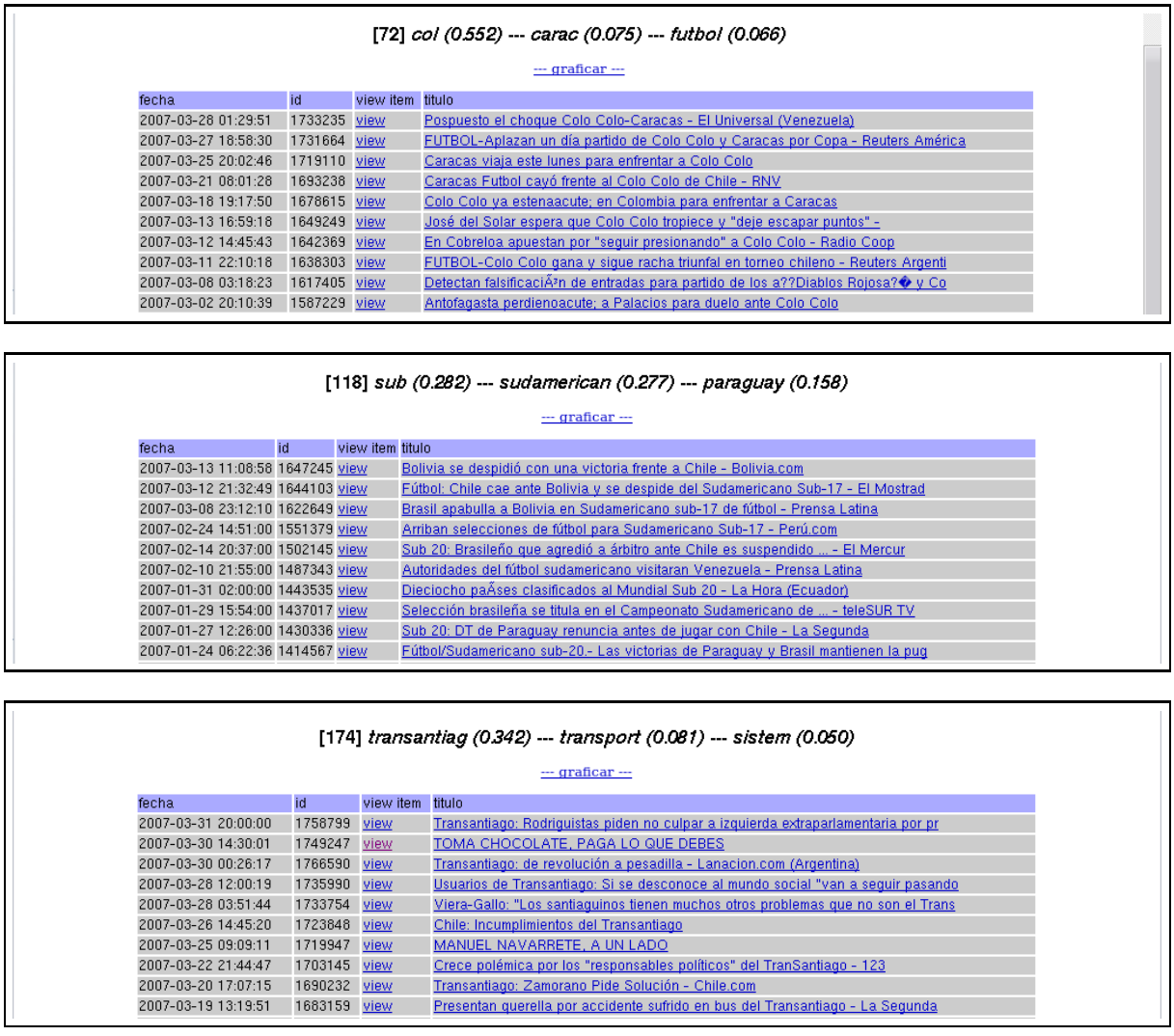
Figura 8.3: Pantalla con los resultados obtenidos en la ejecución del sistema en su forma batch, para el conjunto completo de 2000 artículos.

El resultado aquí obtenido, puede ser comparado con el ranking presentado en el capítulo 3. Se puede apreciar que la lista de los primeros 10 tópicos detectados corresponden a:

- colo colo $\surd$
- sudamericano sub 20 $\surd$
- transantiago $\surd$
- copa davis $\surd$
- michell bachelet $\surd$
- mundial de tenis de mesa
- festival de cine
- argentina
- futbol
- fernando gonzalez $\surd$

Esto muestra un total de 6 aciertos, sobre el total de los primeros 10 resultados. A su vez, dos resultados (*festival de cine*, *tenis de mesa*) corresponden a tópicos de posiciones inferiores, mientras que *argentina* y *futbol*, corresponden a agrupaciones creadas naturalmente, pero que se distancian de lo realizado en la clasificación manual.

Al mostrar el detalle de cada tópico encontrado, se puede verificar el buen comportamiento de la función de similaridad interna. Los tres primeros resultados arrojan:



[72] *col (0.552) --- carac (0.075) --- futbol (0.066)*

--- graficar ---

fecha	id	view item	titulo
2007-03-28 01:29:51	1733235	view	Pospuesto el choque Colo Colo-Caracas - El Universal (Venezuela)
2007-03-27 18:58:30	1731664	view	FUTBOL-Aplazan un día partido de Colo Colo y Caracas por Copa - Reuters América
2007-03-25 20:02:46	1719110	view	Caracas viaja este lunes para enfrentar a Colo Colo
2007-03-21 08:01:28	1693238	view	Caracas Futbol cayó frente al Colo Colo de Chile - RNV
2007-03-18 19:17:50	1678615	view	Colo Colo ya estenaacute; en Colombia para enfrentar a Caracas
2007-03-13 16:59:18	1649249	view	José del Solar espera que Colo Colo tropiece y "deje escapar puntos" -
2007-03-12 14:45:43	1642369	view	En Cobreloa apuestan por "seguir presionando" a Colo Colo - Radio Coop
2007-03-11 22:10:18	1638303	view	FUTBOL-Colo Colo gana y sigue racha triunfal en torneo chileno - Reuters Argenti
2007-03-08 03:18:23	1617405	view	Detectan falsificaciÃ³n de entradas para partido de los a??Diablos Rojosa?€ y Co
2007-03-02 20:10:39	1587229	view	Antofagasta perdienoacute; a Palacios para duelo ante Colo Colo

[118] *sub (0.282) --- sudamerican (0.277) --- paraguay (0.158)*

--- graficar ---

fecha	id	view item	titulo
2007-03-13 11:08:58	1647245	view	Bolivia se despidió con una victoria frente a Chile - Bolivia.com
2007-03-12 21:32:49	1644103	view	Fútbol: Chile cae ante Bolivia y se despide del Sudamericano Sub-17 - El Mostrad
2007-03-08 23:12:10	1622649	view	Brasil apabulla a Bolivia en Sudamericano sub-17 de fútbol - Prensa Latina
2007-02-24 14:51:00	1551379	view	Arriban selecciones de fútbol para Sudamericano Sub-17 - Perú.com
2007-02-14 20:37:00	1502145	view	Sub 20: Brasileño que agredió a árbitro ante Chile es suspendido ... - El Mercur
2007-02-10 21:55:00	1487343	view	Autoridades del fútbol sudamericano visitaran Venezuela - Prensa Latina
2007-01-31 02:00:00	1443535	view	Dieciocho paÃses clasificados al Mundial Sub 20 - La Hora (Ecuador)
2007-01-29 15:54:00	1437017	view	Selección brasileña se titula en el Campeonato Sudamericano de ... - teleSUR TV
2007-01-27 12:26:00	1430336	view	Sub 20: DT de Paraguay renuncia antes de jugar con Chile - La Segunda
2007-01-24 06:22:36	1414567	view	Fútbol/Sudamericano sub-20.- Las victorias de Paraguay y Brasil mantienen la pug

[174] *transantiag (0.342) --- transport (0.081) --- sistem (0.050)*

--- graficar ---

fecha	id	view item	titulo
2007-03-31 20:00:00	1758799	view	Transantiago: Rodriguistas piden no culpar a izquierda extraparlamentaria por pr
2007-03-30 14:30:01	1749247	view	TOMA CHOCOLATE, PAGA LO QUE DEBES
2007-03-30 00:26:17	1766590	view	Transantiago: de revolución a pesadilla - Lanacion.com (Argentina)
2007-03-28 12:00:19	1735990	view	Usuarios de Transantiago: Si se desconoce al mundo social "van a seguir pasando
2007-03-28 03:51:44	1733754	view	Viera-Gallo: "Los santiaguinos tienen muchos otros problemas que no son el Trans
2007-03-26 14:45:20	1723848	view	Chile: Incumplimientos del Transantiago
2007-03-25 09:09:11	1719947	view	MANUEL NAVARRETE, A UN LADO
2007-03-22 21:44:47	1703145	view	Crece polémica por los "responsables políticos" del TranSantiago - 123
2007-03-20 17:07:15	1690232	view	Transantiago: Zamorano Pide Solución - Chile.com
2007-03-19 13:19:51	1683159	view	Presentan querella por accidente sufrido en bus del Transantiago - La Segunda

Figura 8.4: Detalle (10 resultados) de los primeros tres tópicos detectados por el sistema.

Una particularidad observada en los resultados, para el caso del segundo tópico *sudamericano sub 20*, en realidad muestra resultados que mezclan dos eventos distintos: el sudamericano sub 20 ocurrido en el mes de enero, y el sudamericano sub 17, ocurrido durante marzo. Si bien esto puede parecer un error, es un resultado esperable, dado que no se realiza en esta etapa el purgado de artículos, por lo que es difícil para el sistema, identificar que ambos casos corresponden a eventos distintos. En la práctica, esto no ocurriría, gracias a la eliminación que se realiza de los artículos obsoletos, en el procesamiento online.

También es posible graficar el histograma correspondiente a estos resultados. Dos de ellos se pueden comparar directamente con los histogramas presentados en la figura 8.5.

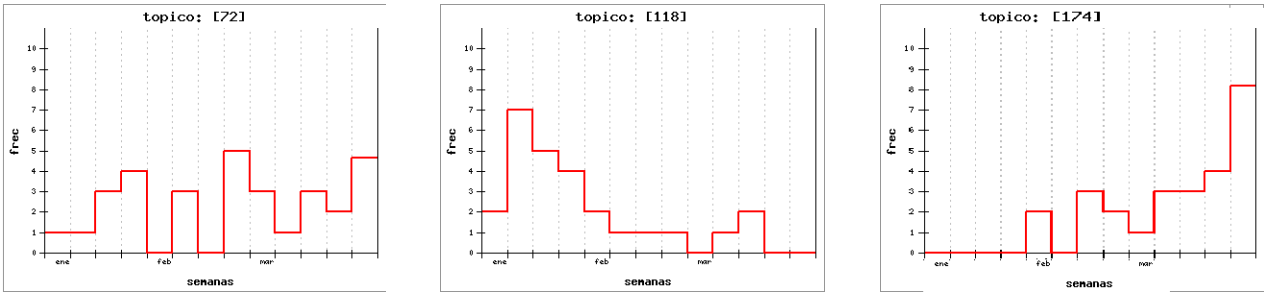


Figura 8.5: Gráficos de frecuencias para los tres primeros tópicos encontrados.
colo colo - sudamericano sub 20 - transantiago

Finalmente, se puede mostrar una posible implementación de la propuesta de portada realizada por el sistema. Esta implementación corresponde a una componente independiente de la aplicación principal, ya que el objetivo del sistema desarrollado se centra en obtener y entregar la información de los tópicos, mientras que la construcción de la portada y su despliegue, son tareas de la componente de vista del sistema de orbitando.com.

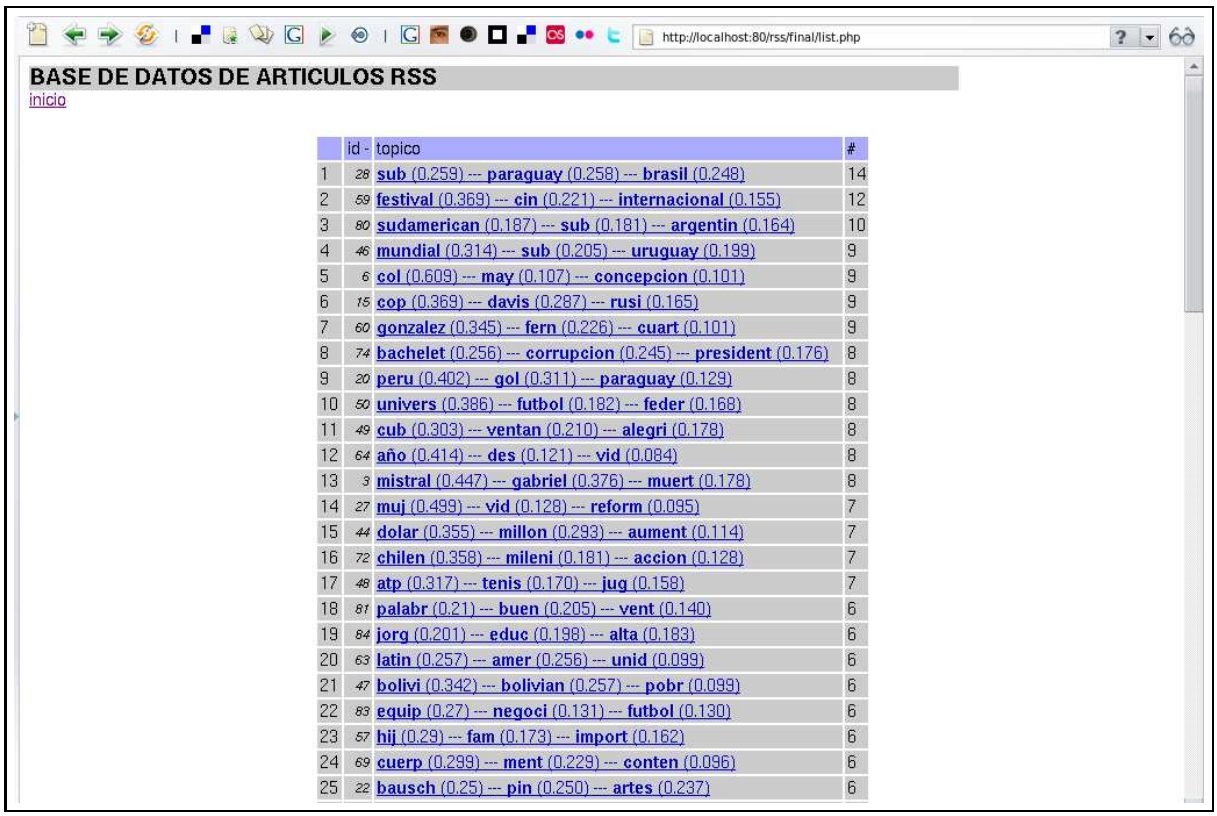
De esta manera, se tiene:

Figura 8.6: Pantalla que muestra una posible implementación de la ventana de titulares representativos de cada tópico.

Ejecución online:

El otro caso de ejecución del sistema, corresponde al caso online, en el cual se realizó primero una fase de clustering batch, y posteriormente, se agregaron artículos, en orden cronológico, simulando la llegada continua de documentos. A continuación, se muestran los resultados de una simulación realizada considerando el mes de enero-2007 para la etapa batch, y los siguientes dos meses procesados en la modalidad online.

El resultado correspondiente al procesamiento batch del mes de enero, es el siguiente:



	id -	topico	#
1	28	sub (0.259) --- paraguay (0.258) --- brasil (0.248)	14
2	59	festival (0.369) --- cin (0.221) --- internacional (0.155)	12
3	80	sudamerican (0.187) --- sub (0.181) --- argentin (0.164)	10
4	46	mundial (0.314) --- sub (0.205) --- uruguay (0.199)	9
5	6	col (0.609) --- may (0.107) --- concepcion (0.101)	9
6	15	cop (0.369) --- davis (0.287) --- rusi (0.165)	9
7	60	gonzalez (0.345) --- fern (0.226) --- cuart (0.101)	9
8	74	bachelet (0.256) --- corrupcion (0.245) --- president (0.176)	8
9	20	peru (0.402) --- gol (0.311) --- paraguay (0.129)	8
10	50	univers (0.386) --- futbol (0.182) --- feder (0.168)	8
11	49	cub (0.303) --- ventan (0.210) --- alegri (0.178)	8
12	64	año (0.414) --- des (0.121) --- vid (0.084)	8
13	3	mistral (0.447) --- gabriel (0.376) --- muert (0.178)	8
14	27	muj (0.499) --- vid (0.128) --- reform (0.095)	7
15	44	dolar (0.355) --- millon (0.293) --- aument (0.114)	7
16	72	chilen (0.358) --- mileni (0.181) --- accion (0.128)	7
17	48	atp (0.317) --- tenis (0.170) --- jug (0.158)	7
18	81	palabr (0.21) --- buen (0.205) --- vent (0.140)	6
19	84	jorg (0.201) --- educ (0.198) --- alta (0.183)	6
20	63	latin (0.257) --- amer (0.256) --- unid (0.099)	6
21	47	bolivi (0.342) --- bolivian (0.257) --- pobr (0.099)	6
22	83	equip (0.27) --- negoci (0.131) --- futbol (0.130)	6
23	57	hij (0.29) --- fam (0.173) --- import (0.162)	6
24	69	cuerp (0.299) --- ment (0.229) --- conten (0.096)	6
25	22	bausch (0.25) --- pin (0.250) --- artes (0.237)	6

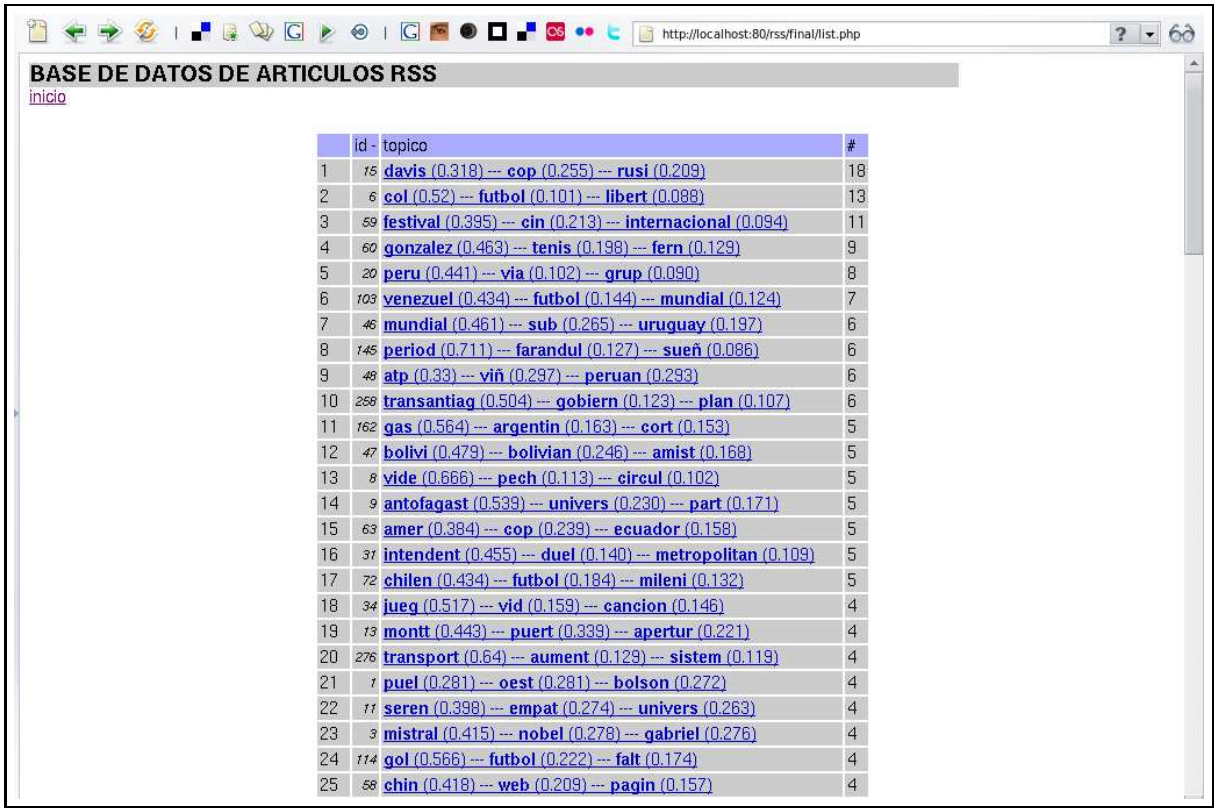
Figura 8.7: Pantalla con los resultados correspondientes al mes de enero.

Este resultado corresponde al mismo tipo de experimento realizado anteriormente, pero sólo considerando el primer mes. Lo interesante es comparar este resultado con el ranking de la tabla 3.4, que muestra el ranking realizado manualmente para el mismo período.

A continuación, se ejecutó el proceso para el caso online. En este punto es donde se ejecutan los procesos propuestos en el capítulo 7, en particular, lo referente a las ventanas de tiempo. Esto significó que a partir del primer día de febrero, se considera una ventana móvil de 30 días que se desplaza eliminando los artículos de antiguedad mayor a un mes. Al mismo tiempo, el sistema realiza la tarea de expansión de la lista de palabras, pasando de 1245 a 1838 palabras, al cabo del primer mes de procesamiento. Finalmente, se realiza también la poda de tópicos, eliminando

aquellos tópicos que no superan los dos elementos.

El resultado, al día 1 de marzo fue el siguiente:

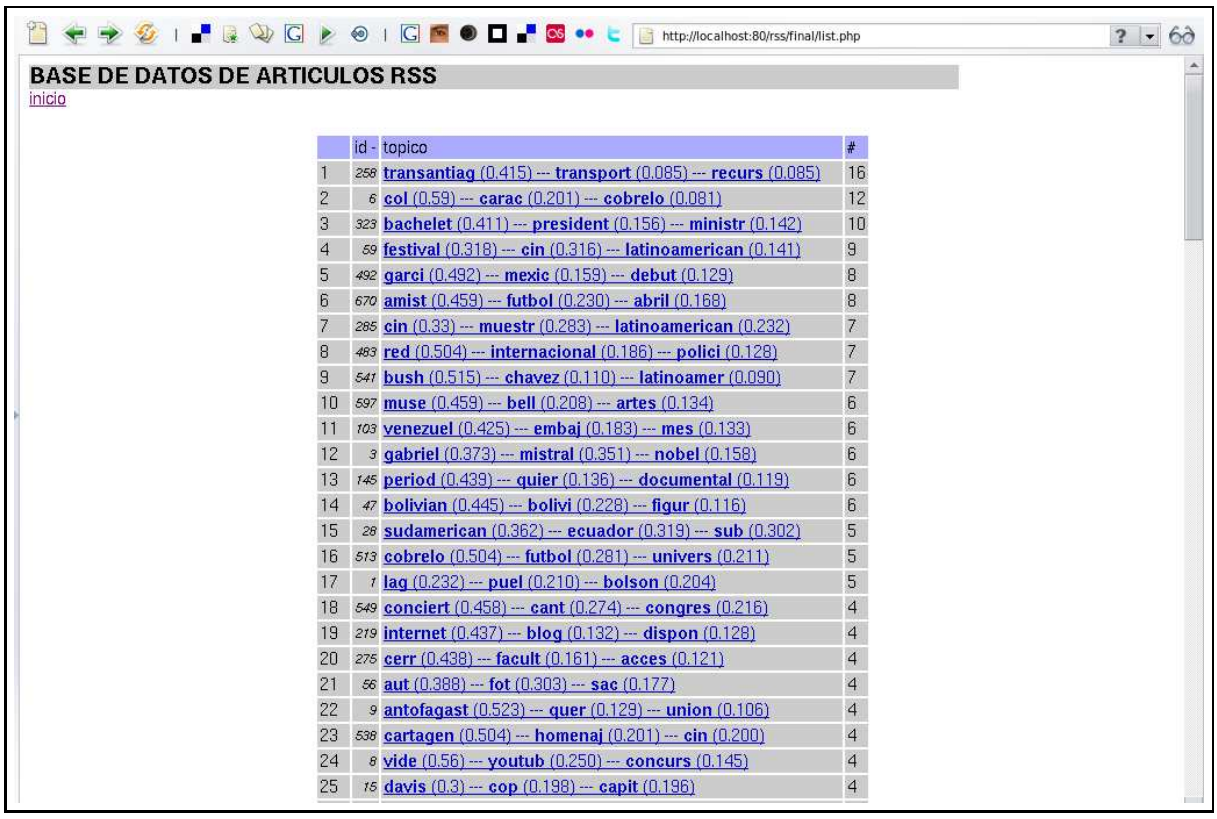


	id -	topico	#
1	15	davis (0.318) --- cop (0.255) --- rusi (0.209)	18
2	6	col (0.52) --- futbol (0.101) --- libert (0.088)	13
3	59	festival (0.395) --- cin (0.213) --- internacional (0.094)	11
4	60	gonzalez (0.463) --- tenis (0.198) --- fern (0.129)	9
5	20	peru (0.441) --- via (0.102) --- grup (0.090)	8
6	103	venezuel (0.434) --- futbol (0.144) --- mundial (0.124)	7
7	46	mundial (0.461) --- sub (0.265) --- uruguay (0.197)	6
8	145	period (0.711) --- farandul (0.127) --- sueñ (0.086)	6
9	48	atp (0.33) --- viñ (0.297) --- peruan (0.293)	6
10	258	transantiag (0.504) --- gobiern (0.123) --- plan (0.107)	6
11	162	gas (0.564) --- argentin (0.163) --- cort (0.153)	5
12	47	bolivi (0.479) --- bolivian (0.246) --- amist (0.168)	5
13	8	vide (0.666) --- pech (0.113) --- circul (0.102)	5
14	9	antofagast (0.539) --- univers (0.230) --- part (0.171)	5
15	63	amer (0.384) --- cop (0.239) --- ecuador (0.158)	5
16	31	intendent (0.455) --- duel (0.140) --- metropolitan (0.109)	5
17	72	chilen (0.434) --- futbol (0.184) --- mileni (0.132)	5
18	34	jueg (0.517) --- vid (0.159) --- cancion (0.146)	4
19	13	montt (0.443) --- puert (0.339) --- apertur (0.221)	4
20	276	transport (0.64) --- aument (0.129) --- sistem (0.119)	4
21	1	puel (0.281) --- oest (0.281) --- bolson (0.272)	4
22	11	seren (0.398) --- empat (0.274) --- univers (0.263)	4
23	3	mistral (0.415) --- nobel (0.278) --- gabriel (0.276)	4
24	114	gol (0.566) --- futbol (0.222) --- falt (0.174)	4
25	58	chin (0.418) --- web (0.209) --- pagin (0.157)	4

Figura 8.8: Pantalla con los resultados correspondientes al mes de febrero.

Lo que se muestra aquí, permite apreciar la aparición del tópico *transantiago*, además del dominio del tópico relevante de ese mes: *copa davis*. También aparecen *copa america* (lugar 6), *atp de viña del mar* (lugar 9), mientras que *sudamericano sub 20* ha descendido lugares, como era esperable (es un evento del mes de enero).

En último lugar, se procede a ejecutar nuevamente el sistema, para el mes de marzo, obteniendo el siguiente resultado, correspondiente al día 1 de abril:



	id -	topico	#
1	258	transantiag (0.415) --- transport (0.085) --- recurs (0.085)	16
2	6	col (0.59) --- carac (0.201) --- cobrelo (0.081)	12
3	323	bachelet (0.411) --- president (0.156) --- ministr (0.142)	10
4	59	festival (0.318) --- cin (0.316) --- latinoamerican (0.141)	9
5	492	garci (0.492) --- mexic (0.159) --- debut (0.129)	8
6	670	amist (0.459) --- futbol (0.230) --- abril (0.168)	8
7	285	cin (0.33) --- muestr (0.283) --- latinoamerican (0.232)	7
8	483	red (0.504) --- internacional (0.186) --- polici (0.128)	7
9	541	bush (0.515) --- chavez (0.110) --- latinoamer (0.090)	7
10	597	muse (0.459) --- bell (0.208) --- artes (0.134)	6
11	103	venezuel (0.425) --- embaj (0.183) --- mes (0.133)	6
12	3	gabriel (0.373) --- mistral (0.351) --- nobel (0.158)	6
13	145	period (0.439) --- quier (0.136) --- documental (0.119)	6
14	47	bolivian (0.445) --- bolivi (0.228) --- figur (0.116)	6
15	28	sudamerican (0.362) --- ecuador (0.319) --- sub (0.302)	5
16	513	cobrelo (0.504) --- futbol (0.281) --- univers (0.211)	5
17	1	lag (0.232) --- puel (0.210) --- bolson (0.204)	5
18	549	conciert (0.458) --- cant (0.274) --- congres (0.216)	4
19	219	internet (0.437) --- blog (0.132) --- dispon (0.128)	4
20	275	cerr (0.438) --- facult (0.161) --- acces (0.121)	4
21	56	aut (0.388) --- fot (0.303) --- sac (0.177)	4
22	9	antofagast (0.523) --- quer (0.129) --- union (0.106)	4
23	538	cartagen (0.504) --- homenaj (0.201) --- cin (0.200)	4
24	8	vide (0.56) --- youtub (0.250) --- concurs (0.145)	4
25	15	davis (0.3) --- cop (0.198) --- capit (0.196)	4

Figura 8.9: Pantalla con los resultados correspondientes al mes de marzo.

Dado que la ventana de tiempo considerada es de 30 días, es posible comparar los resultados presentados con cada uno de los meses, según la tabla 3.4. Entre los fenómenos observados, aparece el crecimiento del tópico *transantiago*, el cual se consolida en el mes de marzo en el primer lugar, o el tópico *copa davis*, que ahora muestra una fuerte caída. Finalmente, se observa la aparición de tópicos propios del mes de marzo, como el *partido amistoso chile-brasil* (lugar 6), o el resurgimiento de *michell bachelet*, que había desaparecido de los primeros lugares durante el mes de febrero.

Capítulo 9

Conclusiones

...o que fue lo que aprendí al realizar este proyecto.

El desarrollo de este proyecto permitió realizar un primer acercamiento al estudio del procesamiento de artículos sindicados, y todos los temas relacionados con el cambio que está viviendo la web en el presente, pasando de una web estática, a una *web viva*. Este fue el tema motivador del proyecto, y permitió conocer con detalle este proceso de cambio en los medios de comunicación y cómo este cambio se ve reflejado en los contenidos de la web y el cómo los usuarios se relacionan con la red.

Para llevar a cabo la implementación del sistema, fue necesario llegar a conocer con detalle un conjunto de tecnologías involucradas. En primer lugar, se abordó el tema de la recuperación de información, y se conocieron las diversas técnicas que permiten representar documentos en sistemas informáticos. En este campo, se conoció en detalle el proceso de extracción de características, en la forma de un vector de palabras, y todos los problemas asociados a esta tarea.

Las técnicas de clustering corresponden a la tarea de la minería de datos utilizada para el procesamiento de los documentos, con el fin de obtener aquella muestra de tópicos buscada. Durante el desarrollo del proyecto se conocieron los distintos algoritmos de clustering, estudiando las alternativas disponibles en búsqueda de las mejores opciones, y se pudieron probar distintas opciones a través de la herramienta utilizada. Además, se conoció en detalle también, el problema del clustering incremental, uno de los principales desafíos de este proyecto, y se comprobó que se trata de un tema en pleno desarrollo, lo cual plantea una serie de inquietudes respecto a las mejoras que se podrían implementar a futuro en el sistema.

Finalmente, se implementó un prototipo del sistema, lo que permitió llevar a cabo las pruebas que validaron lo estudiado en el desarrollo del proyecto. Como meta final del proyecto, se planea realizar la implementación formal del sistema en el sitio web *orbitando.com*, con el fin de aprovechar los resultados obtenidos en el contexto de la web sindicada chilena, generando la componente de la portada del sitio correspondiente a los principales temas o tópicos de actualidad, buscando ser una "ventana a la blogósfera".

Esta memoria buscó aportar con un estudio sobre la sindicación de contenidos, tratando de aplicar técnicas conocidas de la minería de datos al caso particular de los artículos RSS. Se pudo comprobar que estas técnicas necesitan en general un conjunto de ajustes específicos, que tienen relación principalmente con el reducido tamaño de los documentos, y con el carácter informativo: independientemente de que se trate de noticias o artículos de blogs, el factor común siempre es el tratar de comunicar algo. En este caso, utilizando la sindicación.

El sistema si bien fue desarrollado en su totalidad, aún admite una serie de mejoras en búsqueda de mejores resultados. Por ejemplo, mejorando las técnicas de clustering incremental, o bien, mejorando aún más el proceso de transformación a vectores (por ejemplo, refinando la lista de palabras a filtrar o modificando las ponderaciones de la función de pesos). De esta manera, este proyecto puede ser el punto de partida de estudios posteriores sobre el tema de la web sindicada, al hacer un primer planteamiento del conjunto de problemas propios de este tipo de datos.

Bibliografía

[1] J. Allan, J. Carbonell, G. Doddington, J. Yamron, J., Y. Yang; "*Topic Detection and Tracking Pilot Study, Final Report*", Topic Detection and Tracking group, DARPA, USA

[2] R. Baeza-Yates; "*Recuperación de la Información: Modelos, Estructuras de Datos, Algoritmos y Búsqueda en la Web*", Apuntes del curso Recuperación de Información, Universidad de Chile, 2005

[3] R. Baeza-Yates, B. Ribeiro-Neto; "*Modern Information Retrieval*", Addison-Wesley, 1999

[4] P. Berkhin; "*Survey of Clustering data mining techniques* ", Accrue Software In. 2002

[5] S. Bowman, C. Willis; "*Nosotros, el medio*", The Media Center, American Press Institute, USA, 2003

[6] L. O'Callaghan, N. Mishra, A. Mey erson, S. Guha, R.Motwani; "*Streaming-Data Algorithms For High-Quality Clustering*", Stanford University, HP Labs, University of Pennsylvania, 2002

[7] M. Dash, H. Liu, X. Xu; "*'1+1>2': Merging distance and density based clustering*", National University of Singapure, Singapur; Arizona State University, USA; Siemens AG, Alemania, 2001

[8] M. Ester, H.P. Kriegel, J. Sander, M. Wimmer, X. Xu; "*Incremental clustering for mining in a data warehousing environment*", University of Munich, Alemania , 1998

[9] D. Fisher "*Knowledge Acquisition via Incremental Conceptual Clustering*", 1987

[10] B. Fung, K. Wang, M. Ester; "*Hierarchical document clustering*", Simon Fraser University, Canada

[11] M. Garre, J.J. Cuadrado, M.A. Sicilia; "*Comparación de diferentes algoritmos de clustering en la Estimación de coste en el desarrollo de software*", Universidad de Alcala, España

[12] R. Gil-García, J. Badía; "*Algoritmos de Agrupamiento*", Universidad Jaime I, 2002

[13] C.Gupta; "*GenIc: A Single Pass Generalized Incremental Algorithm for Clustering*", University of Illinois, USA

[14] K. Hammouda, M. Kamel; "*Incremental Document Clustering using Cluster Similarity Histograms*", University of Waterloo, Canada

[15] Heaps, H.S. "*Information Retrieval : Computational and Theoretical Aspects*", New York, Academic Press, 1978

[16] G. Karypis; "*CLUTO: A clustering toolkit*", Technical Report; University of Minnesota, USA, 2003

[17] B. Kovachm, T. Rosentiel *"The Elements of the Journalism* (2001)

[18] P. Luhn, "*The automatic creation of literature abstracts*", IBM Journal of Research and Development, 2, 1pags. 59-165. 1958.

[19] P. Luhn, *"Keyword-in-Context Index for Technical Literature."* Report RC 127. New York: IBM Corp., Advanced System Development Division, 1959.

[20] M.F. Porter, "*An algorithm for suffix stripping*", 1980

[21] N. Sahoo; "*Incremental Hierarchical Clustering Text Documents* ", 2006

[22] G. Salton, A. Wong, C.S. Yang; "*A vector space model for automatinc indexing*", Cornell University, USA

[23] I. Witten, E. Frank; "*WEKA: machine learning algorithms in java*", University of Waikato, 2000

[24] M. Wurst; "*The word vector tool*", University of Dortmund, 2006

[25] Y. Yang, T. Pierce, J. Carbonell; "*A study on retrospective and on-line event detection* ", Carnegie Mellon University, USA, 1998

[26] Y. Zhao, G. Karypis; "*Criterion Functions for Document Clustering*", University of Minnesota, 2002

[27] S. Zhong; "*Efficient online spherical k-means clustering*", Florida Atlantic University,USA, 2005

[28] G.K. Zipf, "*Human behavior and the principle of least effort*", Cambridge, MA, Addison-Wesley, 1949.

Referencias Web

[29] Cluto, Family of Data Clustering Software Tools
http://glaros.dtc.umn.edu/gkhome/views/cluto

[30] David Sifry - creador de technorati.com. Entrevista diario El Pais, España. (01/06/2006)
http://www.elpais.com/articulo/portada/David/Sifry/fundador/Technorati/soy/editor/siglo/XXI/elpcibpor/20060601elpcibpor_1/Tes

[31] David Sifry - The State of Blogosphere (abril 2007). Blog de David Sifry
http://www.sifry.com/stateoftheliveweb/

[32] Proyecto Snowball
http://snowball.tartarus.org/

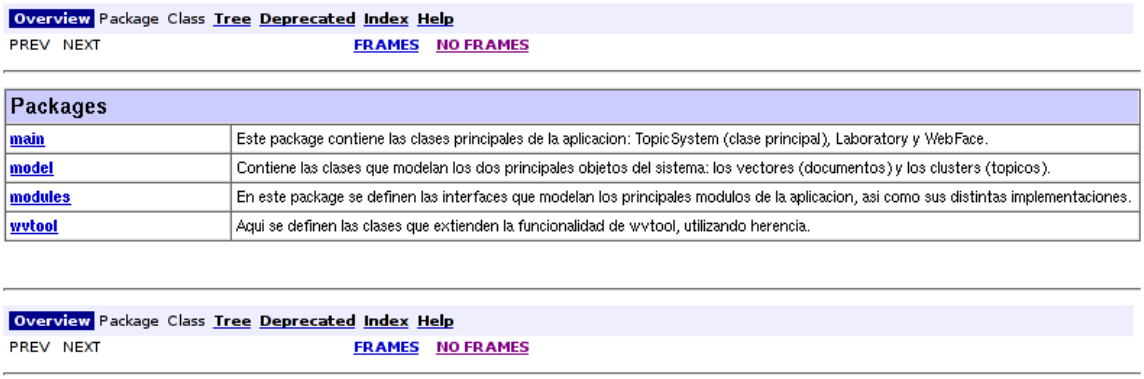
[33] The Porter Stemming Algorithm
http://www.tartarus.org/ martin/PorterStemmer/index-old.html

[34] WVTool, The Word Vector Tool
http://nemoz.org/joomla/index.php?option=com_content&task=view&id=43&Itemid=83

Apéndice A

Documentación de la aplicación

La siguiente es la documentación de la aplicación, en formato *javadoc*. Este es un resumen de cada *package* del sistema, junto a cada una de sus claes. La versión completa de la documentación se encuentra en: *http://www.danielgomez.cl/memoria/javadoc/*.



Overview Package Class **Tree** **Deprecated** **Index** **Help**

PREV NEXT **FRAMES** **NO FRAMES**

Packages	
main	Este package contiene las clases principales de la aplicacion: TopicSystem (clase principal), Laboratory y WebFace.
model	Contiene las clases que modelan los dos principales objetos del sistema: los vectores (documentos) y los clusters (topicos).
modules	En este package se definen las interfaces que modelan los principales modulos de la aplicacion, asi como sus distintas implementaciones.
wvtool	Aqui se definen las clases que extienden la funcionalidad de wvtool, utilizando herencia.

Overview Package Class **Tree** **Deprecated** **Index** **Help**

PREV NEXT **FRAMES** **NO FRAMES**

Package main

Este package contiene las clases principales de la aplicacion: TopicSystem (clase principal), Laboratory y WebFace.

See:
Description

Class Summary	
Laboratory	Programa para realizar pruebas del sistema de segmentacion de articulos RSS.
TopicSystem	Clase principal del sistema de segmentacion de articulos RSS.
Util	Esta clase provee de metodos utilitarios para la aplicacion.
WebFace	Programa que representa la capa web del sistema de segmentacion RSS.

Exception Summary	
TopicSystemException	Esta clase modela una excepcion ocurrida durante la ejecucion de alguno de los casos de uso del sistema.

Package main Description

Este package contiene las clases principales de la aplicacion: TopicSystem (clase principal), Laboratory y WebFace.

La clase TopicSystem corresponde al corazon del sistema, la cual permite ensamblar los diversos modulos de la aplicacion: String2Vector, Vector2Cluster, ClusteringOnline. Ademas, se proveen dos clases que implementan la capa de interaccion con el usuario: el acceso web (metodos que retornan xml) y un programa realizado para ejecutar los experimentos.

Package model

Contiene las clases que modelan los dos principales objetos del sistema: los vectores (documentos) y los clusters (topicos).

See:
Description

Class Summary	
Article	Esta clase representa a un articulo de la base de datos.
ItemVector	Esta clase modela el vector de un articulo del sistema.
Member	Esta clase representa a un miembro de un cluster.
TopicCluster	Esta clase modela un cluster (un topico) del sistema.
Word	Esta clase modela una palabra dentro del espacio de palabras de la aplicacion.

Package model Description

Contiene las clases que modelan los dos principales objetos del sistema: los vectores (documentos) y los clusters (topicos). Tambien posee las clases que modelan articulos, miembros de un cluster, y palabras.

Package modules

En este package se definen las interfaces que modelan los principales modulos de la aplicacion, asi como sus distintas implementaciones.

See:
Description

Interface Summary	
ClusteringOnline	Interfaz para realizar el clustering incremental.
String2Vector	Interfaz para el proceso de creacion del espacio vectorial de palabras.
Vector2Clusters	Interfaz para el proceso de clustering batch.

Class Summary	
CluteVector2Clusters	Implementacion del metodo de clustering batch, usando Cluto.
DummyClusteringOnline	Implementacion -simple- del algoritmo de clustering incremental.
SimpleBufferClusteringOnline	Implementacion de single-pass con buffer, para el clustering incremental.
WVTString2Vector	Implementacion del proceso de creacion del espacio vectorial de palabras, utilizando WordVectorTool.

Package modules Description

En este package se definen las interfaces que modelan los principales modulos de la aplicacion, asi como sus distintas implementaciones. Los modulos del sistema son tres: transformacion de documentos a vectores, generacion batch de los clusters, y clustering online.

Package wvtool

Aqui se definen las clases que extienden la funcionalidad de wvtool, utilizando herencia o implementacion de interfaces.

See:
Description

Class Summary	
DBInputList	Esta clase permite expandir una query en un conjunto de consultas.
DBLoader	Esta clase implementa un Loader para WVTool, que carga una query desde la base de datos.
MyTFIDF	Esta clase permite variar la funcion de calculo del vector de pesos.
MyWVTool	Esta clase extiende la funcionalidad de la clase WVTool.
RSSCharConverter	Un conversor para preparar los caracteres de un texto
RSSInputFilter	Very simple tag ignoring reader.

Package wvtool Description

Aqui se definen las clases que extienden la funcionalidad de wvtool, utilizando herencia o implementacion de interfaces.

En *http://www.danielgomez.cl/memoria/*, se encuentra una versión online de la documentación de este proyecto, incluyendo material que no fue incluido en este documento, listados de algunos de los resultados, y referencias en internet a material complementario.

Apéndice B

Parámetros del Sistema

A continuación se muestra el archivo de propiedades utilizado por la aplicación, que permite apreciar el conjunto de parámetros utilizados. Cada uno de estos valores puede ser modificado, permitiendo variar los resultados, según la calidad de lo obtenido, o según necesidades relacionadas con la implementación.

```
# parametros del sistema
WEIGHTS=pfidf
STEMMING=true
STOPWORDS=true
CHARMAPPER=rss
INPUTFILTER=rss
MINCHARS=3
MIN_PRUNE_BATCH=4
MAX_PRUNE_BATCH=2000
MIN_PRUNE_ONLINE=2
MAX_PRUNE_ONLINE=2000
TAIL_TOPICS=0.1
SIZEBUFFER=3

# parametros de cluto
SIM=cosine
CLMETHOD=direct
CRFUN=i1
CSTYPE=best

# valores PFIDF
IDF=log
SCORE1=2
SCORE2=1
SCORE3=1

# ventanas de tiempo
ALERT_WORDLIST=100
ALERT_PURGE=100
ALERT_GARBAGE=100
WINDOW_PURGE=30
WINDOW_RANKING=5
DELTATBATCH=2007-01-01:2007-02-01
RELATEDS=10

# umbrales clustering
THRESHOLD_SINGLEPASS=0.3
THRESHOLD_GARBAGE=1

# filesytem
HOME=/home/dgomez
TMPDIR=/tmp
CLUTODIR=/home/dgomez/programs/cluto-2.1.1/Linux
WVTOOLDIR=/home/dgomez/programs/wvtool-1.1

# database
DRIVER=com.mysql.jdbc.Driver
URL=jdbc:mysql://localhost/rss
USER=dgomez
PASS=dgomez
```