Gestión Eficiente del Índice Invertido para Flujos de Documentos en Tiempo Real

Trabajo que presenta para aplicar al título de
Licenciado en Sistemas de Información

Esteban Andrés Ríssola

Dirigida por

Mg. Gabriel Hérnan Tolosa

Septiembre 2015

Agradecimientos

...Gracias Totales.
Gustavo Cerati

Quisiera agradecer en primer instancia a mis padres, siendo tantos los motivos pero principalmente por su aliento y apoyo incondicional desde el comienzo de mi carrera. Les agradezco por darme la oportunidad bajo su incansable esfuerzo de poder estudiar aquello que me apasionaba y por incentivarme a perseguir mis metas y sueños sin importar cuánto se tropiece en el camino o cuán largo resulte este. A ellos, simplemente gracias.

También me gustaría agradecer a mi director Gabriel Tolosa, por depositar en mi su confianza y esfuerzo, motivándome cada día a seguir adelante. Le agradezco por su incondicional disposición para brindar una mano (o las dos), siendo no sólo un ejemplo de profesionalismo sino también de carisma.

El último agradecimiento quisiera otorgarlo a aquellos compañeros y amigos con quienes he compartido desde el principio (y no tanto) experiencias no sólo académicas sino también personales, permitiéndome disfrutar de una vida universitaria realmente amena e inolvidable.

Resumen

Uno de los mayores factores que ha favorecido la prominente evolución de la Web reside en la creciente popularidad de los sitios orientados a la generación de contenido por parte de los usuarios (como Facebook, MySpace, Twitter, Flickr, Tumblr, Blogger, entre tantos otros). Si bien cada uno de ellos presenta diversas particularidades, todos procuran brindar la posibilidad de interactuar y compartir el contenido que las personas generan (mensajes, fotos, videos, enlaces, etc.) con quienes conforman su denominada red social. En particular los sitios de microblogging, como Twitter, se constituyen como una nueva forma de comunicación en la cual los usuarios son capaces de reseñar su estado actual en mensajes cortos, y cuya difusión se lleva a cabo mediante mensajes instantáneos, teléfonos móviles, e-mail o la Web en el momento en que fueron generados.

Es en este contexto donde el concepto de búsqueda en tiempo real emerge introduciendo nuevos requisitos y desafíos a los sistemas de búsqueda tradicionales, miles de nuevas actualizaciones deben procesarse inmediatamente luego de su generación, mientras los usuarios esperan que el contenido se encuentre disponible en materia de segundos. La única estrategia plausible que permite alcanzar el rendimiento requerido para satisfacer estas exigencias consiste en mantener el índice invertido y sus estructuras asociadas en memoria principal. Tal situación admite reducir considerablemente los tiempos de accesos, tanto para la lectura como para la escritura, en comparación con otros dispositivos de almacenamiento.

Si bien hoy en día puede considerarse al almacenamiento primario como un recurso escaso (en comparación con aquel secundario), también es cierto que se dispone de equipamiento de altas prestaciones, el cual junto a los avances en arquitecturas de búsqueda distribuida, debilitan aquella suposición que plantea la necesidad de alojar por cuestiones de tamaño al índice invertido en almacenamiento secundario en lugar de hacerlo en memoria. Lamentablemente no siempre es posible acceder a este tipo de hardware, principalmente por su costo de adquisición y mantenimiento.

De esta manera, la necesidad de plantear algoritmos y estructuras de datos eficientes cuyo fundamento se sustente en técnicas y estrategias proporcionadas en la bibliografía conocida resulta realmente imperiosa. Princi-

palmente, esto no sólo permite superar aquellas limitaciones que el hardware subyacente presenta sino también cumplimentar las exigencias que la búsqueda en tiempo real impone con la mayor efectividad y eficiencia alcanzables. Por ello, en el presente trabajo se introduce el concepto de *invalidadores de entradas de índices* (IEI), un enfoque que apunta a invalidar y desalojar selectivamente aquellas entradas del índice invertido cuya ausencia no degrade considerablemente la efectividad en la recuperación. En consecuencia, el índice se vuelve más pequeño y puede aumentar la eficiencia general. Complementariamente, se estudia la dinámica que caracteriza a las colecciones propias del área de microblogging empleando un dataset real aportado por la comunidad científica y se evalúa desempeño de las estrategias propuestas utilizando Zambezi, un motor de búsqueda diseñado para simular ambientes de búsqueda propios de sistemas de recuperación en tiempo real.

Índice

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

I have no special talents. I am only passionately curious.

Albert Einstein

1.1. Evolución de la Web

Durante las últimas décadas hemos atestiguado el explosivo crecimiento que ha protagonizado la Web, ya sea por cuestiones relacionadas al avance tecnológico, facilidad de acceso, intercambios comerciales, aspectos de índole cultural y generacional, entre otras. Si bien resulta realmente complejo (sino imposible) definir con precisión el tamaño de este extenso conglomerado de sitios y contenido distribuido en millones de computadoras, en 2005 Gulli y Signorini [38] estimaban que comprendía aproximadamente 11.5 miles de millones de páginas. Al día de hoy es posible establecer que esta cifra se acerca a los 47 miles de millones[1], siendo de gran importancia destacar que corresponde a aquella fracción accesible pues el resto constituye la denominada *web oculta*[2].

Es claro que la Web evoluciona constantemente, ya sea tanto en tamaño como en diversidad de contenido, mientras nuevas páginas surgen otras dejan de existir. Varios trabajos han profundizado en esta cuestión [30, 25, 60, 17, 32], en particular [63] compete a un estudio realizado en el año 2004, donde se monitorizaron semanalmente por un período de un año 154 sitios Web[3], de los cuales se caracterizó dinamismo y evolución, abocándose principalmente al análisis de contenido y estructura de enlaces. El número total de páginas

[1] World Wide Web Size: `http://www.worldwidewebsize.com`

[2] Web Oculta [45] o Web Profunda[15]: Corresponde a aquel fragmento de la *World Wide Web* que no puede ser accedido y, en consecuencia, indexado por los motores de búsqueda convencionales.

[3] Sitio Web: Conjunto de páginas relacionadas, comúnmente servidas desde un único dominio Web (*e.g.*, acm.org, hp.com, unlu.edu.ar).

descargadas cada semana osciló entre los 3 y 5 millones, con un promedio de 4.4 millones, pudiendo estimar que en este lapso se crean nuevas páginas a una tasa del 8 % (alrededor de 320 millones)[4]. Por otra parte, fue posible establecer que una vez concebida cada página resultan realmente menores (sino nulos) los cambios que esta atraviesa durante su tiempo de vida.

1.2. Indexación y Búsqueda

Estas cifras permiten definir un panorama aproximado, aunque no menos certero, del volumen de datos con los que debe lidiar un Sistema de Recuperación de Información (SRI). Tales sistemas, se enfocan principalmente en simplificar a los usuarios el acceso a la información de su interés [14], mediante la representación, almacenamiento y organización de diversos objetos (tales como páginas Web), persiguiendo la mayor eficiencia y efectividad en la ejecución dicha labor. Los usuarios presentan *necesidades de información*, estas resultan ser aquellas temáticas acerca de las cuales el individuo desea saber más y en consecuencia, se caracterizan por ser arbitrarias y de variada complejidad. En los casos más simples comprenden la búsqueda del sitio de una institución o compañía, mientras que en aquellos más complejos pueden abarcar cuestiones más refinadas, como la obtención de cierta información necesaria para concretar una determinada labor (*e.g.*, tarea escolar, resoluciones judiciales de un caso, etc.).

Asimismo, compete al usuario traducir esta necesidad de información en una *consulta* (o sucesión de ellas), mediante el empleo de un conjunto de palabras clave que permitan resumirla de manera tal que pueda ser presentada posteriormente al SRI; siendo la meta de estos recorrer largas colecciones de documentos[5] con el fin de encontrar aquellas piezas de información (por naturaleza carentes de una estructura definida [54]) *relevantes* para el usuario. En otras palabras, que permitan aproximar con la mayor precisión posible la respuesta a la necesidad planteada. Este hecho presenta un reto no sólo en la manera en la que debe extraerse la información de los documentos, sino también en la definición de cómo emplearla para establecer una medida de relevancia. En este sentido ningún SRI resulta capaz aún de proporcionar una respuesta "ideal" en todos los casos, pues en primera instancia la relevancia (intrínsecamente subjetiva [67, 14]) se constituye como una valoración personal subordinada tanto a la actividad a resolver como al contexto en que esta se desarrolla y por otro lado, continua siendo un problema compatibilizar la expresión de la necesidad de información entre el lenguaje y los documentos

[4] De acuerdo a estimaciones del tamaño de la Web correspondientes al período estudiado.

[5] El concepto de *documento* hace alusión a la unidad sobre la cual se ha decido construir el sistema de recuperación [54]. Convirtiéndose en consecuencia, en aquel objeto que será retornado como respuesta a una búsqueda.

[76]. A pesar de ello, es posible cuantificar la calidad de los resultados de una búsqueda, tal evaluación se conoce como *efectividad* y comprende el cómputo de dos variables: (1) *precisión*, fracción de los resultados retornados que resultan relevantes a la necesidad de información, y (2) *exhaustividad*, fracción de los documentos relevantes (del conjunto total de relevantes) recuperados por el sistema.

Es claro que para dar respuesta a las consultas que los usuarios remiten al SRI, se torna imperioso representar y organizar de alguna manera la información contenida en el conjunto de documentos, denominado de aquí en adelante como *corpus* o *colección*[6]. Cada elemento de este cuerpo se construye a partir de sucesiones de palabras que forman estructuras gramaticales (*e.g.*, oraciones y párrafos), escritos en lenguaje natural. En el caso particular de una página Web se adicionan otros elementos tales como etiquetas de marcado[7], enlaces, imágenes y otros objetos multimedia, etc. Con el objeto de llevar a cabo operaciones sobre una colección es imprescindible en primera instancia alcanzar una representación lógica de todos sus componentes, la cual puede conformarse de un conjunto de términos, frases u otras unidades (sintácticas o semánticas) que admitan de alguna forma caracterizarlos [76] (*e.g.*, bolsa de palabras[8]).

Una vez obtenida dicha representación, el próximo paso compete a la construcción de aquellas estructuras de datos sobre las que esta se almacenará y que, posteriormente, permitirán dar soporte a la recuperación de forma eficiente. Se conoce a tal estructura con el nombre de *índice invertido*[9], y se conforma de todas aquellas palabras distintas que componen la colección (*i.e.*, el *vocabulario* o *lexicon*), donde para cada término se mantiene una lista de los documentos que lo contienen. Cada elemento de esta lista que registra si una palabra forma parte del contenido de un documento, además de otras estadísticas como la frecuencia y correspondientes posiciones dentro del texto, se denomina de manera convencional *posting*. Por consiguiente, el conjunto completo recibe el nombre de *lista de posteo* (o *lista invertida*)[10] y se conoce como *indexación* al proceso involucrado en la construcción de este índice . Asimismo, se preservan otros datos de utilidad, tales como el número de documentos que aloja cada término (*i.e.*, la longitud de la lista de posteo), que permiten mejorar la eficiencia del sistema al momento de procesar una consulta consulta [54].

Como se mencionó anteriormente la necesidad de un usuario puede ser compleja de expresar, siendo que la primer dificultad se presenta en construir un enunciado de consulta que la refleje con exactitud. En innumerables ocasiones el SRI debe lidiar con situaciones subjetivas, ambiguas (como la

[6] Otras acepciones posibles son: base de datos textual o documental.
[7] HTML: Hyper Text Markup Language.
[8] Conjunto de palabras a las que se asocia su frecuencia en los documentos.
[9] Denominado en otros contexto como *archivo invertido*.
[10] El conjunto de listas invertidas recibe el nombre de *postings*.

polisemia) e incompletas y por ello, tanto los documentos como las consultas se interpretan de modo tal que el proceso de recuperación defina un grado de *similitud* entre éstos. El resultado final es la obtención de una lista ordenada de documentos, conocida como *ranking* (o *top-k* si se limita a una cantidad específica), confeccionada de acuerdo a la probabilidad de que cada uno resulte relevante para el usuario [14]. En este sentido, el conjunto de respuesta no es exacto, pues pueden retornarse documentos que no sean relevantes a la consulta en cuestión [76].

1.2.1. Poda (Pruning)

Uno de los mayores impactos que presenta la Web en la búsqueda y recuperación aquí descriptas, subyace tanto en el tamaño de la colección tratada como en el volumen de usuarios que remiten consultas diariamente. Dado que esta ha crecido más que cualquier otra colección, los SRI conocidos en este contexto como *Motores de Búsqueda Web* (MBW) de propósitos generales, deben procesar e indexar volúmenes de texto sin precedentes[11], con el objetivo de resolver un amplio número de consultas en un tiempo de respuesta interactivo. En consecuencia, es correcto afirmar que enfrentan desafíos realmente formidables en términos de rendimiento y escalabilidad, ya que no sólo resulta importante considerar la calidad de los resultados de la búsqueda (*efectividad*), sino también la velocidad con la cual los mismos son generados (*eficiencia*) [79].

Es por esta cuestión que existen diversas técnicas destinadas a gestionar el índice invertido, entre las más comunes se cuenta con algoritmos de compresión los cuales se orientan a organizar de forma más eficiente la información almacenada, de modo que las listas de posteo cuenten con una representación más compacta, por ejemplo PForDelta [85]. Complementariamente, las estrategias de *poda (pruning)* intentan encontrar la manera de descartar aquella información cuya ausencia no represente una disminución que sea percibida por el usuario en el rendimiento general del sistema. Asimismo, contribuyen a alcanzar los niveles de rendimiento deseados, ya que proporcionan para ello métodos orientados a enfrentar uno de los mayores cuellos de botella en el procesamiento de consultas, *i.e.*, la longitud de la estructura encargada de almacenar las listas de posteo, la cual puede crecer fácilmente a miles de MBs o incluso GBs para términos de búsqueda comunes (aproximadamente lineal con el tamaño de la colección) [28, 71].

Mientras los enfoques de pruning *estáticos* procuran reducir el espacio requerido para el almacenamiento del índice invertido, suprimiendo aquellas entradas cuya potencial contribución al puntaje de un documento es tan pequeña que su ausencia tendrá un efecto despreciable sobre la efectividad

[11] Un ejemplo concreto, es el caso del motor de búsqueda comercial *Yahoo!*, donde se reporta contar con un índice de 100 mil millones de documentos [22].

global; los *dinámicos* (también conocidos como procesamiento optimizado de los top-k o *early termination*) se orientan a evitar el cálculo del puntaje de aquellos documentos que no lograrán formar parte del ranking final, disminuyendo de esta manera el tiempo empleado en el procesamiento de consultas, *e.g.*, el bien conocido algoritmo WAND [19]. Resulta evidente que la aplicación de estas estrategias da lugar a un compromiso (*tradeoff*) natural entre el tiempo de procesamiento de consultas y la precisión del sistema, que debe abordarse bajo la adopción de ciertos principios, *i.e.*, encontrar el equilibrio adecuado en el cual el incremento del primero no supere un cierto umbral de tolerancia fijado para el segundo.

1.2.2. Caching

Otro mecanismo empleado para lidiar de forma eficiente con el extenso volumen de consultas que los MB deben procesar, esta vez en un nivel superior al índice invertido, es la utilización de varios niveles *caché*. La razón bajo la cual subyace el éxito de esta estrategia radica en aprovechar aquellos datos frecuente o recientemente servidos, evitando volver a ejecutar el proceso que permitió obtenerlos, a la vez que se logra aminorar la carga de trabajo sobre aquel componente encargado de computar dicho proceso [13]. Esta optimización posibilita enmascarar la latencia de operaciones costosas y puede aplicarse en diferentes niveles donde los tiempos de respuesta y/o requisitos de procesamiento resultan diversos (*e.g.*, memoria principal o disco duro, recursos locales o remotos, etc.).

Se considera un acierto (*hit*) cuando se da respuesta una solicitud haciendo uso de los resultados almacenados en el caché; en el caso contrario, se define un fallo (*miss*), hecho que implica computar los correspondientes resultados y, comúnmente, incorporarlos al caché. Cuando este último se encuentra completo resulta necesario desalojar (*evict*) ciertas entradas, tal decisión se lleva a cabo de acuerdo a una política de reemplazo (*e.g.*, LRU, FBR [55], LFU [13]), cuyo principal objetivo es maximizar la tasa de aciertos[12] [16]. Complementariamente, el contenido del caché puede construirse de forma estática (*off-line*), basado en información histórica y de actualización periódica, ó dinámica (*online*), donde las correspondientes entradas se reemplazan conforme a una secuencia de solicitudes.

En el caso particular de los MB tal estrategia puede aplicarse a nivel de (1) *resultados*, cuyo objetivo consiste en incorporar aquel conjunto de documentos relevantes recuperados tras procesar una consulta, o (2) *listas de posteo*, caso en el cual se almacenan secciones de las listas invertidas de los términos involucrados en la consulta. En la práctica, una entrada típica de un cache de resultados se compone de una cadena de caracteres y ciertos datos, como URLs o snippets[13], acerca de los correspondientes docu-

[12] Fracción de las solicitudes que servidas desde el caché.

[13] Breve fragmento de texto que sintetiza el contenido de una URL retornada como

mentos (*top-k*) que conforman el conjunto solución, y ocasionalmente, otra información auxiliar (*e.g.*, marcas de tiempo). Dada la importante cuantía de consultas que los usuarios remiten diariamente, el caché representa un componente crucial en el rendimiento general, ya que permite aminorar el tráfico de solicitudes a ser procesadas, como así también contribuye a reducir el tiempo de procesamiento promedio. En gran medida esto se debe a que la distribución de frecuencias de las consultas se ajusta a una *ley de potencias* [84, 22, 1], suceso que implica a que unas pocas poseen una frecuencia muy alta y muchas se remiten un número reducido de veces, a menudo sólo una vez[14]. Como consecuencia directa de este fenómeno los cachés de resultados pueden alcanzar en la práctica elevadas tasas de aciertos [22].

Es importante destacar que a diferencia de su aplicación en otros contextos, los cachés de resultados no presentan limitaciones de memoria puesto que es posible almacenar millones de entradas en disco y aún así, mejorar las prestaciones del sistema. Sin embargo, uno de los mayores retos que este mecanismo enfrenta es mantener la denominada *coherencia del cache* [22, 1]. Es decir, el hecho que la Web resulte ser una colección tan dinámica implica que los índices subyacentes se encuentran constantemente sujetos a numerosas modificaciones, por consiguiente existe una vasta probabilidad que una fracción significativa de los resultados previamente computados, y almacenados se vuelvan obsoletos con el tiempo. Adicionalmente, el problema de la *frescura* se torna aún más severo en la medida en que la capacidad del cache se incrementa, por lo que una de las técnicas más simples para lidiar con este problema consiste en asociar a cada entrada un *tiempo-de-vida* (TTL), cuya expiración indica la necesidad de invalidar la entrada en cuestión y eventualmente desalojarla o actualizarla.

1.3. Microblogging

Uno de los mayores factores que ha favorecido la prominente evolución de la Web reside en la creciente popularidad de los sitios orientados a la generación de contenido por parte de los usuarios [40, 46] (como Facebook, MySpace, Twitter, Flickr, Tumblr, Blogger, entre tantos otros), al punto de ocupar cinco de las quince primeras posiciones de las sitios más visitados de la Web en 2007 [44] y donde se registran millones de usuarios[15]. Si bien cada uno de ellos presenta diversas particularidades, todos procuran brindar la posibilidad de interactuar y compartir el contenido que las personas generan (mensajes, fotos, videos, enlaces, etc.) con quienes conforman su denominada

resultado de una búsqueda.

[14] Denominadas *singleton*.

[15] Por ejemplo, Facebook detalla entre sus estadísticas contar en promedio con 968 millones usuarios activos (Facebook About: `https://newsroom.fb.com/company-info/`).

red social[16]. Frecuentemente, tales redes se componen de cientos e incluso miles (en caso de las celebridades) de enlaces [82].

Para el presente trabajo resultan de particular interés los denominados sitios de *microblogging*. Estos se constituyen como una nueva forma de comunicación en la cual los usuarios son capaces de reseñar su estado actual en mensajes cortos[17] (denominados *posts*), y cuya difusión se lleva a cabo mediante mensajes instantáneos, teléfonos móviles, e-mail o la Web en el momento en que fueron generados [43]. Por lo general, un microblog es administrado por un único individuo o entidad; comprende publicaciones periódicas y breves (comúnmente denominadas *actualizaciones de estado*) que de forma colectiva presentan a los lectores (llamados *seguidores*) información de carácter temporal, cuyo autor ha considerado interesante (aseveración de índole subjetiva [31]). Cada usuario declara quienes serán las personas que le interesa *seguir*, caso en el cual es notificado cada vez que dicha persona publique un nuevo mensaje. Una cuestión importante que merece aclaración es que si un usuario es *seguido* por otro, no existe reciprocidad implícita en tal relación, hecho que genera que los enlaces de la red social sean dirigidos [40]. Mientras varios microblogs resultan de naturaleza personal y solamente de interés para conocidos del autor; otros poseen una audiencia de mayor amplitud, ya sea porque pertenecen a celebridades o debido a que ofrecen datos de actualidad (*e.g.*, noticias, instituciones). A pesar de compartir ciertas características con los servicios de *blogging* regulares, como la presentación de las entradas en orden cronológico inverso[18], esta nueva actividad se diferencia de la última en aspectos realmente marcados. En primera instancia, complementa la necesidad de las personas de disponer de una manera aún más rápida de comunicarse, ya que al fomentar la escritura de mensajes cortos reducen los requisitos de tiempo y elaboración que limitan a los usuarios al generar el contenido en cuestión. Asimismo, divergen en la frecuencia de las actualizaciones puesto que en promedio un autor prolífico de un blog renueva sus entradas en períodos que abarcan días; mientras que un microblogger es capaz de publicar gran cantidad de posts en un mismo día.

Uno de los casos paradigmáticos ha sido Twitter[19] quien, además de superar a sus competidores (como Jaiku[20] y Pownce[21]), actualmente cuenta con 302 millones[22] de usuarios activos y se presenta, desde su aparición en octubre de 2006, como uno de los actores principales de esta escena. En su jerga cotidiana cada post recibe el nombre de *tweet* y se limita a 140 caracteres. Si bien las actualizaciones de estado son breves, frecuentemente ofrecen enlaces

[16] También conocida como *red de conocidos*.
[17] Por lo general menos de 200 caracteres.
[18] De aquella de mayor actualidad a la más antigua.
[19] Página Principal: `https://twitter.com/`
[20] Jaiku: `https://en.wikipedia.org/wiki/Jaiku`
[21] Pownce: `https://en.wikipedia.org/wiki/Pownce`
[22] Twitter About: `https://about.twitter.com/company`

a material externo como páginas web, imágenes o cualquier otro contenido disponible en línea. Aunque las razones en las cuales radica la popularidad de esta plataforma resulten complejas de precisar, es claro que se ha convertido en el centro de varios estudios con diversos enfoques [43, 46, 41, 86, 47], en gran medida a causa de la abundante cantidad de usuarios y principalmente gracias a la API[23], que permite a los investigadores descargar datos desde Twitter (*i.e.*, tweets, perfiles de usuarios, relaciones[24], entre otros). Algunos trabajos se avocan al estudio de las razones que motivan a las personas a emplear este servicio, como el caso de Java et. al [43], quienes determinan que entre las principales intenciones se encuentran actualizaciones autobiográficas, conversación entre pares, intercambio de información y reporte de noticias de actualidad. Complementariamente determinaron la existencia de comunidades, cuyos miembros presentan intereses congruentes sin encontrarse limitados por las fronteras geográficas (región, país, continente) y donde sus intenciones no sólo radican en el tópico compartido sino también en experiencias personales, charla diaria, etc. Asimismo, no resulta menor destacar que el exponente de la ley de potencias encontrado en este estudio para los grafos sociales en Twitter fue aproximadamente -2.4, muy similar a aquellos parámetros hallados para la Web [29]. Estas cuestiones permitieron establecer la presencia de un alto grado de correlación y reciprocidad en la red. En contraposición Kwak et al. [47] aseveran que de acuerdo al análisis concretado sobre la topología seguidor-siguiendo, la distribución de seguidores no se ajusta a una ley de potencias, además de observar baja reciprocidad entre los usuarios, especialmente en el caso de las celebridades y los medios masivos de comunicación. Este aspecto también ha sido abordado en [40] donde se analizaron las relaciones de reciprocidad[25] entre los usuarios y cómo estas se relacionan con la generación de posts. Entre las conclusiones obtenidas se determina la existencia de dos redes diferentes, (a) una muy densa constituida por los seguidores y a quienes sigue un usuario, y (b) otra más dispersa y simple conformada de aquellas personas con las cuales el usuario mantiene contacto (*i.e.*, intercambio directo de mensajes[26]). Este hecho permite plantear que una conexión entre dos personas no necesariamente implica una interrelación entre ellos, y por lo tanto gran cantidad de los enlaces declarados resultan sin sentido desde el punto de vista de la interacción.

[23] API ó *interfaz de programación de aplicaciones* constituye un conjunto de subrutinas, funciones y procedimientos que ofrece una determinada biblioteca para ser utilizado por otro software como una capa de abstracción.

[24] *Seguidos* y *seguidores*.

[25] A pesar de que la reciprocidad no resulta obligatoria se mantiene en promedio en un 90 por ciento en lo que los autores definen como *amistad*.

[26] En el contexto de Twitter esta acción recibe el nombre de *mención*.

1.3.1. Búsqueda en Tiempo Real

Es claro que la búsqueda y recuperación sobre este tipo de colecciones, como así también la gestión de las estructuras encargadas de ello, presentan ciertas diferencias e introducen nuevos requisitos y desafíos en comparación con aquella desarrollada en el ámbito de la Web [20, 11]. A considerar:

Procesamiento de consultas de baja latencia y alto *throughtput*[27]. Resulta imperioso en este contexto resolver (ó evaluar) la mayor cantidad de consultas posibles, incurriendo en el menor tiempo admisible para ello con el fin de garantizar a los usuarios una experiencia satisfactoria.

Alta tasa de ingestión y disponibilidad inmediata. Los usuarios esperan que el contenido se encuentre disponible (*searchable*) dentro de un período reducido de tiempo (en el orden de unos pocos segundos), por ende aquel componente encargado de mantener y gestionar el índice invertido debe estar diseñado para recibir un constante flujo de documentos (con tasas de arribo elevadas e incluso ráfagas repentinas), buscando alcanzar tanto baja latencia como alto throughput en el cumplimiento de su cometido. A diferencia de los enfoques "tradicionales" (*e.g.*, Web), donde dicho proceso se considera una operación por lotes (*batch*); en tales casos se admite que el contenido esté disponible en minutos, horas o incluso días.

Lecturas y escrituras concurrentes. Las estructuras del índice invertido deben actualizarse continuamente en la medida en que ingresan nuevos documentos (indexación incremental), al mismo tiempo que son accedidas para resolver las consultas entrantes. En contraposición a la búsqueda Web donde el índice invertido resulta principalmente de sólo lectura.

Predominio del factor temporal. La marca de tiempo de un documento (*timestamp*) es uno de las factores más importantes para ordenar los resultados, más allá de la implementación de otras métricas de relevancia orientadas a mejorar el ranking. Contrariamente a lo que sucede en la búsqueda web, donde la marca de tiempo de una página posee un rol relativamente menor en la relevancia que aportará a la lista final de resultados.

Complementariamente otros autores como Efron [31] plantean que en varias ocasiones la unidad[28] de recuperación no resulta tan evidente o necesita al menos un cierto grado de formalización para permitir el desarrollo de una operación exitosa. Si bien dado un corpus y un conjunto de términos de búsqueda retornar posts individuales puede resultar útil, presentar al usuario

[27] Cantidad de datos procesados por unidad de tiempo.
[28] Puede ser más de una, *e.g.*, persona, post, etc.

una lista de publicaciones fuera de contexto no constituye la mejor manera satisfacer su necesidad de información o al menos la única. De igual manera, no debe obviarse el alto grado de subjetividad que los documentos presentan, ya que una innumerable cantidad expresan opiniones. Por último, aunque no menos trascendente, es el papel que desempeñan el *tiempo* y el *lugar*; los usuarios publican mensajes en instantes y localizaciones particulares siendo que esta información afecta profundamente la noción de relevancia. En primera instancia esto se debe a que gran cantidad de posts corresponden a tópicos con un horizonte de tiempo limitado, hecho que de alguna manera afecta la *frescura* de los resultados servidos; siendo de prominente importancia la inmediatez (*immediacy* ó *recency*) [57]. Por otro lado, los metadatos geográficos (*i.e.*, latitud y longitud) de una publicación también resultan ser una fuente de utilidad para producir respuestas con mayor efectividad.

Varios trabajos han profundizado en estas últimas cuestiones como [34, 50, 26], particularmente Nepomnyachiy et al. [61] desarrollan una arquitectura de búsqueda orientada a la resolución eficiente de consultas conformadas por componentes espaciales, temporales y textuales. Con el fin de proporcionar el soporte necesario para este tipo de recuperación, se emplean múltiples fragmentos de índices dispuestos en un árbol (*R-Tree*) de poca profundidad, basándose en la distribución espacial de los datos geo-etiquetados. En particular cada nodo representa un área geográfica específica, donde todos aquellos datos pertenecientes a dicha extensión se almacenan dentro o debajo del mismo. Los nodo hoja contienen un índice invertido (independiente) para todos los documentos encontrados dentro los límites espaciales definidos. La geo-localización junto con las marcas de tiempo presentes en los documentos permiten expandir la consulta, de forma tal que sólo aquellas listas invertidas que se encuentren dentro de las cotas especificadas se emplearán para conformar el ranking final.

En [58] Metzler et al. plantean que a diferencia de los blogs, en el ámbito de los microblogs no existe una *tarea* de búsqueda bien definida, siendo que los sistemas actuales solamente proveen la habilidad de recuperar posts individuales en respuesta a una consulta. Aunque esta constituye una utilidad limitada ya que muy pocas necesidades de información pueden ser satisfechas por una reducida pieza de texto (*i.e.*, tweet) y por ello, proponen la "recuperación estructurada de eventos". Dada una consulta que describe un evento en particular (*e.g.*, un terremoto) el objetivo consiste en recuperar una lista ordenada (ranking) de representaciones estructuradas del mismo. Estas se constituyen como una enumeración de intervalos de tiempo (*timespans*) durante los cuales ha ocurrido una instancia del evento en cuestión, que a la vez ha sido activamente discutida dentro del flujo producido por el sistema de microblogging. Complementariamente, para cada uno de estos lapsos son recuperados un reducido conjunto de mensajes relevantes, cuyo objetivo es proporcionar al usuario un resumen del evento ocurrido en el

intervalo de tiempo tratado. Con el fin de reducir el impacto de las diversas dificultades que presenta la recuperación sobre este tipo de sistemas (como la escasa información en las consultas ó la utilización de jergas) se presenta una técnica de expansión no supervisada de la consulta, en la cual los términos de expansión se generan de acuerdo a su co-ocurrencia temporal. La consulta resultante es empleada para seleccionar los intervalos de tiempo que posteriormente serán recuperados.

Asimismo, en [36] se define un esquema de modelado de tópicos orientado a la resolución de consultas sobre corpus conformados por tweets (o documentos afines al área de microblogging). El modelado de tópicos permite representar los documentos como bolsas de palabras sin considerar el orden de las mismas como un parámetro relevante, empleándose para "descubrir" temas (o tópicos) dentro de la colección tratada y poder así, compararlos contra aquellos estimados de las consultas entrantes. Tal estimación es utilizada posteriormente para conformar el ranking de aquellos documentos considerados como relevantes a la consulta. Con el fin de incrementar la eficiencia del proceso de selección, los tweets son aglomerados con aquellos caracterizados como "similares" en un proceso denominado generación de supertweets. Este proceso implica la definición de varios criterios de similitud, basados en el contenido textual de cada tweet (como los hipervínculos). Debido a que el número de posibles tópicos o conceptos resulta prácticamente ilimitado en el caso de las conversaciones provenientes de tweets, además de la volatilidad de los mismos con el paso del tiempo, es necesario algún tipo modelo a ser aplicado fuera de línea para reducir el rango posible.

Por otro lado, Chen et al. [24] implementan un esquema de indexación adaptativo orientado a sistemas de microblogging, en el cual el principal objetivo subyace en procesar e incorporar al índice aquellos posts que con gran probabilidad formarán parte de los resultados de búsqueda ("distinguidos") y retrasar la indexación del conjunto restante ("ruidosos"). Así, a partir del análisis del contenido de cada documento se determina su tipo y se toman las acciones correspondientes de acuerdo al caso. Tal estrategia permite reducir significativamente los costos computacionales asociados al proceso de indexación sin comprometer la efectividad del SRI. Complementariamente los autores plantean un nuevo esquema para la generación del ranking, basado en la relación existente entre usuarios y posts (*PageRank* del usuario [80, 83], popularidad de los tópicos, frecuencia de los términos y marcas de tiempo).

1.3.2. Consultas

Una cuestión importante que asimismo ha de ser considerada para satisfacer las exigencias que presenta la recuperación en este contexto, es explorar y caracterizar el comportamiento de búsqueda; en otras palabras comprender qué tipos de consultas remiten los usuarios y qué información desean

obtener mediante su resolución. En el ámbito de la Web existen tres clases de consultas, a saber [18]: (a) *navigacionales*, el propósito consiste en alcanzar un sitio en particular [29], (b) *informacionales*, la intención radica en adquirir o encontrar información la cual se asume que se encuentra presente en una o más páginas Web (estáticas, sin interacción adicional), y finalmente (c) *transaccionales*, cuyo objetivo es llegar a un sitio en el cual se concretará una interacción posterior[30].

Teevan et al. [75] han desarrollado una revisión sistemática del comportamiento de búsqueda en Twitter y qué lo diferencia de la búsqueda tradicional sobre la Web, con el fin de establecer cómo y por qué las personas buscan contenido generado en microblogs. Para ello, comparan aspectos de las consultas remitidas a Twitter con aquellas dirigidas a un motor de búsqueda Web tradicional y analizan de qué forma el mismo usuario emplea ambos medios para buscar contenido similar. Principalmente la búsqueda se utiliza para monitorizar contenido, a diferencia de la Web donde el objetivo subyace en aprender y desarrollar un tema en particular. Complementariamente, las personas emplean Twitter para encontrar información temporal relevante como así también aquella relacionada a otros usuarios. Por otro lado, las búsquedas implican mayor contenido social e información de eventos, mientras que en la Web los resultados abarcan hechos más básicos y de contenido esencialmente navigacional. También resulta importante destacar la diferencia en la cantidad de palabras empleadas en las consultas a cada entidad (1.64 para el caso de Twitter y 3.08 para la Web, si bien son más cortas contienen palabras de mayor longitud) de la misma forma que su contenido, resultando de especial atención el uso de los operadores "@"[31] y "#"[32] los cuales poseen un significado específico (*metadatos generados por los usuarios* [31]) y son regularmente empleados tanto por creadores como consumidores de contenido. Finalmente, otro aspecto en el que difieren concierne al texto y la metodología para presentar los resultados de búsqueda al usuario; mientras que en Twitter el contenido completo de cada resultado es presentado al usuario en forma de lista, en el caso de la Web el ranking final se compone de una lista de hipervínculos acompañados de un pequeño de fragmento de texto (*snippet*) extraído algoritmicamente, orientado a asistir al usuario en la selección del enlace a visitar y que incluso en algunos casos puede satisfacer la necesidad de información. Adicionalmente, este hecho permite advertir una divergencia tanto en el volumen de información disponible al procesar

[29] *e.g.*, UNLu. Objetivo probable: `http://www.unlu.edu.ar`

[30] Principales categorías: shopping, descarga de archivos, servicios web varios, acceso a ciertas bases de datos, búsqueda de servidores, etc.

[31] Empleado para hacer referencia al alias de un usuario. Aquellos términos en un tweet precedidos por el símbolo "@" representan un enlace al perfil del usuario referenciado.

[32] Los términos en un tweet precedidos por el símbolo "#" representan hipervínculos. Haciendo clic en ellos se desencadena una búsqueda de aquellos posts que contengan la etiqueta referenciada.

un consulta [33] como en los patrones del lenguaje empleados.

Otras estadísticas que merecen atención son aquellas obtenidas en [42] donde se reporta que el 30 % de las consultas del set de datos analizado resultan *únicas* contra un 59 % en para caso de la Web. Si bien existen ciertas diferencias con aquellas presentadas en el estudio anterior (23.19 % para Twitter versus 49.73 % para la Web), se trata de dos datasets recuperados de sitios diferentes. Accesoriamente, se caracteriza la distribución de longitud de las consultas donde más de un 44 % contenían un sólo término, un 30 % dos y cerca de un 26 % tres o más, resultando la longitud promedio 2.32 términos.

1.4. Motivación

De esta manera, considerando el modo en que los servicios de microblogging han evolucionado y por consiguiente el volumen de datos[34] que es necesario gestionar diariamente, la única estrategia plausible que permite alcanzar el rendimiento requerido para satisfacer las exigencias previamente expuestas consiste en mantener el índice invertido y sus estructuras asociadas en memoria principal. Esto admite reducir considerablemente los tiempos de accesos, tanto para la lectura como para la escritura, en comparación con otros dispositivos de almacenamiento. Si bien hoy en día puede considerarse al almacenamiento primario como un recurso escaso [11] (en comparación con el almacenamiento secundario), también es cierto que se dispone de equipamiento de altas prestaciones, el cual junto a los avances en arquitecturas de búsqueda distribuida, debilitan aquella suposición que plantea la necesidad de alojar el índice invertido en almacenamiento secundario en lugar de hacerlo en memoria [7], no siempre es posible acceder a este tipo de hardware, principalmente por su costo. Se torna indispensable encontrar el modo de mantener en las listas invertidas solamente aquella información que permita alcanzar prestaciones de efectividad razonables (o aceptables). Incluso se ha planteado en la bibliografía que la distribución de los términos (tanto en los documentos como en las consultas) cambian rápidamente en intervalos de tiempo acotados (*e.g.*, una hora), fenómeno conocido como *churn* [52], hecho por el cual los términos más frecuentes de un instante pueden resultar diferentes de aquellos en el próximo. Asimismo, la naturaleza casual de este tipo de plataformas junto con su limitación en el número de caracteres admitidos producen textos en los cuales las palabras raramente se repiten dentro de un documento [61], además del efecto que esta restricción

[33] El número promedio de palabras por consulta en los resultados de Twitter fue 19.55, versus 33.95 en los snippets Web.

[34] De acuerdo a cifras reportadas se publican 500M de Tweets por día (Twitter About: `https://about.twitter.com/company`), siendo que estimaciones aseguran que por segundo se generan en promedio 6K (Internet Live Stats: `http://www.internetlivestats.com/twitter-statistics/`).

presenta en el contenido en sí mismo (*i.e.*, abreviaciones, palabras alargadas o incorrectamente escritas).

Es por ello que sería razonable contar con un componente que monitoriza la manera en que evoluciona el vocabulario de esta clase de colecciones, con el fin de selectivamente *invalidar* y *desalojar*[35] aquellas entradas del índice invertido cuya ausencia no afectará en una medida perceptible la efectividad en la recuperación. Esto permite disminuir la longitud del índice y en consecuencia, puede conducir a mantener esta dimensión constante en el tiempo. Es claro que resulta posible ajustar cuán agresiva puede plantearse tal poda, aunque será necesario encontrar el correspondiente compromiso entre eficiencia (en términos de almacenamiento) y efectividad. Asimismo, para dotar a este componente de tal capacidad es requerido el análisis del vocabulario con el propósito de reconocer la dinámica que lo caracteriza y así identificar aquellas entradas del índice que pueden ser podadas.

1.5. Objetivos

Dada la creciente trascendencia y popularidad que albergan actualmente los servicios de microblogging, como Twitter, nuevos desafíos y requisitos propios de su contexto deben abordarse al intentar dar soporte a la búsqueda y recuperación. Por tal motivo, resulta imperiosa la necesidad de plantear algoritmos eficientes cuyo fundamento se sustente en técnicas y estrategias proporcionadas en la bibliografía conocida. Principalmente, esto permite no sólo superar aquellas limitaciones que el hardware subyacente presenta sino también cumplimentar las exigencias citadas con la mayor efectividad y eficiencia alcanzables.

Es por esta razón que el objetivo principal del presente trabajo radica en proponer y diseñar algoritmos eficientes que permitan la gestión y mantenimiento del índice invertido, de forma tal que sea posible acelerar el proceso de resolución de consultas sobre un corpus de flujos de documentos (*stream documents*) en tiempo real sin degradar la efectividad. Encontrándose entre los objetivos particulares:

- Modelar el comportamiento y la dinámica del vocabulario de colecciones de flujos de documentos en tiempo real.
- Proponer e implementar un conjunto de invalidadores capaces de permitir una gestión eficiente del índice invertido sobre la base del estudio de la dinámica del vocabulario obtenido.
- Evaluar el desempeño de los invalidadores propuestos mediante la aplicación de algoritmos clásicos de resolución de consultas, como WAND

[35] En el área del Pruning Estática se conoce a esta política como "Poda de Palabras Clave" (*Keywords Pruning*) [62]; consiste en podar del índice invertido de forma "horizontal" mediante la eliminación de entradas completas (Término y Lista de Posteo).

[19] cuantificando la efectividad y eficiencia obtenidas.

1.6. Aportes

Con el desarrollo del presente trabajo se espera contribuir al estudio de la dinámica que caracteriza a las colecciones propias del área de microblogging, planteando asimismo, un conjunto de *invalidadores* para índices invertidos orientados a la búsqueda y recuperación en tiempo real basados en conceptos provenientes del área de pruning y caching. Adicionalmente la evaluación de las técnicas propuestas se realiza empleando un dataset real aportado por la comunidad científica [56] y utilizando Zambezi [7], un motor de búsqueda diseñado para simular ambientes de búsqueda propios de sistemas de recuperación en tiempo real.

1.7. Organización del Trabajo

A continuación se describe la secuencia de capítulos, así como una breve descripción de su alcance:

Capítulo 2: Recuperación de la Información. En este capítulo se introducen aquellos conceptos básicos concernientes a la Recuperación de la Información (RI), junto con algunos de los problemas abordados en el área. Se comienza con un resumen de la literatura académica acerca de los componentes involucrados en el proceso de búsqueda y recuperación, como así también aquellos tópicos relacionados a la evaluación dicha tarea. Asimismo, se enfatiza el tratamiento de flujos de documentos (*streaming documents*), considerando los requisitos necesarios para dar soporte a servicios de microblogging.

Capítulo 3: Invalidadores de Entradas de Índices Invertidos. En este capítulo se emprende el análisis sobre el vocabulario obtenido tras procesar la colección Tweets2011, con el fin de caracterizar su dinámica. La segunda mitad corresponde a la definición de los invalidadores de entradas de índices propiamente dichos.

Capítulo 4: Experimentos y Resultados. En este capítulo se presentan en primera instancia la metodología empleada con el fin de evaluar el desempeño de los invalidadores propuesto. Posteriormente, los resultados obtenidos y su correspondiente análisis son desarrollados.

Capítulo 5: Conclusiones y Trabajos Futuros. En este capítulo se introducen las conclusiones obtenidas tras la elaboración del presente trabajo

y se definen aquellas líneas futuras.

Capítulo 2

Recuperación de la Información

Mathematical reasoning may be
regarded rather schematically
as the exercise of a combination of
two facilities, which we may call
intuition and ingenuity.

Alan Turing

2.1. Introducción

Como se explicitó en la Sección 1.2 los Sistemas de Recuperación de Información (SRI) constituyen la interfaz existente entre las necesidades de información de los usuarios, expresadas mediante un determinado lenguaje de consulta, y la colección o corpus subyacente. En este sentido deben lidiar con el problema de encontrar y presentar aquellos documentos no estructurados que satisfagan la necesidad en cuestión[1] [54]. El concepto *no estructurado* hace referencia a un conjunto de datos con una estructura semántica arbitraria (como así también a la organización física y lógica de los documentos), la cual resulta poco clara a la vez que ambigua para su interpretación mediante una computadora; en oposición por ejemplo a las bases de datos relacionales, en la cuales los datos poseen una estructura con un formato previamente definido y un significado implícito (sintaxis y semántica no ambigua). El término *documento* concierne a la granularidad de la información presentada al usuario, *e.g.*, abstracts, artículos, páginas Web, capítulos de libros, e-mails, oraciones, snippets, entre tantos otros. Incluso la acepción no se encuentra restringida solo a información textual, ya que un usuario puede encontrarse interesado en recuperar y acceder a datos multimedia, como videos, audio o imágenes. Por último, *colección* hace alusión al repositorio

[1] No debe olvidarse que la inherentemente cuota de subjetividad contenida en el interés del usuario implica que el problema de satisfacer una necesidad de información siempre se encontrará abierto [35].

de documentos del cual la información es recuperada.

Es importante destacar que la Recuperación de la Información (RI) no resulta ser un área nueva sino que encuentra sus orígenes en el siglo pasado, sin embargo en la actualidad adquiere un rol realmente significativo debido al valor que presenta la información [76]. Por otro lado, incluye un amplio espectro de tareas como la clasificación (asociar un documento a una o más clases a partir de un conjunto predeterminado, *e.g.*, detección de Spam dentro de una colección de e-mails), *clustering* (agrupamiento de documentos con tópicos en común de acuerdo a su contenido) o filtrado (descarte de documentos que no presentan relación con un tema en particular). Como puede observarse, todas estas actividades comparten la necesidad de contar con la representación de un documento de modo que pueda ser comprendida y procesada por un sistema computacional. De acuerdo a Baeza-Yates, es plausible abordar el problema de la RI desde dos puntos de vista distantes a la vez que complementarios [14]: (A) el *computacional*, relacionado al diseño y construcción de estructuras de datos y algoritmos eficientes que mejoren la calidad de las respuesta, y (B) el *humano*, avocado al estudio del comportamiento de los usuarios, sus necesidades y cómo esto afecta la organización y operación sobre la información. Asimismo, la tarea de recuperar información puede plantearse de diversas formas en relación a la manera en el que usuario interactúa con el sistema, o bien a partir de las facilidades que este le presenta. A considerar:

Recuperación inmediata: El usuario define su necesidad de información y posteriormente, obtiene referencias a los documentos que el sistema determina como relevantes.

- Recuperación *ad-hoc*, en general la necesidad de información que el usuario presenta se traduce a una consulta en texto libre que posteriormente el sistema procesa y evalúa.
- Navegación o *browsing*, el sistema ofrece una interfaz con tópicos por los cuales el usuario "navega" obteniendo referencias a documentos relacionados (*e.g.*, *Open Directory*[2]). En este caso no se expresa una consulta explícita, hecho que facilita la búsqueda a usuarios que no cuentan con una necesidad clara.

Recuperación Diferida: El usuario detalla sus necesidades definiendo un "perfil" de modo tal, que el sistema entrega de forma continua aquellos documentos que se incorporen a la colección y concuerden con el mismo (*i.e.*, relevantes). Esta práctica recibe el nombre de *filtrado y ruteo*.

Los SRI pueden distinguirse en relación a la escala en la que operan y al grado de especialización (*i.e.*, *horizontales* y *verticales*), considerando

[2]Página Principal: `http://www.dmoz.org/`

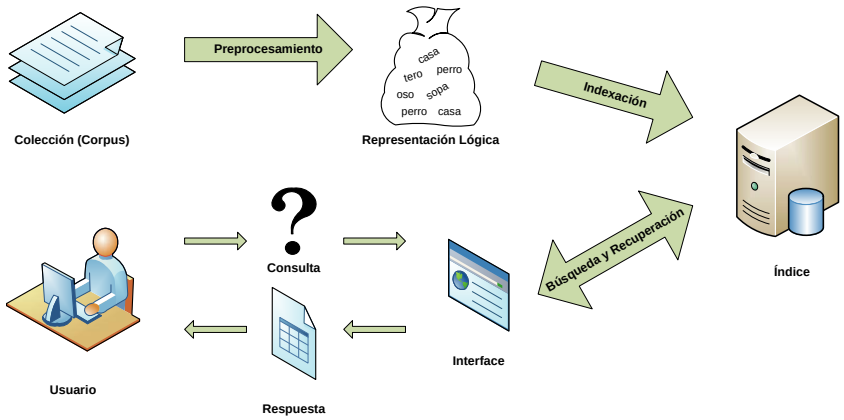


Figura 2.1: Arquitectura básica de un SRI

la cantidad y el tipo de datos con los que deben lidiar, como así también los diferentes retos que a estos competen. Un ejemplo de sistemas de recuperación de propósitos generales son los Motores de Búsqueda Web (MBW), los cuales se encargan de procesar miles de millones de documentos Web [54]. Para el presente trabajo resultan de especial interés los sistemas de búsqueda en tiempo real que operan sobre corpus de provenientes de sitios de microblogging; en contraposición a los primeros el ámbito resulta de mayor especificidad, al igual que las restricciones y limitaciones que se tornan necesarias afrontar (Sección 1.3.1).

2.1.1. Arquitectura de un SRI

Como se observa en la Figura 2.1 un SRI se conforma de un conjunto de componentes esenciales, los cuales permiten el desarrollo de dos procesos fundamentales a la tarea en cuestión: la indexación por un lado y la búsqueda y recuperación por otro. El primero concierne a la construcción del índice invertido y sus estructuras asociadas (Sección 2.4) a partir de la obtención de una representación lógica (Sección 2.2) de los documentos. El segundo es responsable de emplear dicha estructura para resolver las consultas entrantes y generar así, la lista de resultados correspondiente (Sección 2.3). Por último, una interfaz permite la interacción con el sistema de manera tal que sea posible especificar la consulta en un lenguaje preestablecido y presentar la respuesta de la misma (por lo general se trata de referencias a documentos que pueden contener información útil).

En el caso de los MBW es posible advertir la presencia de un módulo adicional, el denominado *recolector* o *crawler*. La intrínseca naturaleza distribuida de la Web requiere recolectar los documentos y almacenar copias de los mismos en un repositorio central previamente a ser indexados. El *crawling*

Web se constituye como aquel proceso encargado de concretar esta tarea, resultando ser su principal objetivo obtener de la forma más rápida y eficiente posible el mayor número de páginas Web junto con las estructuras de enlaces que las interconectan [54]. La colección obtenida resulta de un tamaño tal que tanto la construcción eficiente del índice, como su almacenamiento no son factibles para una sola máquina y por ello, es necesaria la partición de esta estructura a través de varias computadoras interconectadas (*cluster*). Tal organización del índice puede desarrollarse [12] (a) por *términos* - u organización global del índice - donde el conjunto total de términos y sus correspondientes listas de posteo son distribuidas a lo largo de los nodos que componen el sistema, ó (b) por *documentos* - u organización local del índice - caso en cual cada nodo contiene un archivo invertido para un subconjunto de los documentos.

Es importante destacar que cada enfoque cuenta con ventajas y desventajas, ya que por ejemplo en principio la primer estrategia da lugar a una gran concurrencia debido a que un flujo de consultas con términos diferentes puede dirigirse a conjuntos de máquinas diversos, en la práctica su aplicación no resulta trivial pues las consultas con más de un término implican el envió de extensas listas de posteo entre los nodos [54]. Si bien esta cuestión no resulta un problema al particionar la colección por documentos, sí se presenta una dificultad en la disponibilidad de ciertas estadísticas globales empleadas para generar el ranking, las cuales deben computarse de forma distribuida y actualizarse periódicamente. Un componente denominado *broker* se encuentra encargado de ingerir el flujo de consultas entrantes y remitirlas a los correspondientes nodos de búsqueda de acuerdo a la manera en que se encuentre organizado el índice. Como así también es responsable de recibir los resultados intermedios de cada uno de los distintos nodos y generar la lista final.

En la actualidad, Earlybird [20] el sistema de recuperación que da soporte a los servicios provistos por Twitter, representa un hito en el diseño de motores de búsqueda en tiempo real. Si bien se encuentra construido para responder a las necesidades específicas de este sitio de microblogging, posee ciertas similitudes con los sistemas de búsqueda tradicionales. En la Figura 2.2 es posible advertir la arquitectura que este servicio de búsqueda en tiempo real presenta. A diferencia de los MBW la colección no necesita ser recolectada pues se trata de documentos generados por los usuarios en el propio sitio. Una vez emitidos, cada tweet ingresa a un componente denominado *ingestion pipeline*, donde se concretan diferentes operaciones previas al proceso de indexación. Con el objeto de manejar extensos volúmenes, los tweets se distribuyen a través de varias "instancias" de Earlybird en las cuales se almacenan segmentos del índice. Un componente denominado *blender* es el encargado de recibir las consultas de los usuarios y rematarlas a las respectivas instancias. Finalmente, el *updater* provee dinámicamente a las diferentes

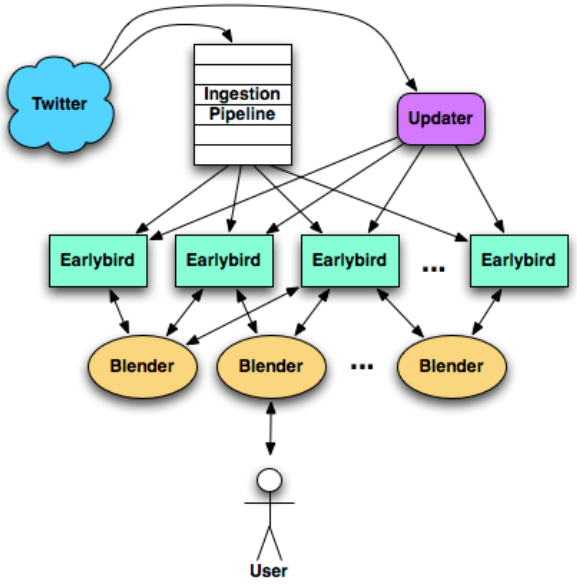


Figura 2.2: Arquitectura del servicio de búsqueda en tiempo real de Twitter. Figura original presentada en [20]

instancias indicadores de relevancia correspondientes a eventos asociados a los tweets (como número de retweets que un tweet recibe, tópicos "calientes" o *trending topics*, entre otros).

2.2. Preprocesamiento

Tal como se explicitó en la Sección 1.2, para que sea posible implementar cualquier mecanismo de recuperación sobre una colección resulta imperioso obtener una representación lógica de los documentos que la conforman. Es claro que no todas las palabras que componen un documento son igualmente representativas de su contenido [14]. Cuestiones tales como su posición, la cantidad de ocurrencias o su función lingüística definen el grado de importancia que cada una posee. La aplicación de un conjunto de criterios en una operación conocida como *preprocesamiento* permite obtener aquellas unidades que posteriormente se emplearán para construir el índice.

De acuerdo a Baeza-Yates, el preprocesamiento de un documento es un proceso que involucra cinco operaciones o transformaciones sobre el texto en cuestión [14], a saber:

Análisis lexicográfico. Cada uno de los documentos que conforman la colección se construyen a partir de sucesiones de palabras que forman estructuras gramaticales (*e.g.*, oraciones y párrafos) y se escriben en lenguaje

natural. El objetivo del análisis lexicográfico consiste en transformar este flujo de caracteres en palabras o *tokens* (unidades candidatas a ser adoptadas como términos de indexación), de aquí que este proceso recibe también el nombre de *tokenización* [37, 54, 27]. Este hecho implica identificar el comienzo y fin de las palabras en diversas circunstancias. Bajo una concepción simplificada se resume al reconocimiento de espacios como separadores de palabras[3]. Sin embargo, es deseable que el analizador sea capaz de lidiar con símbolos no alfabéticos como dígitos y caracteres especiales que conforman las palabras (acentos, diéresis, apóstrofes, etc.), los cuales en numerosas ocasiones son sustituidos por el caracter base relacionado. Por lo general, los números no se consideran términos de indexación útiles, pues sin un contexto que los englobe resultan intrínsecamente vagos. A pesar de ello, pueden presentarse junto a una palabra introduciendo un significado especial (*e.g.*, 415A.C.) y en tal caso, no resulta tan claro que medida aplicar. De aquí que su tratamiento e inclusión en la lista de candidatos a indexar depende del tipo de colección y de la importancia que posean como posibles unidades de búsqueda [76].

Complementariamente, ciertas veces es necesario normalizar y expandir acrónimos, al igual que considerar guiones como conectores de palabras. Las fechas, los nombres propios y ciertas abreviaciones también merecen especial atención. Finalmente, el texto es transformado a mayúsculas o minúsculas. Es importante que señalar que si el texto a procesar posee marcas de formato (como aquellos documentos construidos mediante SGML) son removidas en esta etapa, destacando que es posible conservar ciertos atributos para su posterior utilización (*e.g.*, títulos de documentos, autores, entre otros). Como se advierte, estas operaciones pueden implementarse sin problemas aunque debe prestarse especial cuidado a cada una de ellas, principalmente por su profundo impacto en etapas posteriores [37, 14].

Al analizar documentos provenientes de sitios de microblogging esta operación se torna ciertamente compleja pues en primera instancia los usuarios tienden a producir mensajes altamente condensados (debido al límite de longitud que estos servicios imponen) omitiendo ciertos caracteres siempre que sea posible (*e.g.*, espacios en blanco), ignorando el uso apropiado de mayúsculas y minúsculas y creando abreviaciones no convencionales [48]. Por otro lado, la naturaleza casi instantánea en la que se desarrolla la difusión promueve la presencia de "marcas orales" en los mensajes, tales como emoticones o el empleo no estándar de signos de puntuación. Complementariamente, en el contexto de Twitter debe considerarse el uso de los operadores "@" (*menciones*) y "#" (*hashtags*). Estos conllevan un significado específico, ya que constituyen *metadatos generados por los usuarios* [31] y son regularmente empleados tanto por creadores como consumidores de contenido.

[3] Múltiples espacios son reducidos a uno solo.

Eliminación de palabras vacías. Aquellos términos que presentan gran frecuencia en la mayoría de los documentos de una colección no resultan de gran utilidad (sino nula) para el proceso de recuperación, pues no son buenos discriminadores del contenido de un texto. De hecho, una palabra que ocurre en el 80 % de los documentos no debería considerarse como término de indexación [14] y como tal desestimarse. Este conjunto recibe el nombre de palabras vacías o *stopwords* y se conforma de artículos, preposiciones y conjunciones propias del lenguaje tratado. Asimismo, puede expandirse incluyendo verbos, adverbios y adjetivos. Por otro lado, su eliminación permite reducir el tamaño del almacenamiento necesario para soportar el corpus, hasta un 40 % [33].

A pesar de las ventajas que presenta la aplicación de esta estrategia en diversas ocasiones puede dañar la exhaustividad, ya que ciertas tipos de consultas especiales (como frases, títulos de canciones u otras construcciones) resultan desproporcionalmente afectadas [54]. La tendencia adoptada a través del tiempo ha variado desde la utilización de largas listas de palabras vacías hasta la no eliminación. Los MBW por lo general omiten esta instancia y conservan estas palabras.

Steeming. Parte de la expresividad que proporciona el lenguaje natural está provista por la abundante cantidad de maneras de transmitir una idea. Esto puede resultar un problema para los SRI, ya que se sustentan en la coincidencia de palabras para la búsqueda de documentos relevantes. Por ello en lugar de restringir esta concordancia a unidades que resulten idénticas, una serie de técnicas se han propuesto con el fin de permitir a estos sistemas relacionar palabras que se hayan semánticamente vinculadas. El *steeming* se constituye como una técnica de reducción (*conflotation* [27]) que permite identificar variantes morfológicas de un mismo término y reemplazarlas por el término raíz o lema. Plurales, gerundios y sufijos característicos de tiempos verbales son ejemplos de tales variantes. Una distinción importante es aquella presente entre el concepto de *steeming* y el de *lematización* [54]. Mientras el primero se refiere a un proceso heurístico "crudo" que simplemente recorta el final de las palabras, el segundo se sustenta en la utilización de un vocabulario y la aplicación del análisis morfológico con el fin de remover sólo desinencias y retornar la base de una palabra.

La aplicación de cualquiera de ellas permite: (a) contar con índices de menor tamaño, y (b) disponer de una mayor cantidad de respuestas a una consulta dada, pues previene que las variantes morfológicas afecten la concordancia entre los términos de una consulta y aquellos presentes en los documentos. Sin embargo, esta puede considerarse como una desventaja en ciertos casos ya que aumenta la exhaustividad y disminuye la precisión [76].

Selección de los términos a indexar. De acuerdo a Luhn [53], la impor-

tancia de un término debería establecerse en base a su frecuencia de aparición sobre el texto a procesar. El propósito de esta etapa consiste en obtener el conjunto de palabras (simples o compuestas[4]) que representen de la mejor manera el contenido de los documentos. Los términos de alta frecuencia se eliminan con la asistencia de un diccionario de palabras vacías y aquellos de baja frecuencia, opcionalmente, también son removidos. El conjunto resultante de las operaciones desarrolladas hasta aquí conforma el vocabulario de la colección.

Una cuestión que importante a enfatizar es que aquel conjunto de operaciones de preprocesamiento aplicadas sobre los documentos, debe aplicarse de igual manera a las consultas [27]. Si estas resultan ser diferentes, se dañarán severamente las prestaciones del sistema de recuperación, pues muchos de los términos del índice no coincidirán con los correspondientes términos de la consulta.

2.3. Modelos de Recuperación

Una vez que se han identificado los términos de la consulta y se han recuperado sus correspondientes listas desde el índice invertido, el próximo paso concierne a la evaluación propiamente dicha. Esto implica determinar cuáles son las premisas consideradas para establecer si un documento es relevante o no a una necesidad de información. De manera más específica se define una medida de similitud que una vez computada, permita determinar la relevancia (o no) de los documentos y obtener la lista de resultados o ranking. A este conjunto de premisas se lo conoce como modelo de recuperación.

Uno de los modelos que mayormente se emplea en la práctica es el modelo vectorial [69, 68, 70]. Se basa en cálculos que permiten introducir un orden (ranking) en los documentos recuperados en función de su relevancia respecto de la consulta. Plantea la necesidad de emplear una función de similitud entre el documento y la consulta, para lo cual a cada término se asocia un peso o valor computado en base a la frecuencia que este ha presentado en cada documento respectivamente. Estos pesos se utilizan para calcular el grado de similitud previamente descripto. Por lo general se emplea la métrica de ponderación TF*IDF (*Term Frequency, Inverse Document Frequency*). Este método plantea establecer una relación entre la frecuencia de un término dentro de un documento y su frecuencia en los documentos de la colección. Básicamente se obtiene la frecuencia pura[5] del término t_i en el documento d_j (TF) y se lo multiplica por la inversa de la cantidad de documentos de la

[4]Dos o más palabras que se encuentran asociadas y poseen un significado específico, propio del dominio temático en cuestión. Por ejemplo: sistema operativo. También se las conoce como *multipalabras*.

[5]Opcionalmente se normaliza con la longitud d_j ó la máxima frecuencia de un término en dicho documento.

colección donde aparece t_i (IDF).

De esta manera, se elabora el ranking de forma descendente por el valor de similitud obtenido para cada documento en base a sus términos y se retorna la respuesta al usuario. Este modelo toma en consideración documentos que coinciden parcialmente con los términos de la consulta. El principal resultado de este efecto es que los documentos de la lista de resultados resulta de mayor precisión que aquel conjunto recuperado bajo otros modelos de recuperación [14], como el modelo booleano.

2.4. Estructuras de Datos

De acuerdo a lo planteado hasta aquí, todo SRI parte de un conjunto de documentos o corpus, los cuales debe procesar (de alguna manera) para dar respuesta a las consultas que lo usuarios remiten. Una opción obvia subyace en recorrer la colección por completo, documento por documento, calculando la relevancia que cada uno posee en relación a la consulta. Este enfoque recibe el nombre de búsqueda *secuencial* o *por texto libre* y, además de ineficiente, sólo resulta aplicable sobre colecciones pequeñas (*i.e.*, unos pocos megabytes [14]), en aquellas donde el contenido resulta dinámico o bien, en ocasiones donde no se cuenta con los recursos necesarios para gestionar el espacio de almacenamiento adicional que requieren estructuras auxiliares.

Asimismo, si el sistema emprende operaciones de preprocesamiento (Sección 2.2) sería necesario ejecutar nuevamente cada una de ellas cuando se desee resolver una consulta y por ende, resulta deseable disponer de un conjunto de archivos con la representación lógica de los documentos de la colección. Esto da lugar a que las operaciones previamente mencionadas se concreten una única vez. Aunque, aún bajo dicha estrategia el costo computacional computacional continua siendo alto[6], pues se torna necesario recorrer toda la colección.

Con el propósito de alcanzar mayor eficiencia en la tarea de recuperación, se construyen estructuras de datos auxiliares que soportan la representación lógica de todos los documentos del corpus. De forma genérica, se las conoce como *índices*, y permiten el acceso directo a los documentos que contienen los términos de la consulta. Su contenido está dado por el conjunto de términos que contienen todos los documentos de la colección, *i.e.*, el vocabulario. Una manera de representar la información que debe contener un índice es la matriz documento-término (Figura 2.3). Mientras las filas representan los documentos d_j, las columnas corresponden a los términos t_i del vocabulario. La intersección (d_j, t_i) corresponde al peso o ponderación que se le asigna a t_i en d_j, la cual en el caso más simple puede ser un 0 o un 1, denotando la ausencia o presencia del término en el documento o bien un valor surgido de

[6] $O(n)$ donde n es la cantidad de documentos que componen el corpus [76]

	t_1	t_2	t_3	...	t_n
d_1	$w_{1,1}$	$w_{1,2}$	$w_{1,3}$	...	$w_{1,n}$
d_2	$w_{2,1}$	$w_{2,2}$	$w_{2,3}$	...	$w_{2,n}$
d_3	$w_{3,1}$	$w_{3,2}$	$w_{3,3}$	...	$w_{3,n}$
d_4	$w_{4,1}$	$w_{4,2}$	$w_{4,3}$	...	$w_{4,n}$
d_5	$w_{5,1}$	$w_{5,2}$	$w_{5,3}$	...	$w_{5,n}$
...	...	...	...	...	...
d_n	$w_{n,1}$	$w_{n,2}$	$w_{n,3}$	...	$w_{n,n}$

Figura 2.3: Matriz de asociación documento-término

un criterio de ponderación (Sección 2.3).

Entre las técnicas de indexación más conocidas se encuentran: archivos invertidos, arreglo de sufijos y archivos de firmas, siendo que los primeros se consideran las estructuras más eficientes y flexibles [27], hecho por el cual se han convertido en la elección por excelencia de todo motor de búsqueda moderno. Partiendo de la matriz documento-término (Figura 2.3) se construye un índice que almacena su representación. Esta estructura recibe el nombre de archivo o índice invertido y en su forma más básica constituye un conjunto de términos donde cada uno posee asociada una lista de los identificadores de documentos en los cuales cada término aparece. En otras palabras, cada entrada de este archivo mapea un término con un conjunto de documentos que lo contienen. Para su construcción puede trasponerse la matriz documento-término obteniendo la matriz término-documento, de aquí la razón de por qué se lo denomina *invertido*.

El archivo invertido resulta ser el equivalente computacional al índice encontrado en la sección trasera de cualquier libro de textos. Al igual que este índice se organiza de forma alfabética por los términos que lo componen también lo hace el archivo invertido[7]. Adicionalmente, cada término de este índice se acompaña de una lista de páginas en las cuales la palabra en cuestión fue citada. De la misma manera, cada entrada del archivo invertido se asocia a una estructura denominada *lista de posteo* (o *lista invertida*), la cual alberga un listado de los documentos donde el término ocurre. Cada elemento de

[7] Vale destacar que el orden alfabético no es estrictamente necesario ya que puede emplearse una tabla hash para la búsqueda.

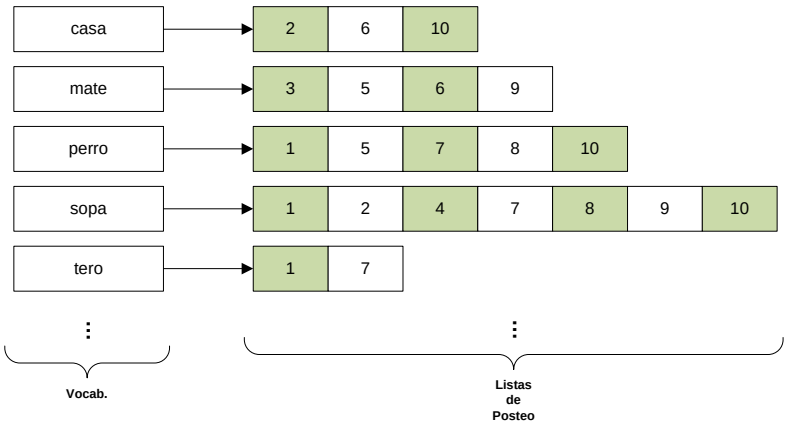


Figura 2.4: Componentes básicos de un índice invertido

esta lista recibe el nombre de *posting*, y la sección de esta unidad que hace referencia a un documento específico se denomina usualmente *puntero*. Al conjunto de las listas de posteo se lo conoce como *postings*. Cada documento de la colección se asocia a un número que lo identifica unívocamente (docID) dentro de la colección con el fin permitir la eficiencia en su almacenamiento. Adicionalmente, cada entrada de la lista puede contener información extra como la frecuencia del término en el documento o su posición en texto. Tal decisión dependerá del modelo de recuperación adoptado.

Tradicionalmente, las listas de posteo se ordenan por los identificadores de los documentos [3, 87, 27], hecho que permite que ciertas estrategias de procesamiento de consultas se concreten con mayor eficiencia, además de propiciar la compresión de las listas. A pesar de ello, las listas pueden ordenarse mediante otros tipos de criterio, como la frecuencia [66] o el puntaje de impacto [5, 6, 4]. Cada uno presenta diferentes compromisos (*tradeoffs*) entre requisitos de espacio, organización necesaria para facilitar la compresión, mecanismos de actualización y mantenimiento y soporte de diversos algoritmos de evaluación de consultas.

Resumiendo, el índice invertido (Figura 2.4) se conforma de dos estructuras de datos fundamentales. Por un lado, el vocabulario (o lexicon), que contiene una entrada por cada token indexado de la colección (y por ende sin repeticiones), junto con otras estadísticas globales[8] y una referencia a su correspondiente lista. Por otro, las listas de posteo, que albergan aquella información que relaciona términos y documentos. Mientras que el lexicon puede mantenerse en memoria, pues resulta órdenes de magnitud menores que las listas de posteo, estas últimas residen comúnmente en el disco [35]. Complementariamente, por cuestiones de eficiencia es posible representar los

[8] Como el número de documentos en los que el término ocurre, conocido como frecuencia en los documentos ó simplemente *df*.

términos mediante identificadores unívocos (en reemplazo de cadenas de caracteres) [54]. La construcción del índice invertido es un proceso por lotes [54] que se lleva a cabo fuera de línea, sin interacción alguna por parte del usuario y por ello, no es posible comenzar con esta operación hasta tanto no se haya obtenido la colección por completo.

2.4.1. Indexación Incremental

El proceso de construcción del índice descripto previamente asume que los documentos que comprenden la colección nunca cambian. Sin embargo, ciertas colecciones resultan ser dinámicas, hecho que implica que su composición sufre modificaciones con el paso del tiempo, nuevos documentos se agregan, otros se modifican y muchos desaparecen (recuérdese las características de la Web, Sección 1.1). En su mayoría, la literatura se avoca a resolver el problema de agregar nuevos documentos. La indexación dinámica o incremental es también conocida como indexación en-línea, ya que los documentos arriban al sistema una vez que este se encuentra en pleno funcionamiento, es decir disponible para recibir consultas.

Como se mencionó, las listas de posteo son generalmente almacenadas en memoria secundaria. Debido a las características físicas que estos dispositivos presentan es conveniente disponer de los datos de forma contigua si se desea maximizar el rendimiento en las operaciones de entrada/salida [54]. Por ello, concretar ciertos cambios sobre las postings involucra extender sus estructuras incurriendo en procesos que consumen gran cantidad de tiempo. Esto puede degradar en gran medida la efectividad del sistema recuperación [35], ya que ciertos fragmentos del índice no se encuentran disponibles para su consulta mientras dure la correspondiente actualización. La opción más simple para lidiar con esta cuestión consiste en periódicamente re-construir el índice por completo. Este suele ser un buen enfoque si el número de cambios acumulados en el tiempo resulta relativamente reducido y sobre todo, si el retraso para permitir la búsqueda sobre los nuevos documentos puede admitirse. Asimismo, es necesario contar con los recursos suficientes para construir el nuevo índice mientras el anterior aún se encuentra disponible para servir consultas. Actualmente, muchos MBW prosiguen este patrón [35].

De esta manera, es posible advertir cómo las actualizaciones sobre el archivo invertido adicionan complejidad y tensiones a ser consideradas durante el proceso de construcción del índice, tales como el procesamiento de consultas y la tasa de inserción de documentos. Además, el hecho de brindar soporte a operaciones de eliminación/actualización fuerza la utilización de procedimientos de mantenimiento más complicados a la vez que sofisticados. Por otro lado, contar con un índice invertido en el cual las listas de posteo se almacenan de forma contigua sobre el disco otorga gran velocidad a la evaluación de consultas, pues una búsqueda en disco es suficiente para encontrar y leer la lista de posteo asociada a un término. Sin embargo mantener la

contigüidad de las listas luego de una actualización introduce gran complejidad en el diseño de los algoritmos encargados de ello, además de conducir a operaciones poco eficientes. Asimismo, si se considera un índice en el cual se concibe cada lista de posteo como una lista enlazada discontigua de entradas, las actualizaciones pueden concretarse de forma rápida. Aunque la resolución de consultas se convierte en una operación de reducidas prestaciones ya que leer una lista de posteo involucra reiterados accesos a disco. De esta manera, es posible plantear que el desafío que los algoritmos de actualización enfrentan subyace en la búsqueda de un compromiso entre la velocidad de actualización y la eficiencia en la evaluación de consultas [7].

2.4.2. Indexación de Flujos de Documentos

En situaciones donde se torna imperioso que las actualizaciones sobre la colección se encuentren disponibles lo antes posible, es deseable dotar al SRI de la capacidad de indexar documentos mientras estos son recolectados o bien, al mismo instante en que ingresan al sistema, como sucede en los servicios de microblogging. En tales situaciones se espera que los documentos ingeridos formen parte de índice prácticamente sin retraso alguno, con el fin de poder incluirlos en el espacio de búsqueda (Sección 1.3.1). En otras palabras, no puede concebirse al proceso de indexación como una operación por lotes pues los usuarios esperan que el contenido se encuentre disponible en términos de segundos [20, 11]. Esto implica que aquel componente encargado de construir y mantener el índice y sus estructuras asociadas esté preparado para recibir un flujo continuo de documentos (con tasas de arribo elevadas y usualmente, con ráfagas repentinas) y alcance tanto baja latencia como alto throughput en el desarrollo de sus operaciones.

Si bien la indexación incremental ha sido abordada en literatura [77, 59, 51, 39], la mayoría de estos enfoques asumen que el índice invertido no cabe en memoria principal y en consecuencia, debería residir en el disco. El hecho de recuperar secciones del archivo invertido desde este dispositivo puede impactar de forma negativa sobre tiempo insumido en el procesamiento de consultas [74], pues por sus características físicas se considera un dispositivo de acceso "lento". Contrariamente, la memoria principal presenta mayor velocidad tanto para la lectura como para la escritura de datos [54] y por ello, mantener el índice invertido allí permite alcanzar mayores prestaciones en términos de eficiencia. Incluso, en situaciones donde la indexación y la evaluación de consultas deben desarrollarse de forma concurrente, construir un índice en el disco introduce un mayor desafío [7].

Uno de los ejemplos más destacados en el diseño de índices en memoria para flujos de documentos es Earlybird [20]. Este motor de búsqueda desarrollado bajo las necesidades específicas del contexto en el que opera (*i.e.*, Twitter) construye y mantiene el índice invertido y todas sus estructuras asociadas completamente en memoria principal. Basándose en sus princip-

ios de diseño y considerando los requisitos que la búsqueda en tiempo real demanda Asadi et al. [11] presentan un esquema orientado a generalizar la construcción y actualización de índices invertido completamente en memoria. Principalmente plantean la necesidad de contar con una política de asignación de memoria que permita alcanzar el balance adecuado entre tiempo (velocidad en la evaluación de consultas) y espacio (utilización de la memoria). Una política que resulte demasiado agresiva conlleva a la subutilización de la memoria, ya que en su mayoría el espacio asignado se encuentra vacío. Mientras que definir un política excesivamente conservativa reduce las prestaciones del sistema, pues la reserva de memoria es un proceso relativamente costoso, además de provocar la fragmentación de las listas de posteo.

La asignación de espacio para las listas invertidas debe concebirse como un proceso dinámico, ya que su crecimiento se encuentra solamente limitado por el tamaño de la colección. Este factor introduce ciertos desafíos que necesitan ser enfrentados. En primera instancia la longitud de las listas de posteo es muy variable y en consecuencia, la cantidad de memoria a ser asignada para alojar cada una se vuelve una cuestión compleja. Por otro lado, bloques demasiado pequeños resultan en patrones de acceso a memoria subóptimos durante el recorrido sobre las listas. Con el fin de lidiar con estas cuestiones, los autores proponen crear cuatro *pools* para alojar las postings. Cada uno se subdivide en secciones denominadas *slices*, las cuales albergan las postings correspondientes a un término. Las slices del primer pool permiten albergar un número reducido de entradas por lista, siendo que en los pools sucesivos esta cifra resulta cada vez mayor. En la medida en que los respectivos términos incrementen sus ocurrencias, las listas se "expanden" a través de los diferentes pools. Si bien esto genera que las misma no resulten directamente contiguas los distintos segmentos se enlazan mediante punteros a través de los diferentes pools. Finalmente, por cada entrada del diccionario (estructura de datos encargada de almacenar el vocabulario) se mantiene un puntero a la "cola" de la lista de posteo del término en cuestión. Esto permite recorrer la lista en orden cronológico inverso.

Posteriormente, los mismos autores introducen en [10] una evolución al esquema previamente citado. Principalmente, exploran de qué forma impacta la contigüidad de las listas de posteo tanto en el rendimiento de la recuperación como en la construcción y mantenimiento del índice invertido, cuando ambos procesos se concretan en memoria. De esta manera, describen un algoritmo de indexación incremental del cual se destacan tres componentes (Figura 2.5): el diccionario, *buffer maps*, y *segment pools*. Cada término de la colección se mapea con un número entero que lo identifica unívocamente. El diccionario permite asociar cada término con su respectivo identificador, además de albergar otros datos de utilidad al esquema. Un *buffer map* constituye básicamente un arreglo de apuntadores, donde cada posición de su índice se corresponde con el identificador de un término. Cada apuntador se

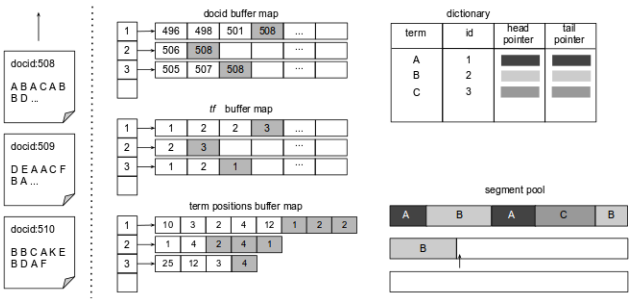


Figura 2.5: Componentes de un indexador en memoria principal. Figura original presentada en [10]

asocia a un arreglo de enteros (*buffers*) y, por lo general, se cuenta con dos buffer maps, uno para almacenar los identificadores (docIDs) de los documentos en los cuales el término ocurre y otro destinado a la frecuencia. En el caso en que se desee dar soporte búsquedas posicionales un tercer buffer map para contener las posiciones de cada término es adicionado. Inicialmente, cada buffer se limita a un bloque de 128 entradas con el fin de maximizar la compresión.

La estrategia consiste en acumular postings en los buffer maps hasta que estos se llenan, para luego comprimirlos y volcarlos al segment pool, donde residen los diversos segmentos de cada lista posteo. Complementariamente, cada unidad que se aloja en el pool cuenta con aquella información necesaria para localizar la siguiente en la cadena. Claramente bajo este enfoque las lista invertidas resultan discontiguas y por ello, los autores plantean una optimización: cada vez que los buffers para un término se completan, se expanden en un factor de dos. Los segmentos aún poseen una longitud de 128 entradas (con el fin de favorecer la compresión) excepto que se almacenan uno a continuación del otro sin necesidad de enlazarlos. Este hecho que permite que aquellas listas de posteo que tienden a extender ese resulten incrementalmente contiguas. Por otro lado, las experiencias permitieron demostrar que bajo este esquema es posible obtener tiempos de respuesta estadísticamente indistinguibles de aquellos alcanzados bajo un índice absolutamente contiguo. Asimismo, existen algoritmos de resolución de consultas, como WAND, cuyo diseño genera patrones de accesos a memoria impredecibles reduciendo la capacidad de utilizar aquellas operaciones de entrada/salida optimizadas que otorgan los procesadores modernos.

Como se puede notar, esta organización del índice invertido resulta diferente a aquella que se adopta cuando esta estructura debe residir el disco. Por otro lado, la configuración de las listas de posteo permite disponer del espacio estrictamente necesario para su almacenamiento, expandiéndolo cuando el caso lo requiera. Además, esto da lugar a que la inserción de nuevos documentos (indexación incremental) no resulte un proceso complejo a la vez que

eficiente, permitiendo así la búsqueda sobre ellos de forma casi inmediata.

2.4.3. Estrategias de Indexación Eficiente

Conforme una colección de documentos incrementa su volumen, más compleja se torna la capacidad de resolver un amplio número de consultas en tiempos de respuesta interactivos. La razón de esto subyace principalmente en el hecho de que uno de los mayores cuellos de botella en el procesamiento de consultas, es la longitud de la estructura encargada de almacenar las listas de posteo, la cual puede crecer fácilmente a miles de MBs o incluso GBs para términos de búsqueda comunes (aproximadamente lineal con el tamaño de la colección) [28, 71]. Esta es la razón por la cual surgen técnicas orientadas a proporcionar una gestión del índice más eficiente.

Por un lado, se cuenta con algoritmos de compresión los cuales se orientan a organizar de forma más eficiente la información almacenada, de forma tal que las listas de posteo cuenten con una representación más compacta, por ejemplo PForDelta [85]. Complementariamente, las estrategias de poda (*pruning*) intentan encontrar la manera de descartar aquella información cuya ausencia no represente una disminución que sea percibida por el usuario en el rendimiento general del sistema. En particular los enfoques de pruning *estáticos* procuran reducir el espacio requerido para el almacenamiento del índice invertido, suprimiendo aquellas entradas cuya potencial contribución al puntaje de un documento es tan pequeña que su ausencia tendrá un efecto despreciable sobre la efectividad global.

En la última década han sido propuestos varios enfoques orientados a desarrollar diversas estrategias de poda estática. Los primeros en estudiar este concepto, fueron Carmel et al. [23] quienes introducen en su investigación dos métodos caracterizados como *centrados en los términos* [2], donde sólo se mantienen en las listas de posteo una determinada cantidad de entradas, en relación al impacto que el puntaje (*score*) individual de cada una de ellas tendría si el término formará parte de una consulta. Büttcher y Clarke. [21] presentan una estrategia de poda estática esta vez definida como *centrada en el documento* [2], ya que el proceso de poda itera sobre los documentos en primera instancia, a diferencia del caso anterior en el cual lo hacía sobre los términos (o listas de posteo), nuevamente con versiones uniforme y adaptativa. Por último, otros trabajos a considerar son [49, 2] donde se propone potenciar las técnicas de poda estática a partir de la utilización de sets de consultas (*query logs*) con el fin de incrementar la precisión del índice obtenido, y [72] en el cual se discute la combinación de este conjunto de métodos con el área de caché de resultados.

Es importante destacar que estos enfoques tradicionalmente se aplican sobre estructuras almacenadas en el disco. Su principal objetivo es disminuir la cantidad de bloques que a leer cuando se evalúa una consulta, evitando incurrir en costosas operaciones de entrada/salida. Como se mencionó, en

el contexto de búsqueda en tiempo real el índice invertido se aloja completamente en memoria y por ello estas estrategias no resultan provechosas, siendo necesarios otros enfoques. Una de las contribuciones de este trabajo es presentar técnicas que permitan una gestión más eficiente del índice cuando este reside en memoria.

2.5. Proceso de Recuperación

Cómo su nombre lo expresa la tarea primordial de todo SRI es recuperar y proporcionar al usuario aquella información relevante su necesidad de información. En particular, cuando la recuperación es *ad hoc* el punto de entrada al sistema es una consulta, la cual debe ser procesada y evaluada con el fin de identificar aquellos documentos que se consideren relevantes a ella. Existen dos técnicas básicas [81, 19, 28, 78, 71] para el recorrido del índice invertido al momento de procesar una consulta. A saber[9]:

- **Documento-a-la-vez (*DAAT*)**[10]: Esta estrategia consiste en evaluar la contribución de cada término de la consulta con respecto a un solo documento antes de proseguir con el próximo, es decir que el puntaje de total de cada documento es computado antes de avanzar al siguiente.

- **Término-a-la-vez (*TAAT*)**[11]: Esta estrategia consiste en procesar los términos de la consulta uno a uno, acumulando puntajes parciales de los documentos involucrados. El puntaje final se desconoce hasta tanto no se evalúen todos los términos.

El primer enfoque presenta dos ventajas: (1) implica un menor consumo de memoria durante la ejecución de una consulta, puesto que no se necesita mantener estructuras de datos intermedias para almacenar puntajes parciales, y (2) aprovecha el paralelismo de E/S con mayor efectividad, ya que es posible recorrer listas de posteo simultáneamente, localizadas en unidades de almacenamiento distintas. Mientras que la segunda técnica es mayormente empleada en sistemas de recuperación de información (SRI) tradicionales demostrando un buen desempeño para colecciones pequeñas [19, 78].

Mientras los métodos de pruning estáticos presentados en la sección anterior tienen por objetivo disminuir el espacio empleado para el almacenamiento del índice invertido, permitiendo de esta forma manejar un volumen de datos de menor magnitud y, en consecuencia, reducir en cierta medida los requerimientos de tiempo y memoria incurridos durante el procesamiento de consultas, las estrategias de pruning dinámico procuran acotar o eliminar

[9] Es importante destacar que existe una tercera estrategia denominada *Puntaje-a-la-vez (SAAT)* [28].

[10] Por sus siglas en inglés, *Document-at-a-time*.

[11] Por sus siglas en inglés, *Term-at-a-time*.

durante la evaluación de cada consulta el cálculo del puntaje de documentos que no formarán parte de los primeros k resultados del ranking, con el fin de ahorrar accesos a disco, descompresión y costos computacionales asociados, beneficiando así la eficiencia.

El algoritmo WAND, presentado en [19], emplea un enfoque *DAAT* y por ende no requiere de estructuras de datos adicionales además del montículo de tamaño k utilizado para almacenar la lista de resultados. Este algoritmo, considerado como una de las técnicas que conforman el *estado-del-arte* en el ámbito de la poda dinámica [28, 78, 71, 79], emplea una estrategia de movimiento de punteros basada en la elección de un pivote de forma iterativa de acuerdo a los términos que el próximo documento debe contener para superar un umbral computado dinámicamente, hecho que permite omitir varios documentos que bajo un enfoque exhaustivo serían evaluados. Es importante destacar, que cada lista de posteo cuenta con un puntero que señala la entrada "actual" de la lista, que avanza en la medida en que la consulta es procesada. De esta manera, a partir cierta información parcial (ocurrencias de cada término, y factores no independientes de la consulta) el algoritmo es capaz de identificar qué documentos, denominados *candidatos*, serán evaluados por completo ignorando largas secciones de la lista que no resultan relevantes a la consulta, favoreciendo de este modo la eficiencia.

Capítulo 3

Invalidadores de Entradas de Índices Invertidos

...the best computer programs are the ones written when the programmer is supposed to be working on something else.

Melinda Varian

3.1. Introducción

El primer paso antes de poder aplicar cualquier tipo de optimización orientada a incrementar el rendimiento del proceso de recuperación, conlleva analizar la colección subyacente. Esto permite caracterizarla y obtener conclusiones acerca de su dinámica y del comportamiento de las palabras en el texto. Una vez concretado tal proceso, es factible emplear dicha información para idear mecanismos que propicien la eficiencia general del sistema.

3.2. Caracterización de la Colección

Tweets2011 [56] se conforma como la colección de referencia de las ediciones 2011 y 2012 del TREC Microblog Track [64, 73]. Este dataset se encuentra diseñado para constituir una muestra reutilizable a la vez que representativa del contexto abarcado por Twitter, además de ser empleada en varios trabajos [26, 11, 9]. Comprende aproximadamente 16 millones de tweets muestreados en el período comprendido entre el 23 de Enero y el 8 de Febrero (inclusive) del año 2011. Debido su metodología de distribución el corpus tiende a degradarse con el paso del tiempo. El principal motivo de ello subyace en el hecho de que los usuarios pueden borrar sus tweets

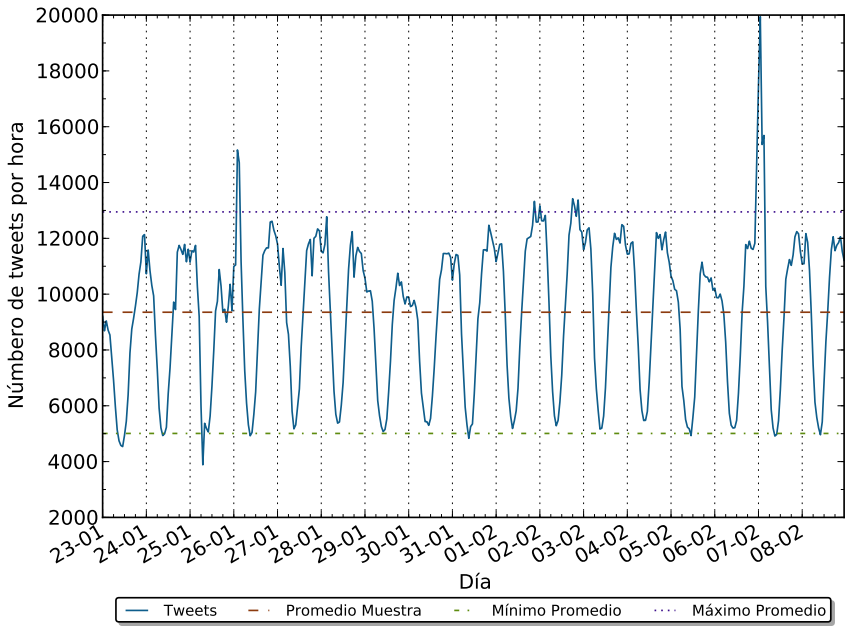


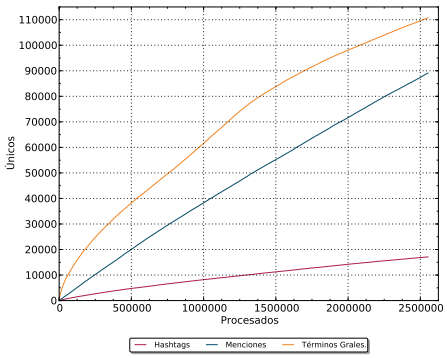
Figura 3.1: Tasa de arribo de tweets

o establecer sus cuentas como privadas y partir de este punto dichos posts dejan de formar parte de la colección.

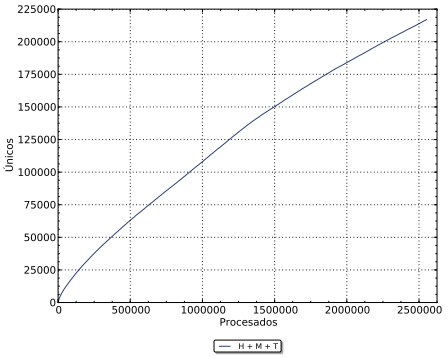
Del volumen original, fueron recolectados con éxito 11,601,066 tweets. Esta cifra comprende posts en múltiples idiomas, por lo cual se mantuvieron sólo aquellos en inglés, obteniéndose un total de 3,815,681 tweets. En la Figura 3.1 puede observarse la distribución de los tweets con el transcurso del tiempo, en promedio 9,352 posts arriban por hora. Cada post se compone de aproximadamente 13.39 palabras útiles y 81.25 caracteres. Conforme a lo planteado en [61], el número de palabras en un tweets resulta considerablemente pequeño, siendo que las palabras raramente se repiten dentro de un mismo documento. Por otro lado, se hallaron en promedio 1.39 y 1.29 hashtags y menciones por tweet, respectivamente.

3.3. Dinámica del Vocabulario

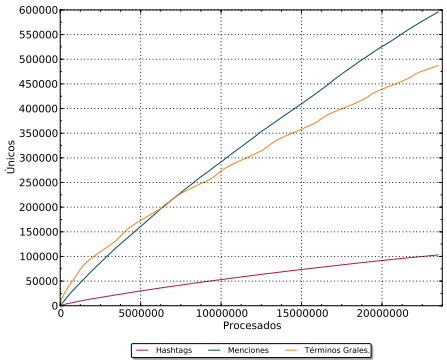
El primer paso para llevar a cabo el estudio del vocabulario y su evolución, fue preprocesar cada post identificando tres tipos de tokens: menciones, hashtags y términos generales. La razón para aplicar esta convención radica en el hecho de que tanto menciones como hashtags poseen un significado y valor particulares dentro de Twitter [31, 75]. En la Tabla 3.1 se presenta la cantidad de tokens hallados para cada tipo junto con el tamaño total del vocabulario (suma de los tres mencionados). De acuerdo la literatura



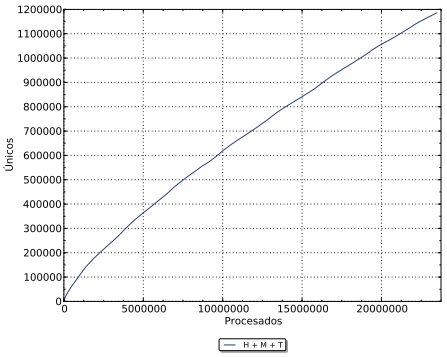
(a) Por tipo de token considerado



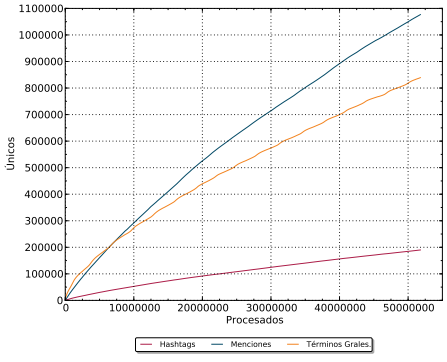
(b) Vocabulario Completo (Sin distinción de tipo)



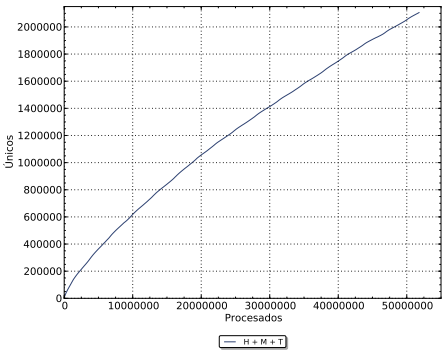
(c) Por tipo de token considerado



(d) Vocabulario Completo (Sin distinción de tipo)



(e) Por tipo de token considerado



(f) Vocabulario Completo (Sin distinción de tipo)

Figura 3.2: Crecimiento del Vocabulario

tradicional de RI una manera práctica de describir cómo el vocabulario y el tamaño de la colección se encuentran relacionados, corresponde a la Ley de Heaps [54, 27]. En particular, permite estimar empíricamente la cardinalidad del vocabulario (y su crecimiento) como una función del tamaño de la colección:

$$v = k * n^{\beta}$$

donde v constituye el tamaño del vocabulario para un corpus de n palabras, mientras k y β representan parámetros que varían de acuerdo a cada colección en particular. De forma general, predice que el número de nuevas palabras se incrementa rápidamente cuando el corpus es pequeño y continúa su incremento indefinidamente, pero a una tasa menos para colecciones extensas [27].

La Figura 3.2 muestra que los tokens considerados crecen más rápido que las predicciones que esta ley establece, exhibiendo más bien un crecimiento lineal. El eje X corresponde al número de tokens procesados. El eje Y muestra los tokens únicos (aquellos que componen el vocabulario). Notése que del conjunto la subfigura 3.2(a) corresponde al vocabulario obtenido para el primer día de la muestra, 3.2(c) al día ocho y finalmente, 3.2(e) al último día. Un comportamiento equivalente se observa sin aplicar una distinción por token (subfiguras 3.2(b), 3.2(d), 3.2(f)). Complementariamente, la Tabla 3.2 muestra la proporción de tokens nuevos hallados luego de comparar el vocabulario obtenido habiendo preprocesado los posts correspondientes al día n y $n + 1$. Diariamente, alrededor de un 72 % de los tokens son nuevas incorporaciones al vocabulario. Este hecho constituye otro indicio acerca de lo dinámico que resulta el vocabulario que presenta este tipo de colecciones.

A pesar de que el número de tokens encontrados se eleva considerablemente rápido, en su mayoría constituyen *hapax*[1], *i.e.*, ocurren en único tweet, representando aproximadamente un 67.7 % del vocabulario total. Como varios trabajos afirman [26, 58, 61], la raíz de este fenómeno se encuentra en la esencia informal que distingue la actividad propia de microblogs junto con su límite de caracteres (140 en el caso de Twitter). En consecuencia, abreviaciones, palabras alargadas, hashtags conformados por varias palabras, jergas de internet y errores de escritura son comunes, en muchos casos deliberados.

Con el fin de profundizar y expandir el análisis hasta aquí desarrollado se estudió cómo la frecuencia de los tokens evoluciona a través de los días, estableciendo tres grupos que permitan conglomerar aquellos con comportamientos equivalentes. Para ello, se empleó un enfoque conocido como *ventana deslizante*. Considérese una ventana de r ranuras, donde cada una corresponde a la observación de un token sobre una base diaria. Si un token ocurre en al menos uno de los documentos procesados del correspondiente día de la muestra, un valor booleano verdadero es establecido en la ranura.

[1] *Hapax legomenon*, o simplemente hapax, es una palabra que ocurre sólo una vez dentro de un contexto

Tipo	Cantidad
Hashtags	189,972
Menciones	1,076,716
Términos Grales.	839,121
Vocabulario	2,105,809

Tabla 3.1: Tamaño del Vocabulario

Diariamente, aquella información correspondiente a la frecuencia de cada token es actualizada a partir de los tweets que ingresan al sistema. Cuando un token aparece por primera vez (*i.e.*, no existe en el vocabulario actual) una nueva instancia de la ventana le es asignada y la primer ranura activada. Así, en la medida en que los días transcurren la ventana recorre las

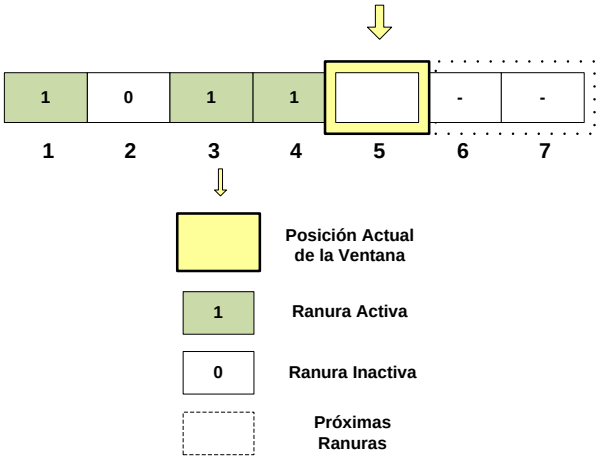


Figura 3.3: Estrategia de ventana deslizante con siete ranuras

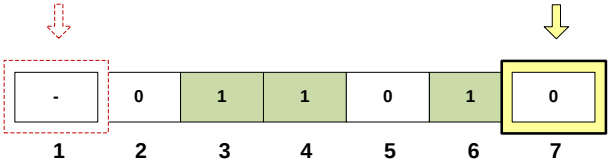


Figura 3.4: Una vez que la ventana alcanza la ranura n, retorna a la posición inicial

Días	Hashtags	Menciones	Términos
23/01 - 24/01	70.68 %	86.30 %	55.36 %
24/06 - 25/01	71.76 %	86.57 %	57.03 %
25/01 - 26/01	70.53 %	85.26 %	55.41 %
26/01 - 27/01	72.27 %	85.78 %	56.90 %
27/01 - 28/01	72.44 %	85.21 %	57.43 %
28/01 - 29/01	74.85 %	87.65 %	59.26 %
29/01 - 30/01	72.80 %	86.29 %	57.19 %
30/01 - 31/01	71.35 %	86.07 %	55.91 %
31/01 - 01/02	71.09 %	85.56 %	56.34 %
01/02 - 02/02	71.71 %	85.65 %	56.86 %
02/02 - 03/02	72.29 %	85.95 %	57.07 %
03/02 - 04/02	72.12 %	85.34 %	57.57 %
04/02 - 05/02	74.74 %	87.55 %	59.11 %
05/02 - 06/02	72.95 %	86.16 %	57.50 %
06/02 - 07/02	71.68 %	86.24 %	56.25 %
07/02 - 08/02	72.21 %	86.38 %	57.16 %

Tabla 3.2: Proporción de tokens nuevos encontrados entre los vocabularios correspondientes al día n y $n+1$

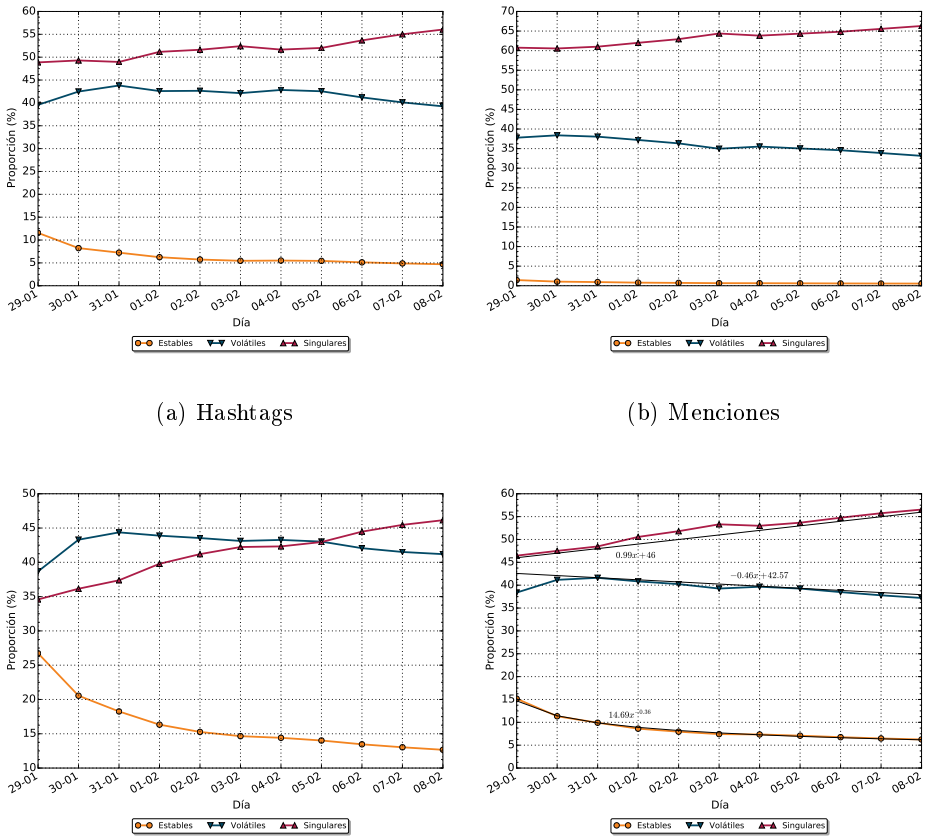
ranuras remanentes, siendo que estas pueden activarse (o no) de acuerdo a la ocurrencia del token en cada día. Como se puede observar en la Figura 3.3 han pasado cincos días desde que el token ocurrió por primera vez en la muestra y además de aquel registro, ha vuelto a ocurrir el día tres y cuatro desde entonces. Una vez que la ventana ha atravesado las siete ranuras un desplazamiento lateral es aplicado, conservando las últimas r observaciones (Figura 3.4). Seguidamente, cada token es clasificado en concordancia con el siguiente criterio:

Sea v_i la ventana de r ranuras que corresponde al token t_i, y v_{ij} su valor en la ranura j. Sea $S_i = \sum_{j=1}^{r} v_{ij}$ y $G(t_i)$ una función que asigna una categoría (o grupo) a cada token de acuerdo a sus ocurrencias diarias, definida como:

$$G(t_i) = \begin{cases} estable, & \text{si } S_i = r \\ volatil, & \text{si } 2 \leq S_i \geq (r-1) \\ singular, & \text{si } S_i = 1 \end{cases}$$

En el presente estudio se consideró $r = 7$ (una semana completa) pues se sostiene que resulta una cifra razonable para examinar el comportamiento de los tokens, dada la dinámica que caracteriza a Twitter. La Figura 3.5 enseña la distribución de los tokens clasificados a lo largo de la muestra. Nuevamente, el análisis fue emprendido por separado para hashtags, menciones y términos

generales, al igual que para el vocabulario completo. Es importante aclarar que es posible comenzar con esta categorización a partir del 28 de Enero, pues hasta entonces la ventana aún no había alcanzado la séptima ranura. A pesar de haber empleado un tamaño de ventana de siete ranuras, es necesario un estudio más exhaustivo para determinar el impacto que este parámetro genera en el esquema de clasificación presentado.



(a) Hashtags

(b) Menciones

(c) Términos Grales.

(d) Vocabulario Completo

Figura 3.5: Proporción de cada tipo de token a través de los días

En el caso del vocabulario completo es posible apreciar que los tokens clasificados como "singulares" crecen de forma prácticamente lineal pues su recta de ajuste presenta una pendiente de 0,99, además de exhibir una tasa de crecimiento diaria de aproximadamente 1 %. En el caso los tokens "volátiles", desestimando la primer observación, es posible ajustar el comportamiento a una función lineal con pendiente $-0{,}46$, donde se observa una tasa de decrecimiento diaria cercana al 0,98 %. Finalmente, los tokens "estables" se ajustan a una ley de potencias ($P_{(x)} = C * x^{-\beta}$) con un β igual a 0,36, siendo

que decrecen a una tasa diaria aproximada del 10 %. En particular, este último grupo permite advertir que si bien continuamente nuevos tokens se adicionan al vocabulario, existen un subconjunto considerablemente menor que se estabiliza y se mantiene conforme transcurren los días.

3.4. Invalidadores de entradas de índice

De acuerdo al análisis expuesto previamente, es posible advertir cuán dinámico resulta el vocabulario obtenido tras preprocesar la colección. Al igual que plantean Lin & Mishne [52], es posible observar en los tweets un fenómeno denominado *churn*. Básicamente, este plantea cuán rápido evoluciona la distribución de los tokens tanto en los tweets como en las consultas, de manera que aquellos tokens que presentaron mayor frecuencia en un intervalo temporal (*e.g.*, una hora), pueden ser muy distintos de aquellos hallados en el período subsiguiente.

Es por tal razón que se torna razonable considerar la poda de ciertas entradas del índice invertido. En particular, aquellos tokens cuyas ocurrencias a través de los días permiten aproximar cuán reducida es su contribución a los resultados de búsqueda. Por otro lado, si bien actualmente se dispone de equipamiento de altas prestaciones, el cual junto a los avances en arquitecturas de búsqueda distribuida, debilitan aquella suposición que plantea la necesidad de alojar el índice invertido en almacenamiento secundario en lugar de hacerlo en memoria [7], no siempre es posible acceder a este tipo de hardware, principalmente por su costo. De esta manera, resulta imprescindible abordar el diseño de algoritmos eficientes que permitan aminorar los requisitos que la indexación en almacenamiento primario plantea con el propósito de dar soporte a búsquedas en tiempo real. Para tal fin, se define un *invalidador de entradas de índice* (IEI), siendo su objetivo invalidar y desalojar selectivamente aquellas entradas del índice invertido cuya ausencia no degrade considerablemente la efectividad en la recuperación. En este trabajo se introducen dos IEI, los cuales se describen a continuación.

3.4.1. Basado en tiempo-de-vida

El primero de los IEI se basa en una estrategia denominada tiempo-de-vida (TTL[2]), como aquella aplicada en caches de resultados [22]. El IEI lleva cabo el seguimiento del tiempo que han persistido las entradas en el índice invertido sin ser actualizadas, de manera que cuando un token excede un valor preestablecido, se elimina junto con su correspondiente lista de posteo. En otras palabras, una entrada es removida cuando la diferencia entre la marca de tiempo actual y aquella que indica su última actualización, resulta mayor al valor del TTL. Cómo su nombre lo indica, este enfoque permite

[2]Por sus siglas en inglés: *time-to-live*.

definir de alguna manera la persistencia de las entradas en el índice, de manera que cuanto menor resulte este valor, mayor deberá ser la frecuencia con la que los tokens ocurren en los sucesivos documentos para permanecer en el índice. Un valor demasiado bajo puede desgastar considerablemente la efectividad, mientras que uno muy alto puede incrementar el tamaño del índice afectando la eficiencia, pues como se observó en secciones previas, el vocabulario crece a un ritmo realmente elevado.

Por ejemplo, suponga que se define un valor de TTL igual a 12 horas. Cuando un nuevo documento ingresa al sistema, comienza procesarse con el fin de extraer aquellas palabras candidatas a formar parte del índice. Complementariamente, considérese que uno de los tokens encontrados ocurre por primera vez en vocabulario, hecho por el cual no sólo se agrega al conjunto de tokens únicos, sino que también junto con él se almacena una marca de tiempo que indica la hora actual. Ahora bien, 6 horas después este token no ha vuelto a ocurrir en ninguno de los documentos ingeridos desde que fue incluido en el vocabulario y al contrastar su marca de tiempo con la hora actual se advierte que no supera el tiempo de vida definido, de manera que todavía puede persistir en el índice por 6 horas más. Si superado este tiempo, el token en cuestión no vuelve a ser encontrado en ninguno de los documentos arribados, entonces debe ser podado del índice. En el caso contrario, su marca de tiempo se actualiza con la hora en la cual ha vuelto a ocurrir y por lo tanto, su persistencia en el índice se extiende otras 12 horas.

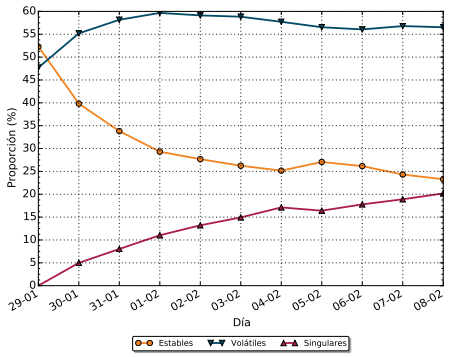


Figura 3.6: Proporción de cada tipo de token a través de los días luego de aplicar invalidador basado en un tiempo-de-vida igual 24 horas

La Figura 3.6 muestra la proporción de tokens conforme transcurren los días luego de aplicar un invalidador basado en un tiempo-de-vida igual a 24 horas. Al contrastar esta gráfica con 3.5(d) es posible advertir que los porcentajes han cambiado. El número de tokens clasificados como singulares en la primer figura decrece considerablemente, en particular se aprecia que el valor máximo correspondiente al último día resulta cercano al 20 %, mientras que en la segunda figura este supera el 55 %. Es de destacar que, si bien los

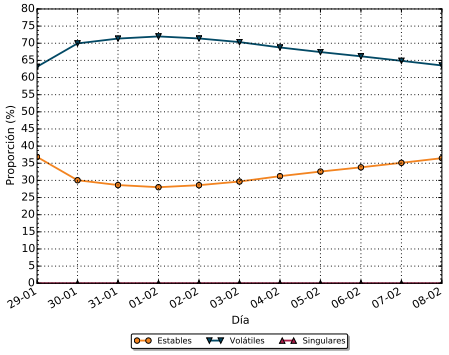
tokens estableces y volátiles presentan un comportamiento similar en ambos casos, la relación respecto del total resulta realmente mayor.

3.4.2. Basado en ventanas deslizantes

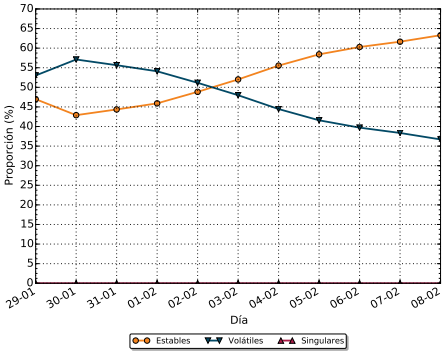
El segundo de los IEI emplea el enfoque de ventanas deslizantes definido al final de la sección anterior. Para su aplicación se requiere la recolección de estadísticas por una cantidad de días equivalente al tamaño de la ventana. Por otro lado, aquellos tokens caracterizados como singulares son directamente eliminados, mientras que para los volátiles se define un umbral de tolerancia entre 2 y $r - 1$, siendo r la cantidad de ranuras de la ventana. Este parámetro permite establecer en cierta medida la agresividad de la poda, pues valores más cercanos a r conllevan a la eliminación de un mayor número de tokens del índice. Es importante destacar que una vez que la ventana ha recorrido las r ranuras se dispone de la información necesaria para desarrollar el proceso de clasificación. Asimismo, este se dispara sólo si el token no ha ocurrido en el último día analizado desde su primera aparición.

Por ejemplo, considérese una ventana de 5 ranuras y un umbral de tolerancia igual a 4. Cuando un nuevo documento es ingerido, se procesa y se extraen sus términos. Adicionalmente, supóngase que uno de los tokens encontrados ocurre por primera vez en vocabulario, por lo cual se incorpora al mismo a la vez que una nueva instancia de la ventana le es asignada y la primer ranura activada. Transcurridos cinco días, el token ha formado parte de documentos procesados los días 4 y 5, además de que la ventana ha recorrido ya las 5 ranuras. Mientras las posiciones 1, 4 y 5 se encuentran activas, resto se encuentran en el estado opuesto. Si bien se cuenta con la información necesaria para que sea posible desarrollar el proceso de clasificación, el token ha ocurrido en el último día analizado y en consecuencia, la posible invalidación se pospone hasta tanto no se cumpla esta condición. Ahora bien, un día más ha transcurrido y de acuerdo al funcionamiento del esquema, la ventana vuelve a la primer ranura y esta se reinicia con el fin de activarse o no en relación a la información recuperada tal día. Suponiendo que el token no ha vuelto a ocurrir en ninguno de los documentos del día en cuestión y puesto que se cuenta con la información necesaria, es válido desarrollar el proceso de clasificación con el fin de establecer si debe persistir en el índice o no. De acuerdo al análisis, 2 de las 5 ranuras se encuentran activas y por lo tanto, es clasificado como volátil. Al comienzo del párrafo se estableció que el umbral de tolerancia era igual a 4, hecho por el cual sólo se admite la permanencia de token volátiles cuyas ranuras activas sumen un total de 4. En este caso, el token debe eliminarse pues su contabilización de ranuras activas no alcanza a satisfacer esta condición.

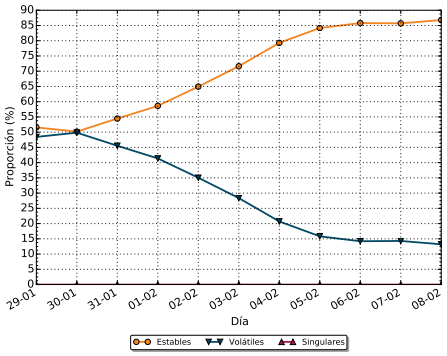
La Figura 3.7 permite apreciar la proporción de tokens conforme transcurren los días luego de aplicar un invalidador basado en un ventanas deslizantes con umbrales de tolerancia 2, 4 y 6. Nuevamente las diferen-



(a) Umbral de Tolerancia = 2



(b) Umbral de Tolerancia = 4



(c) Umbral de Tolerancia = 6

Figura 3.7: Proporción de cada tipo de token a través de los días luego de aplicar invalidador basado en ventanas deslizantes

cias observadas en relación a la Figura 3.5(d) son realmente notables. Como se advierte, el porcentaje de tokens estables se incrementa mientras que el de volátiles decrece en la medida en que la poda resulta cada vez más agresiva. En especial se destaca aquel valor alcanzado en el último día para el umbral 6, el cual se acerca al 85 %. En concordancia con el diseño del invalidador, los tokens singulares siempre se eliminan del índice invertido.

Capítulo 4

Experimentos y Resultados

I can't go to a restaurant and
order food because I keep looking
at the fonts on the menu.

Donald Knuth

4.1. Introducción

Una vez que los invalidadores de entradas de índices (Sección 3.4) han sido definidos conceptualmente, el siguiente paso corresponde a su implementación y evaluación. Para tal fin, se establece una metodología que permite automatizar la ejecución de los respectivos experimentos de manera que cada uno de ellos se concrete en igualdad de condiciones, a la vez que sean computadas aquellas métricas requeridas. Bajo este enfoque es posible plantear una comparación entre ellos, destacando los beneficios de su aplicación como así también sus limitaciones.

4.2. Metodología

El proceso de indexación y recuperación se lleva a cabo empleando Zambezi [7, 9, 10, 8], un motor de búsqueda diseñado para simular ambientes de búsqueda propios de sistemas de recuperación en tiempo real, pues no sólo la indexación se concreta en memoria sino que también todas aquellas estructuras necesarias para la búsqueda se mantienen allí durante la ejecución. Con el fin de lograr su integración con los IEI propuestos fue necesario modificar el código original[1].

Para emular la operación normal de un motor de búsqueda conforme transcurren los días y nuevos documentos son ingeridos, se genera una serie

[1] Página Principal: `http://nasadi.github.io/Zambezi/`

de índices de acuerdo a los documentos pertenecientes a cada uno de los 17 días de la muestra. Es decir, dado el día n y su correspondiente conjunto de documentos, el día $n+1$ comprende aquellos tweets pertenecientes al día anterior más los incorporados en el día en cuestión. En este sentido, se cuenta con 17 índices diferentes, donde cada uno se genera de forma incremental en la medida en que se suceden los días de la muestra. La misma metodología es implementada para el caso de los índices obtenidos a partir de la aplicación de cada invalidador.

Métricas: Con el objeto de evaluar el rendimiento alcanzado por el enfoque propuesto, se cuantifica tanto la eficiencia (tiempo y espacio) y la efectividad. En el primer caso, el tiempo de ejecución se mide en términos del tiempo de reloj o *wall-clock time* para cada consulta. Asimismo, se evalúa el número de entradas invalidadas diariamente junto con la reducción del tamaño del índice. Para determinar la efectividad, se computa la proporción de documentos recuperados en común con el índice sin poda (*baseline*) luego de resolver cada consulta (intersección de ambos conjuntos). Recuérdese que en un escenario de búsqueda en tiempo real una de las principales tareas de búsqueda consiste en presentar los documentos más recientes relacionados a la consulta [57] (*frescura* de los resultados, Sección 1.3.1). Para lograrlo, los tweets recuperados son enseñados al usuario en orden cronológico inverso.

Consultas: Debido a la falta de sets de consultas para la evaluación de sistemas de búsqueda en tiempo real públicamente disponibles, varios trabajos deciden generar logs sintéticos, como [61, 24]. En el presente trabajo, el set empleado constituye un subconjunto de un millón de consultas extraídas del bien conocido AOL Query Log [65] conformado en su completitud de aproximadamente 20 millones de consultas y utilizado en otros trabajos que involucran búsqueda en tiempos real [52, 10, 8]. Sólo se consideran aquellas consultas de longitud (cantidad de términos) menores o iguales a 4, verificando para cada dimensión que se mantuviera en el subconjunto la distribución de frecuencia del set original (Figura 4.1), como así también que los términos formen parte del vocabulario.

Invalidadores: En el caso del IEI basado en el tiempo de vida de las entradas (IEI-TTL), se establece un valor de TTL igual a 24 horas con el fin de verificar si una entrada del índice ha expirado. Aprovechando las marcas de tiempos de los tweets es posible establecer el comienzo y el final de los días en términos de horas. Así cada 24 horas durante las dos semanas abarcadas por la muestra, el índice es recorrido y las correspondientes entradas desalojadas de acuerdo a la heurística definida. Como fue mencionado, el segundo invalidador, aquel basado en el método de ventanas deslizantes (IEI-SW), cuenta con la posibilidad de establecer un umbral de tolerancia para los tokens clasi-

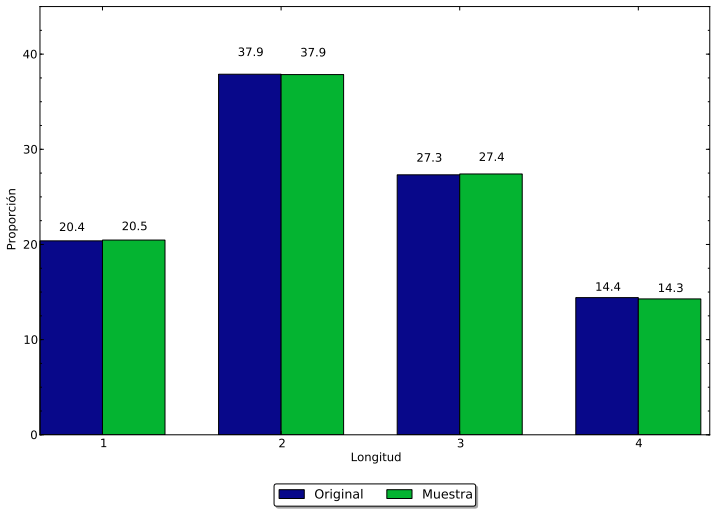


Figura 4.1: Distribución de frecuencias de la muestra obtenida del log de consultas de AOL

ficados como volátiles, permitiendo de alguna manera definir la agresividad de la poda. Para los experimentos, se emplean tres umbrales ($\{2, 4, 6\}$).

De esta manera, para cada día se procesa 1 millón de consultas y se cuantifica tanto eficiencia como efectividad comparando aquellos resultados recuperados bajo el índice original sin poda alguna (baseline) con aquellos obtenidos empleando los índices podados a partir de la aplicación de cada invalidador respectivamente. Se ejecutan tres series de experimentos recuperando los top-k documentos que resultan de mayor relevancia a la consulta (con $k = \{10, 50, 100\}$). Zambezi es configurado para emplear el algoritmo de resolución de consultas WAND (Sección 2.4.3) de tipo disyuntivo (OR) con una adaptación para microblogging, básicamente utiliza las marcas de tiempo como criterio de ordenamiento junto con una versión simplificada de la función de puntaje (o *scoring*) BM25 [14].

4.3. Resultados

En la presente sección se exhiben los resultados obtenidos tras la evaluación de efectividad y eficiencia concretada a partir de la metodología establecida anteriormente. Cada invalidador es analizado por separado destacando aquellas particularidades observadas en cada caso.

4.3.1. Basado en tiempo-de-vida

La evaluación de efectividad muestra que el invalidador basado en la estrategia tiempo-de-vida no degrada los resultados de forma significativa.

La Figura 4.2 enseña la proporción de resultados en común con el baseline (promediado de 1 un millón de consultas) para las tres series de experimentos. En el caso de la recuperación de los top-10, menos de un documento (en promedio) no se encuentra en el conjunto de resultados obtenidos con el índice podado. Tal comportamiento es proporcionalmente similar en las series restantes (top-50 y top-100). Un análisis más profundo de los tokens podados revela que en su gran mayoría resultan casos atípicos, los cuales escasamente ocurren en las consultas. Complementariamente, la Figura 4.3 muestra como se distribuyen las consultas en relación a la efectividad obtenida. Para ello, se define un conjunto de intervalos de ancho igual a 10, comprendidos entre 0 y 100, donde aquel del extremo inferior implica que de los k recuperados ninguno coincide con aquellos del baseline, mientras que el intervalo más a la derecha constituye una intersección igual a k. A efectos de una mejor visualización el eje de las ordenadas se encuentra en escala logarítmica. Como se aprecia, existen consultas para las cuales se obtienen diferentes proporciones de intersección respecto del baseline (eje de las abscisas). Aun así, para la mayoría del conjunto (superior al 75 % en las tres series) la máxima efectividad es alcanzada, siendo que en el resto de los intervalos la proporción resulta inferior o muy cercano al 1 % (10^0). Si bien esta figura corresponde al último día de la muestra, un comportamiento similar se observa en aquellos precedentes.

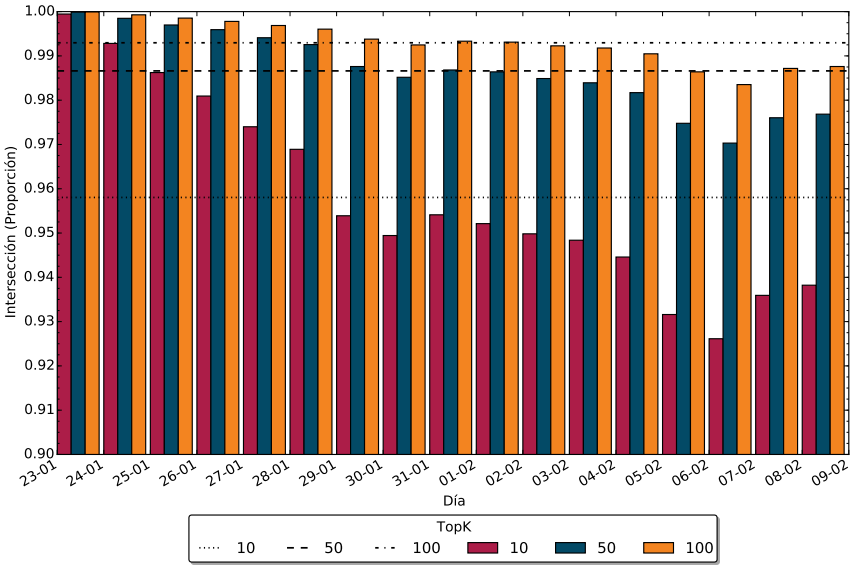


Figura 4.2: Proporción de la intersección cuando se recuperan los top 10, 50 y 100 documentos para IEI-TTL. Las líneas punteadas corresponden a los valores promedios de cada serie

En términos de eficiencia es posible advertir que bajo la aplicación del

invalidador se obtiene una reducción del tiempo total de ejecución para todas las configuraciones de hasta un 6 % en el mejor caso. Estos resultados pueden observarse en la Figura 4.4. El hecho de que el tiempo de ejecución se incremente conforme transcurren los días se atribuye al crecimiento del vocabulario. Asimismo, la diferencia también crece en la medida en que más tokens son adicionados al baseline y en consecuencia, un mayor número de entradas son invalidadas y desalojadas. Complementariamente, la Figura 4.5 permite analizar la eficiencia alcanzada desde otra perspectiva. Esta gráfica bosqueja la distribución de frecuencias del tiempo de ejecución de las consultas del último día de la muestra. Mientras el eje de las abscisas representa los diferentes intervalos, siendo que cada uno posee una duración de 0,05 milisegundos, el eje de las ordenadas corresponde a la frecuencia porcentual, *i.e.*, cantidad de consultas sobre el total. De esta manera, cuánto más a la izquierda se ubique cada barra y menor resulten aquellas localizadas en los intervalos posteriores, mayor es la efectividad pues los intervalos más cercanos a cero representan el menor tiempo de ejecución. Si se observa, a lo largo de las tres series el invalidador posee un mayor número de consultas en el primer intervalo, aquel que se ubica entre 0 y 0,05 milisegundos, en comparación con el baseline, siendo el principal objetivo del invalidador trasladar aquellas consultas que consumen un mayor tiempo de procesamiento a intervalos de menor tiempo. A continuación, el espacio consumido por el índice invertido en las versiones baseline como podado es analizado. La

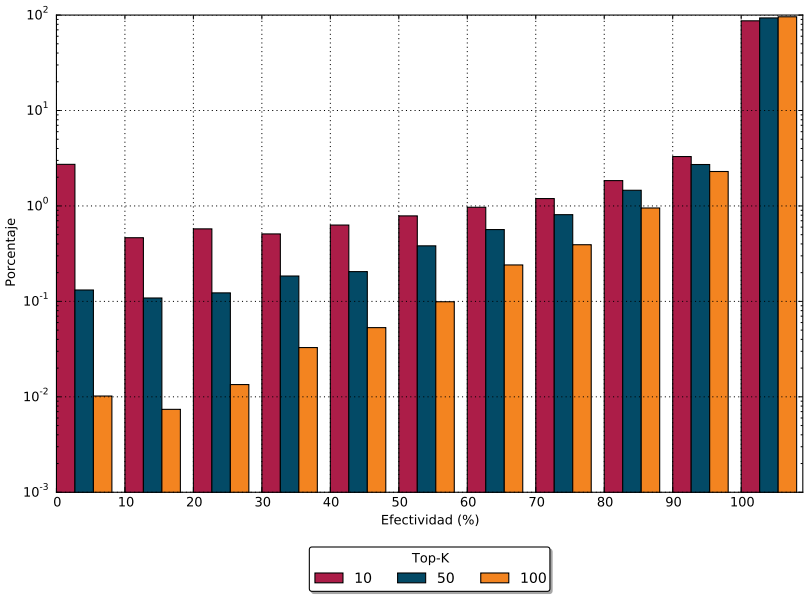


Figura 4.3: Distribución de frecuencias de la intersección de resultados cuando se recuperan los top 10, 50 y 100 documentos correspondientes al último día de la muestra para IEI-TTL

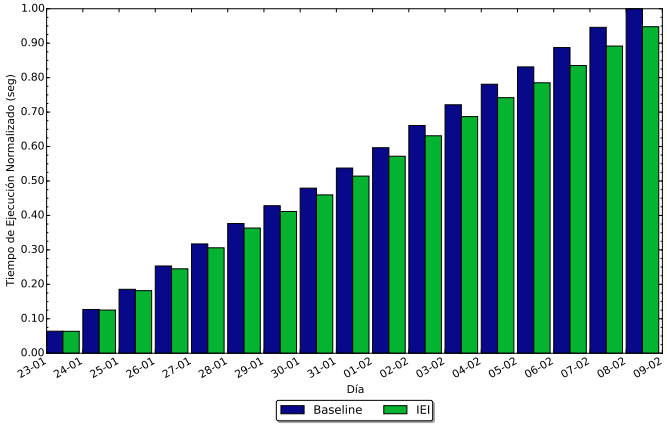
Tabla 4.1 muestra el número de entradas en el vocabulario. La cantidad de entradas válidas decrece considerablemente a lo largo de los días (hasta un 88 %), admitiendo agilizar las búsquedas sobre el vocabulario. Sin embargo, el número total de identificadores de documentos (DocIDs) contabilizados para las listas de posteo resultantes disminuye con menor velocidad (Tabla 4.2), hasta un 9.1 % en el último día.

Tabla 4.1: Número de entradas en el índice invertido

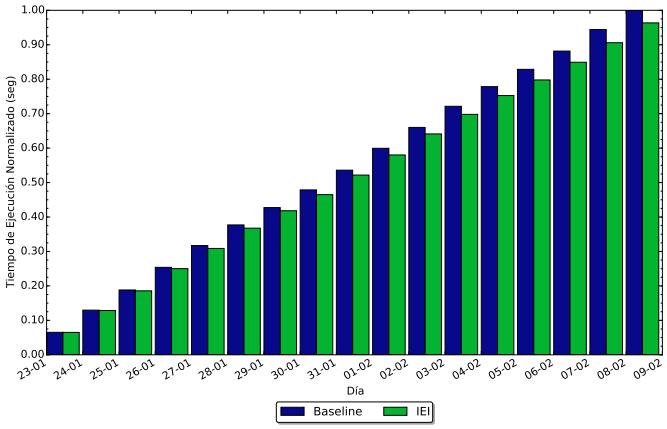
Día	Baseline	Podado	Dif. %
1	217,044	209,379	3.53
2	390,863	231,976	40.65
3	540,456	238,330	55.90
4	692,677	259,240	62.57
5	833,499	250,156	69.99
6	970,939	256,852	73.55
7	1,081,271	223,664	79.31
8	1,186,153	228,312	80.75
9	1,295,470	239,079	81.54
10	1,407,580	258,560	81.63
11	1,519,236	254,509	83.25
12	1,627,904	258,206	84.14
13	1,736,197	259,738	85.04
14	1,827,029	227,970	87.52
15	1,916,499	230,474	87.97
16	2,012,312	245,609	87.79
17	2,105,809	249,280	88.16

Tabla 4.2: Suma de los DocIDs en las postings de todos los tokens

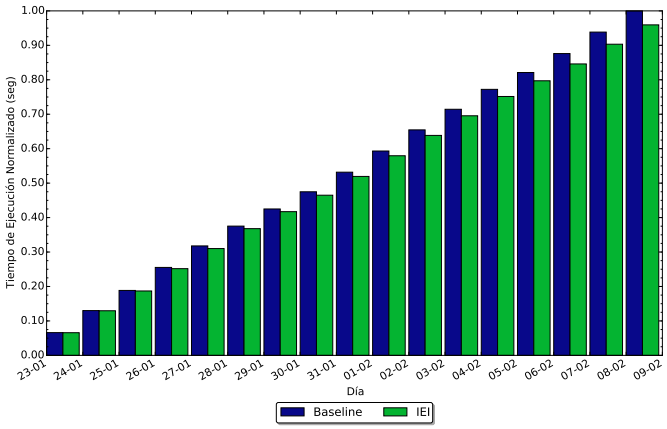
Día	Baseline	Podado	Dif. %
1	2,392,099	2,384,262	0.33
2	5,174,584	4,992,039	3.53
3	7,864,478	7,467,306	5.05
4	10,919,587	10,294,900	5.72
5	13,892,852	12,991,206	6.49
6	16,859,239	15,678,477	7.00
7	19,413,681	17,914,817	7.72
8	22,037,617	20,270,901	8.02
9	24,934,503	22,903,479	8.15
10	27,976,632	25,684,408	8.19
11	31,168,827	28,576,005	8.32
12	34,215,991	31,326,117	8.45
13	37,198,345	33,998,347	8.60
14	39,793,764	36,240,415	8.93
15	42,532,978	38,669,507	9.08
16	45,659,362	41,508,202	9.09
17	48,608,362	44,153,772	9.16



(a) Resultados Top-10

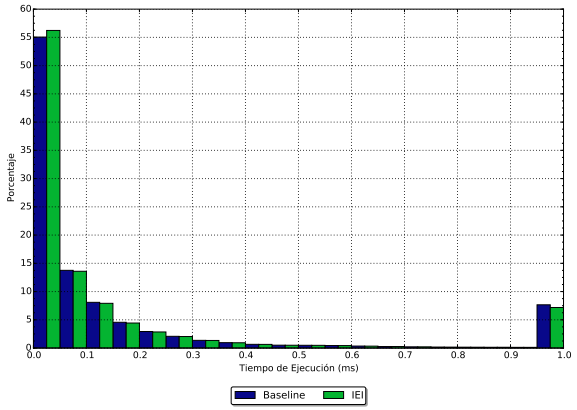


(b) Resultados Top-50

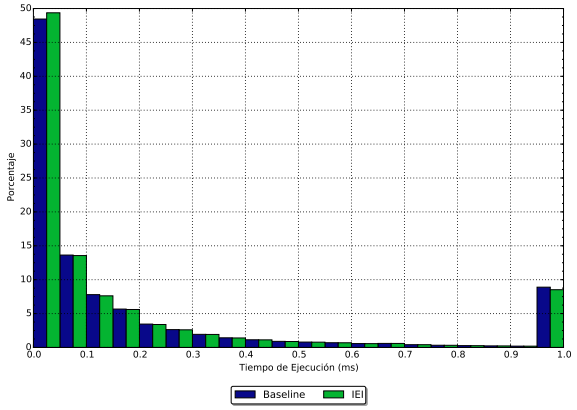


(c) Resultados Top-100

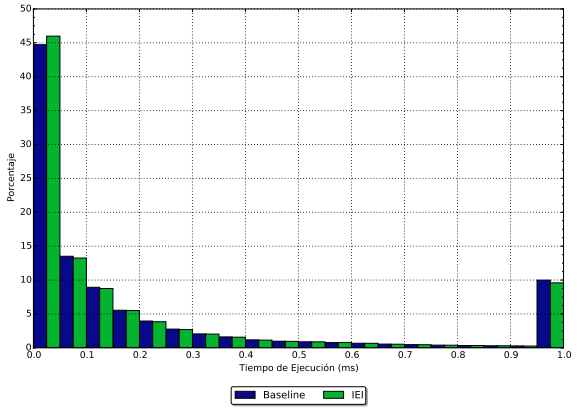
Figura 4.4: Tiempo de ejecución normalizado para 1.000.000 consultas correspondiente a IEI-TTL



(a) Top-10



(b) Top-50



(c) Top-100

Figura 4.5: Distribución de frecuencias del tiempo de ejecución de las consultas correspondientes al último día de la muestra para IEI-TTL

4.3.2. Basado en ventanas deslizantes

El primer invalidador basado en ventanas deslizantes evaluado corresponde a aquel con umbral 2. Como muestra la Figura 4.6, al igual que el invalidador anterior, la efectividad no se ve prácticamente deteriorada. La Figura 4.7 enseña el análisis complementario de efectividad, apreciándose un comportamiento similar al previamente descripto. Sin embargo, la eficiencia obtenida no resulta prometedora si se observa la Figura 4.8. Incluso, es posible observar que exceptuando de los últimos tres días de los top-10 recuperados la evaluación de consultas empleando el índice podado insume ligeramente mayor tiempo en las tres series. Aún en los casos destacados, el máximo beneficio que se obtiene concierne a un 1,18 %. Si bien al observar la Figura 4.9 es posible advertir que para el primer intervalo la frecuencia porcentual del invalidador resulta mayor a la del baseline, también es importante que el resto de las consultas se distribuyan en los intervalos más cercanos a cero; pues una determinada cantidad de consultas en los intervalos próximos a uno implica un deterioro en el rendimiento. Analizando el volumen de entradas remanentes en el vocabulario (Tabla 4.3) se observa que el mismo decrece hasta un 67.75 %. Asimismo, el número total de DocIDs correspondientes a las listas de posteo conservadas tras la poda, alcanza un 5.94 %. Si bien las proporciones resultan diversas, el comportamiento es similar al descripto para el invalidador basado en tiempo de vida.

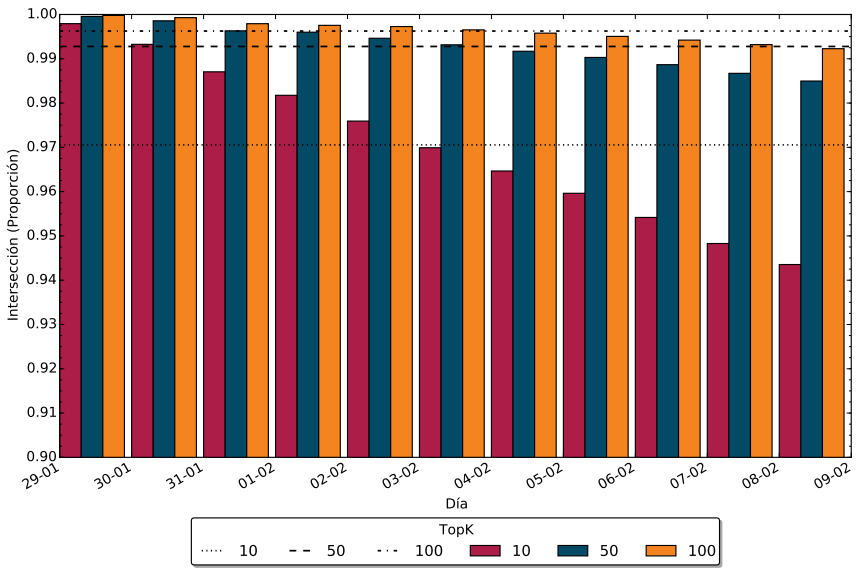


Figura 4.6: Proporción de la intersección cuando se recuperan los top 10, 50 y 100 documentos para IEI-SW2. Las líneas punteadas corresponden a los valores promedios de cada serie

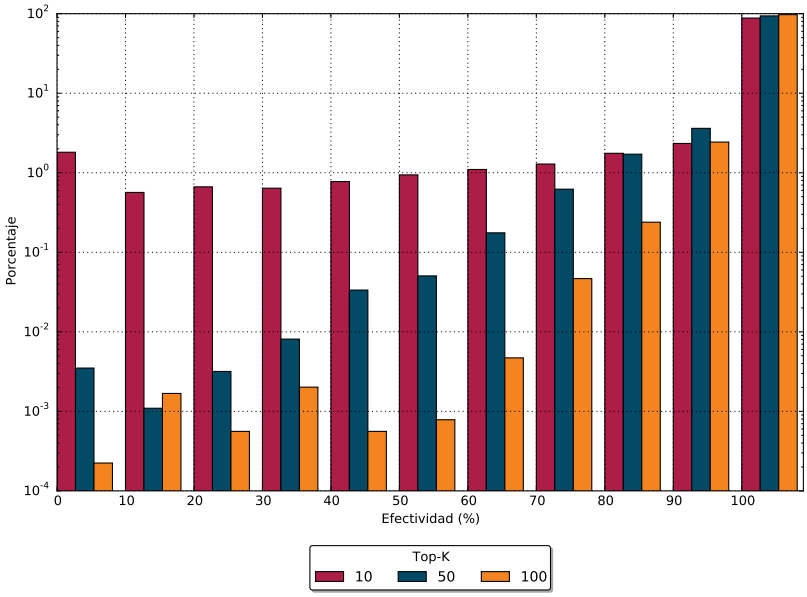


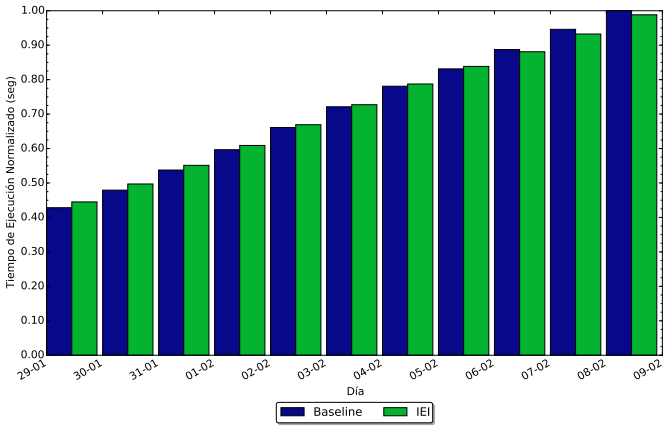
Figura 4.7: Distribución de frecuencias de la intersección de resultados cuando se recuperan los top 10, 50 y 100 documentos correspondientes al último día de la muestra para IEI-SW2

Tabla 4.3: Número de entradas en el índice invertido

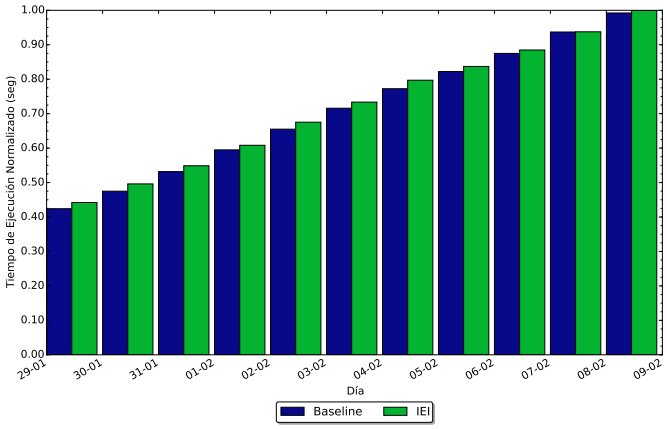
Día	Baseline	Podado	Dif. %
7	1,081,271	953,503	11.82
8	1,186,153	906,235	23.60
9	1,295,470	872,188	32.67
10	1,407,580	835,328	40.66
11	1,519,236	803,664	47.10
12	1,627,904	771,108	52.63
13	1,736,197	763,194	56.04
14	1,827,029	744,523	59.25
15	1,916,499	721,065	62.38
16	2,012,312	701,094	65.16
17	2,105,809	679,049	67.75

Tabla 4.4: Suma de los DocIDs en las postings de todos los tokens

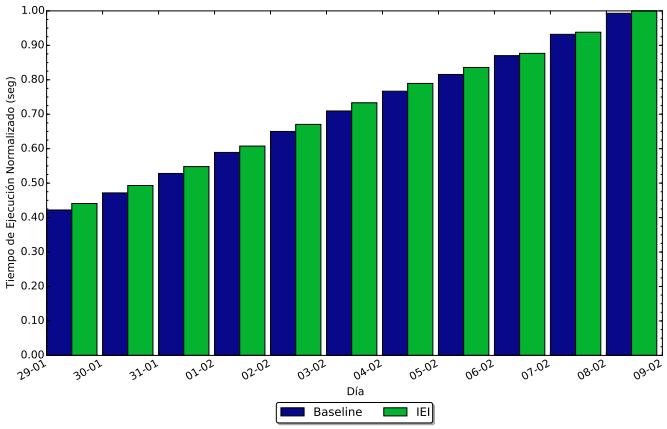
Día	Baseline	Podado	Dif. %
7	19,413,681	19,243,999	0.87
8	22,037,617	21,622,428	1.88
9	24,934,503	24,265,782	2.68
10	27,976,632	27,034,488	3.37
11	31,168,827	29,942,806	3.93
12	34,215,991	32,699,058	4.43
13	37,198,345	35,408,680	4.81
14	39,793,764	37,749,025	5.14
16	42,532,978	40,221,086	5.44
16	45,659,362	43,060,140	5.69
17	48,608,362	45,720,268	5.94



(a) Resultados Top-10

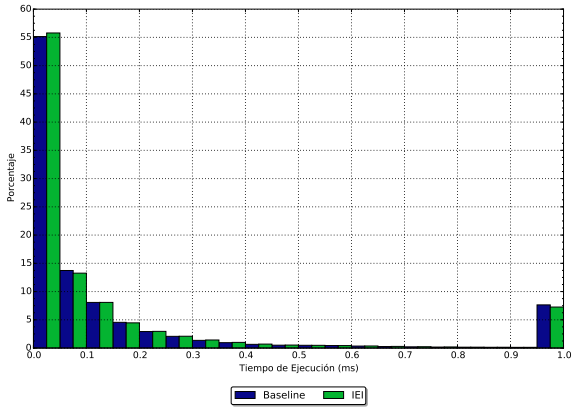


(b) Resultados Top-50

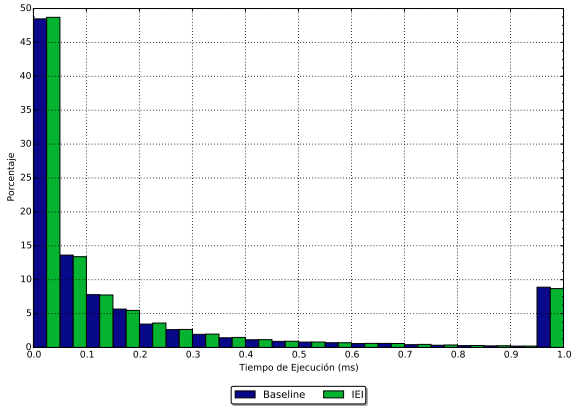


(c) Resultados Top-100

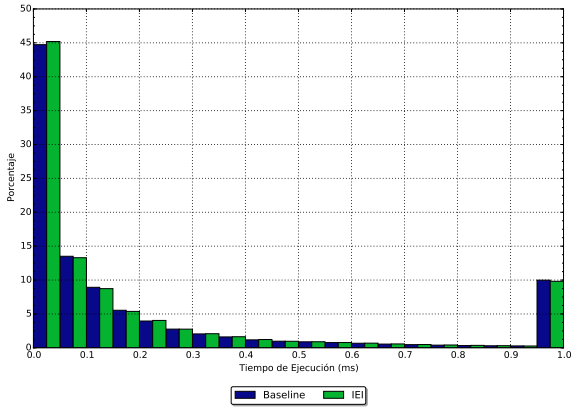
Figura 4.8: Tiempo de ejecución normalizado para 1.000.000 consultas correspondiente a IEI-SW2



(a) Top-10



(b) Top-50



(c) Top-100

Figura 4.9: Distribución de frecuencias del tiempo de ejecución de las consultas correspondientes al último día de la muestra para IEI-SW2

En el caso del invalidador basado en ventanas deslizantes con umbral 4, es posible observar en la Figura 4.12 un incremento en términos de eficiencia en comparación con el invalidador previamente evaluado. Si bien en primera instancia el comportamiento no resulta fructuoso, pues el baseline presenta mejor desempeño, en la medida en se suceden los días se contabiliza para todas las series una reducción del tiempo total de ejecución, de hasta un 6,41 % en el mejor caso. La Figura 4.9 muestra los resultados obtenidos tras analizar la frecuencia para el último día de la muestra. Por otro lado, la evaluación de efectividad muestra que el enfoque aplicado no afecta considerablemente los resultados obtenidos, ya que en el caso de la recuperación de los top-10, un documento de cada diez (en promedio) no se encuentra en el ranking final. Las series restantes exhiben un comportamiento proporcionalmente similar (top-50 y top-100). La Figura 4.10 enseña estos resultados, mientras que la Figura 4.11 corresponde al análisis complementario. El estudio del espacio consumido muestra nuevamente una disminución en el número de entradas remanentes en el vocabulario conforme transcurren los días (hasta un 69.7 %), mientras que en el caso de la cantidad total de DocIDs se advierte un reducción de hasta un 7.26 % en el último día. Las Tablas 4.5 y 4.6 presentan estos resultados, respectivamente.

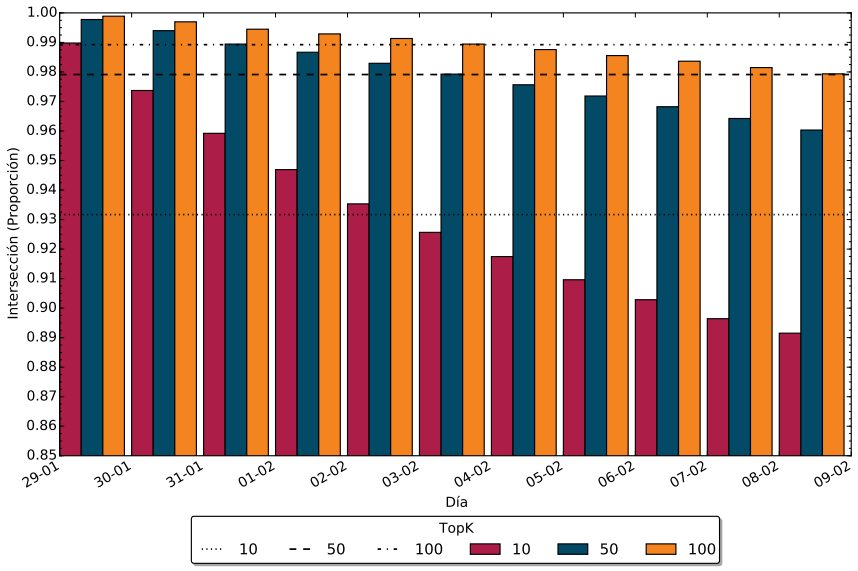


Figura 4.10: Proporción de la intersección cuando se recuperan los top 10, 50 y 100 documentos para IEI-SW4. Las líneas punteadas corresponden a los valores promedios de cada serie

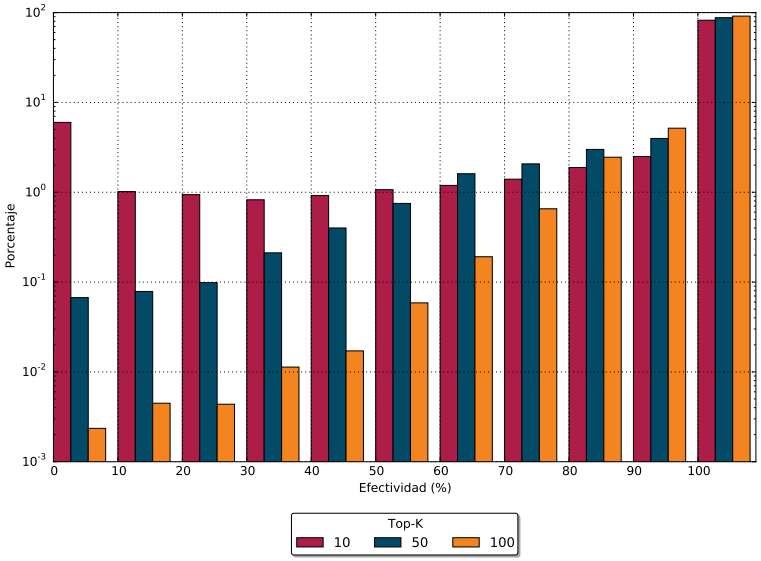


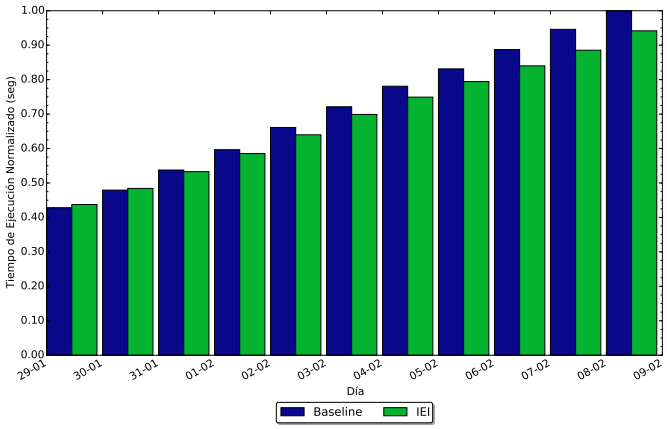
Figura 4.11: Distribución de frecuencias de la intersección de resultados cuando se recuperan los top 10, 50 y 100 documentos correspondientes al último día de la muestra para IEI-SW4

Tabla 4.5: Número de entradas en el índice invertido

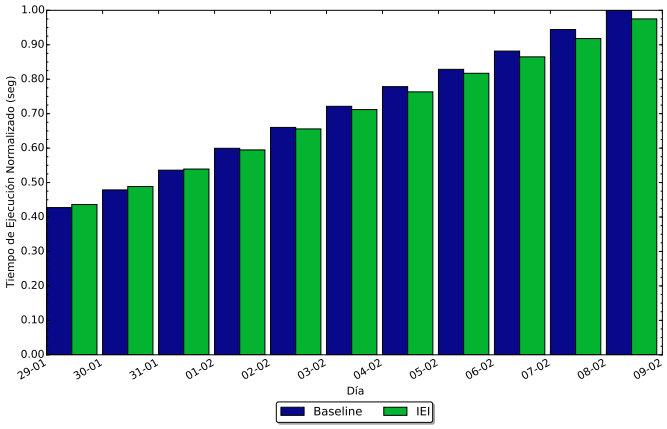
Día	Baseline	Podado	Dif. %
7	1,081,271	934,260	13.60
8	1,186,153	873,004	26.40
9	1,295,470	830,676	35.88
10	1,407,580	788,382	43.99
11	1,519,236	754,785	50.32
12	1,627,904	722,069	55.64
13	1,736,197	715,801	58.77
14	1,827,029	698,237	61.78
15	1,916,499	676,089	64.72
16	2,012,312	658,136	67.29
17	2,105,809	638,118	69.70

Tabla 4.6: Suma de los DocIDs en las postings de todos los tokens

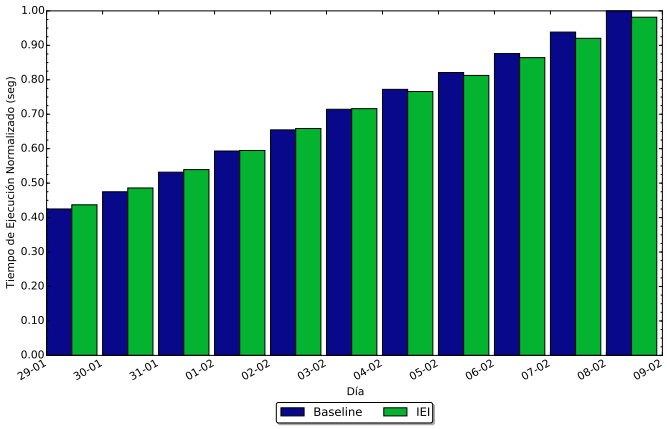
Día	Baseline	Podado	Dif. %
7	19,413,681	19,152,954	1.34
8	22,037,617	21,432,916	2.74
9	24,934,503	23,990,685	3.79
10	27,976,632	26,685,536	4.61
11	31,168,827	29,534,438	5.24
12	34,215,991	32,239,236	5.78
13	37,198,345	34,905,933	6.16
14	39,793,764	37,209,193	6.49
16	42,532,978	39,643,331	6.79
16	45,659,362	42,447,901	7.03
17	48,608,362	45,077,120	7.26



(a) Resultados Top-10

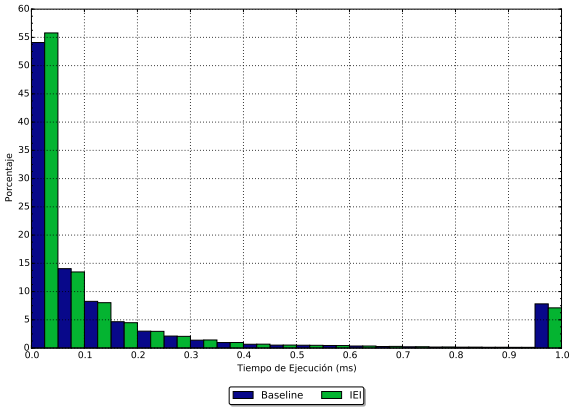


(b) Resultados Top-50

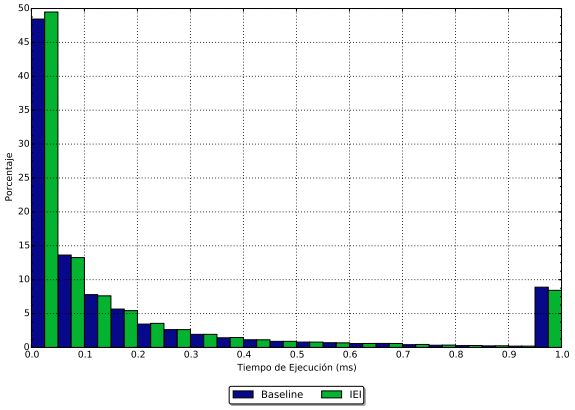


(c) Resultados Top-100

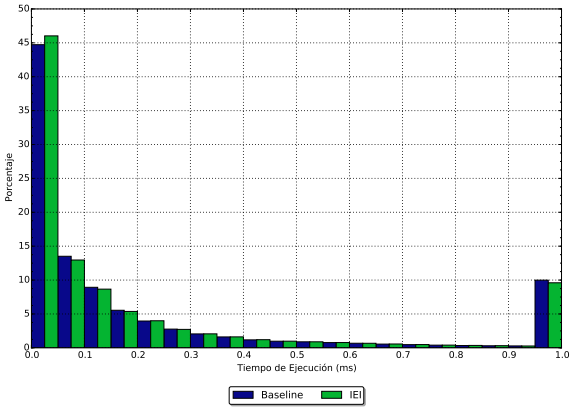
Figura 4.12: Tiempo de ejecución normalizado para 1.000.000 consultas correspondiente a IEI-SW4



(a) Top-10



(b) Top-50



(c) Top-100

Figura 4.13: Distribución de frecuencias del tiempo de ejecución de las consultas correspondientes al último día de la muestra para IEI-SW4

Finalmente, se evalúa el desempeño del invalidador basado en ventanas deslizantes con umbral 6, correspondiendo este a la poda más agresiva. El análisis de efectividad muestra que en el caso de la recuperación de los top-10, al igual que el invalidador estudiado anteriormente, un documento (en promedio) no forma parte del conjunto de resultados obtenido. En las configuraciones restantes, top-50 y top-100, se advierten la ausencia de 5 y 3 documentos, respectivamente. Tales observaciones corresponden a la Figura 4.14, además de la Figura 4.15 donde se puede observar la evaluación complementaria de efectividad. Esta última permite apreciar que la mayoría de las consultas (superior al 75 %) en las tres series se conglomeran en aquel intervalo correspondiente a la máxima efectividad. A diferencia de los invalidadores con umbrales 2 y 4 es posible notar un considerable incremento de la eficiencia, incluso superior al enfoque basado en el tiempo-de-vida. En el mejor de los casos, este se aproxima a un 10 % como lo exhibe la Figura 4.16. En referencia a la Figura 4.17 puede observarse que la cantidad de consultas localizadas en el primer intervalo y en último resulta presentan un diferencia a destacar al contrastar aquellas reunidas por el baseline y el invalidador. En concordancia con aquello mencionado previamente, mientras en el primer caso resulta conveniente que la frecuencia sea mayor, en el segundo que se espera el comportamiento opuesto. Por otro lado, al comparar la presente configuración con la estrategia basada en TTL, se advierte que la Proporción de entradas válidas en el vocabulario resultante es menor, pues en este caso la diferencia es igual a un 70.54 % para el último día (Tabla 4.7), siendo que para la disminución en el número total de DocIDs corresponde a un 8.70 % (Figura 4.8).

Tabla 4.7: Número de entradas en el índice invertido

Día	Baseline	Podado	Dif. %
7	1,081,271	928,040	14.17
8	1,186,153	861,738	27.35
9	1,295,470	816,633	36.96
10	1,407,580	772,451	45.12
11	1,519,236	737,668	51.44
12	1,627,904	704,236	56.74
13	1,736,197	697,567	59.82
14	1,827,029	679,515	62.81
15	1,916,499	657,298	65.70
16	2,012,312	639,921	68.20
17	2,105,809	620,422	70.54

Tabla 4.8: Suma de los DocIDs en las postings de todos los tokens

Día	Baseline	Podado	Dif. %
7	19,413,681	19,080,321	1.72
8	22,037,617	21,278,217	3.45
9	24,934,503	23,771,701	4.66
10	27,976,632	26,400,001	5.64
11	31,168,827	29,181,683	6.38
12	34,215,991	31,825,493	6.99
13	37,198,345	34,435,359	7.43
14	39,793,764	36,671,148	7.85
16	42,532,978	39,040,825	8.21
16	45,659,362	41,800,241	8.45
17	48,608,362	44,380,423	8.70

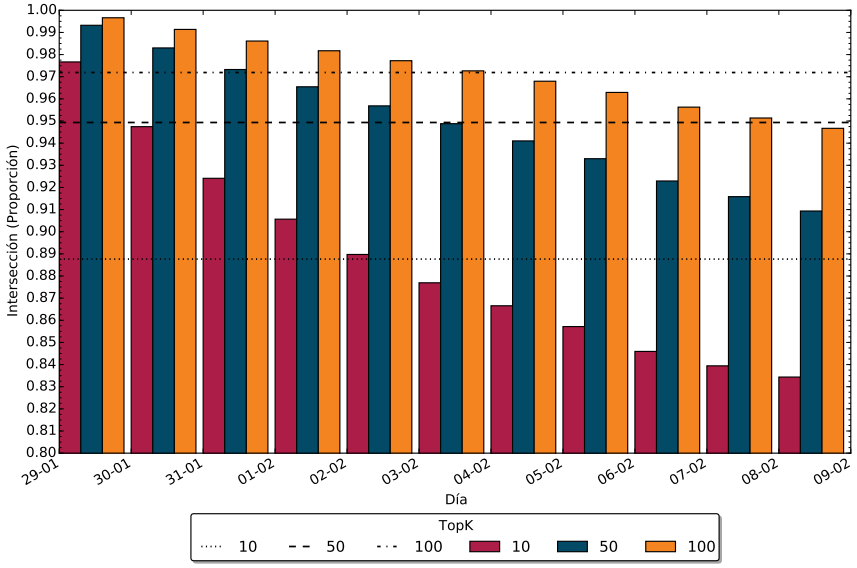


Figura 4.14: Proporción de la intersección cuando se recuperan los top 10, 50 y 100 documentos para IEI-SW6. Las líneas punteadas corresponden a los valores promedios de cada serie

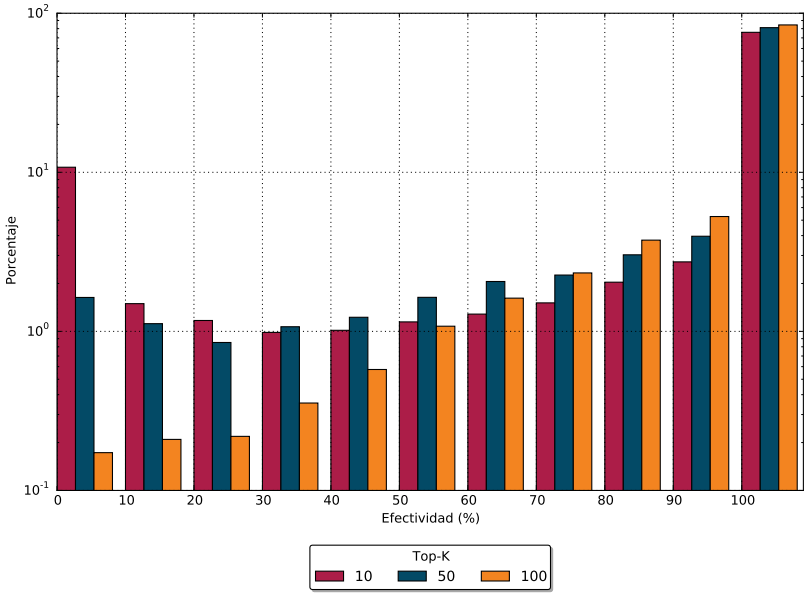
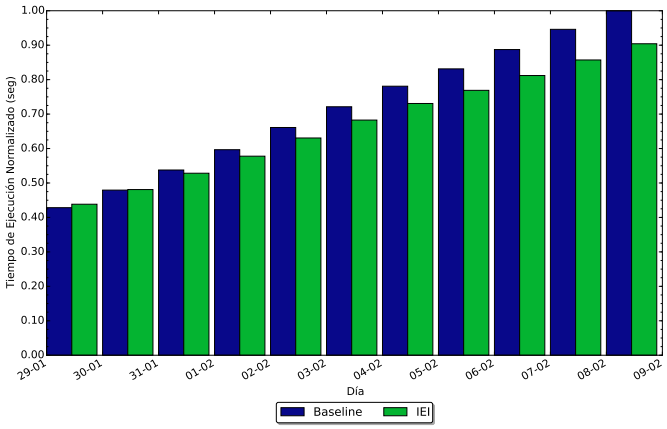
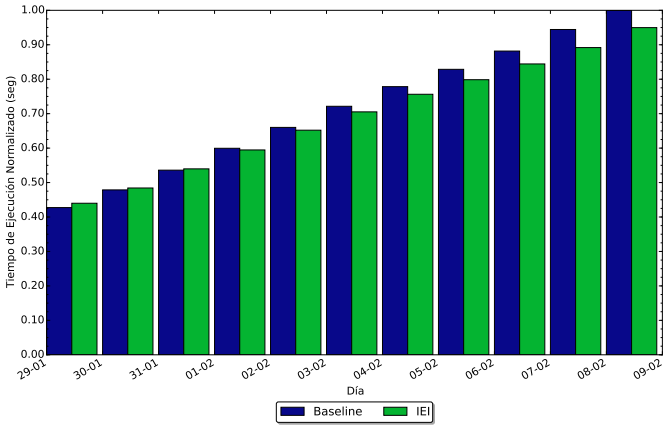


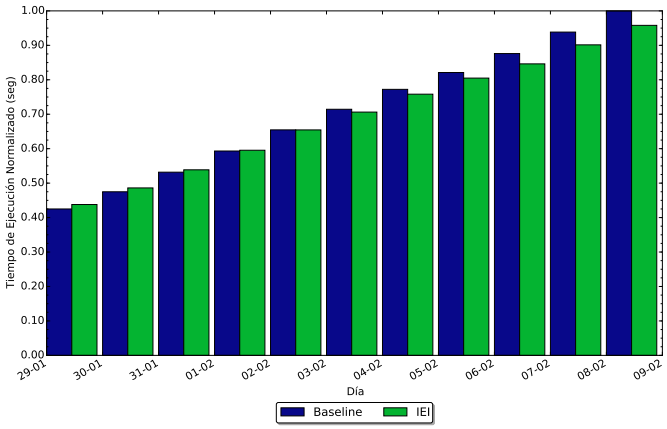
Figura 4.15: Distribución de frecuencias de la intersección de resultados cuando se recuperan los top 10, 50 y 100 documentos correspondientes al último día de la muestra para IEI-SW6



(a) Resultados Top-10

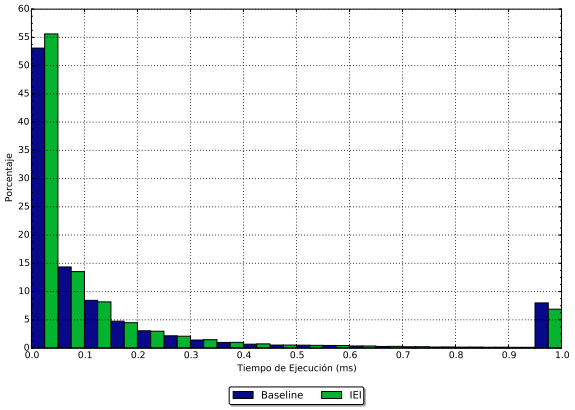


(b) Resultados Top-50

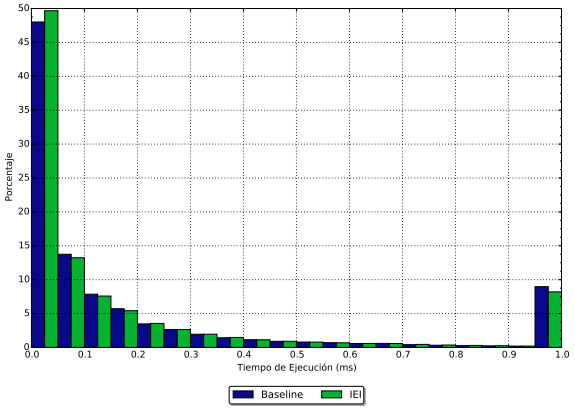


(c) Resultados Top-100

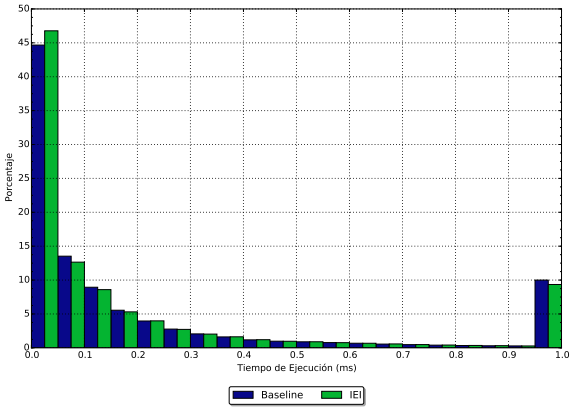
Figura 4.16: Tiempo de ejecución normalizado para 1.000.000 consultas correspondiente a IEI-SW6



(a) Top-10



(b) Top-50



(c) Top-100

Figura 4.17: Distribución de frecuencias del tiempo de ejecución de las consultas correspondientes al último día de la muestra para IEI-SW6

Una cuestión importante a destacar surgida de los análisis hasta aquí planteados, subyace en el constante crecimiento del vocabulario. A pesar de la aplicación de los invalidadores este aún no puede disminuirse y por ello, es posible advertir que una estrategia de poda más agresiva es necesaria para complementar al IEI, posibilitando de alguna manera controlar este crecimiento, en particular una poda al nivel de las listas de posteo puede resultar adecuada. Tal cuestión se ve reforzada si se observa el número total de DocIDs contabilizados para las listas de posteo remanentes, donde en el mejor de los casos la diferencia con el baseline es de un 9.16 %. El hecho de plantear la eliminación de ciertas entradas considerando las longitudes de las postings de los tokens que restan en el vocabulario, puede permitir alcanzar mayores beneficios.

Capítulo 5

Conclusiones y Trabajos Futuros

Your time is limited, so don't waste
it living someone else's life.
Don't be trapped by dogma, which is
living with the results of other
people's thinking.
Don't let the noise of others' opinions
drown out your own inner voice.
And most important, have the courage
to follow your heart and intuition.

Steve Jobs

5.1. Conclusiones

El presente trabajo es el resultado de un proceso cuyas etapas han dejado en mi una enseñanza, ya sea desde el punto de vista meramente académico como de aquel relacionado al futuro ejercicio de mi vida profesional. Considero que el camino recorrido ha implicado un verdadero reto y una prueba no sólo al esfuerzo y la dedicación, sino también a la vocación. Desde la mirada personal, sostengo que se trata de una apuesta donde si bien al comenzar las posibilidades son abundantes no resultan suficientes para determinar con certeza como será la conclusión. Es en este punto donde la figura del director cobra sentido y a quien agradezco profundamente por ayudarme a disipar aquellas dudas y flaquezas que pudieran desviarme de mi objetivo principal e incluso desistir. He descubierto que la principal motivación no subyace simplemente en cumplimentar aquellos objetivos expuestos al comienzo, sino que radica en algo más grande y que desborda el marco comprendido por este trabajo: la curiosidad. El hecho de no conocer o no comprender es lo que me ha llevado cada día a desear investigar, descubrir y aprender el por

qué.

Abocándose particularmente a la elaboración y desarrollo de este trabajo, siento que he realizado un gran avance respecto del conocimiento concerniente a los tópicos abordados. Si bien estos sólo representan una pequeña fracción de una extensa área de estudio como lo es la Recuperación de Información, la experticia recolectada ha sido abundante y ha implicado un verdadero crecimiento desde el comienzo. Asimismo, he comprendido aquellos factores que deben considerarse al llevar a cabo una investigación sobre una temática en particular, al igual que la metodología (o al menos los pasos esenciales) que tal proceso requiere. Un ejemplo válido subyace en no asumir como una verdad absoluta todo aquello que se encuentra en la bibliografía. He aprendido que es bueno cuestionares en ciertas ocasiones el valor de aplicación de lo que está escrito, sobre todo cuando ciertos detalles fundamentales son omitidos, muchas veces por conveniencia y otras porque se asume que forman parte del conocimiento del lector. En algunas oportunidades he podido resolver estas cuestiones consultando a los autores (o a referentes de ellos), aunque no ha sido siempre el caso.

En el presente trabajo se lleva a cabo estudio de la dinámica que caracteriza a las colecciones propias del área de microblogging, empleando para ello un dataset real aportado por la comunidad científica. Es posible determinar que el vocabulario crece más rápido que las predicciones que la ley de Heaps establece, exhibiendo más bien una tendencia lineal. Complementariamente, se examina cómo la frecuencia de los tokens que conforman el vocabulario evoluciona conforme transcurren los días, estableciendo tres grupos que permitan conglomerar aquellos con comportamientos equivalentes. Finalmente, se introduce el concepto de invalidadores de entradas de índices (IEI), un enfoque que apunta a invalidar y desalojar selectivamente aquellas entradas del índice invertido cuya ausencia no degrade considerablemente la efectividad en la recuperación. En consecuencia, el índice se vuelve más pequeño y puede aumentar la eficiencia general.

Los experimentos realizados muestran que las estrategias propuestas permiten reducir el número de entradas en el vocabulario hasta un 88 %, con una disminución del tiempo de ejecución total de un 6 % en el caso del IEI basado en tiempo-de-vida y hasta un 71 % con una eficiencia del 10 % en el caso de aquel invalidador basado en ventanas deslizantes. Sin embargo, el tamaño del índice resultante decrece más lentamente, hasta un 9,1 % para IEI-TTL y un 8,7 % para IEI-SW. Si bien el vocabulario disminuye su cardinalidad pues gran cantidad de entradas son removidas, no es suficiente. Principalmente, la razón subyace en que las listas de posteo podadas no resultan considerablemente extensas en comparación con aquellas que han persistido, esto puede observarse en la relación existente entre el número de tokens eliminados y la proporción de DocIDs remanentes. Es por tal razón que se necesita complementar a los IEI con otra estrategia, en especial plantear la eliminación

de ciertas entradas considerando las longitudes de las listas de posteo de los tokens restantes, puede permitir alcanzar mayores beneficios, como por ejemplo controlar en cierta medida el crecimiento del vocabulario.

5.2. Trabajos Futuros

Como línea de investigación de futura se propone llevar a cabo el análisis de las listas de posteo de los tokens remanentes en el índice invertido, con el fin de determinar qué secciones de las mismas podrían podarse, pues como se observa si bien un gran número de entradas son eliminadas del índice se trata de listas cuya contribución al tamaño total de esta estructura resulta escasa. Plantear una estrategia que considere las longitudes de las postings permitirá disminuir el tamaño final del índice invertido aún más, hecho que puede conducir a un incremento en la velocidad del proceso de resolución de consultas hasta el momento alcanzado. Asimismo, será necesario evaluar qué implicaciones presenta tal enfoque sobre la efectividad buscando un compromiso aceptable.

Complementariamente, sería razonable evaluar el desempeño de los IEI sobre un set de consultas que presente una relación más estrecha con el dataset. Esto se debe a que por un lado, el log y la colección empleados corresponden a contextos diferentes y por otra parte, ambos conjuntos exhiben una diferencia temporal considerable. Como fue expuesto, no existen logs de consultas propios de ambientes de microblogging disponibles a la comunidad científica y en consecuencia, es necesario construir uno sintético. El hecho de utilizar un log sintético puede proporcionar robustez a los resultados obtenidos, permitiendo determinar que los mismos no se han sido sesgados por las cuestiones previamente citadas.

Bibliografía

I believe in evidence. I believe in observation, measurement, and reasoning, confirmed by independent observers. I'll believe anything no matter how wild and ridiculous, if there is evidence for it. The wilder and more ridiculous something is, however, the firmer and more solid the evidence will have to be.

Isaac Asimov

[1] S. Alici, I. Altingovde, R. Ozcan, B. Cambazoglu, and Ã. Ulusoy. Adaptive time-to-live strategies for query result caching in web search engines. In R. Baeza-Yates, A. de Vries, H. Zaragoza, B. Cambazoglu, V. Murdock, R. Lempel, and F. Silvestri, editors, *Advances in Information Retrieval*, volume 7224 of *Lecture Notes in Computer Science*, pages 401–412. Springer Berlin Heidelberg, 2012.

[2] I. S. Altingovde, R. Ozcan, and O. Ulusoy. Static index pruning in web search engines: Combining term and document popularities with query views. *ACM Trans. Inf. Syst.*, 30(1):2:1–2:28, Mar. 2012.

[3] V. Anh and A. Moffat. Structured index organizations for high-throughput text querying. In F. Crestani, P. Ferragina, and M. Sanderson, editors, *String Processing and Information Retrieval*, volume 4209 of *Lecture Notes in Computer Science*, pages 304–315. Springer Berlin Heidelberg, 2006.

[4] V. N. Anh. Pruned query evaluation using pre-computed impacts. In *In SIGIR*, page 372379, 2006.

[5] V. N. Anh, O. de Kretser, and A. Moffat. Vector-space ranking with effective early termination. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 35–42, New York, NY, USA, 2001. ACM.

[6] V. N. Anh and A. Moffat. Impact transformation: Effective and efficient web retrieval. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 3–10, New York, NY, USA, 2002. ACM.

[7] N. Asadi. *Multi-Stage Search Architectures for Streaming Documents*. PhD thesis, University of Maryland, 2013.

[8] N. Asadi and J. Lin. Effectiveness/efficiency tradeoffs for candidate generation in multi-stage retrieval architectures. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 997–1000, New York, NY, USA, 2013. ACM.

[9] N. Asadi and J. Lin. Fast candidate generation for real-time tweet search with bloom filter chains. *ACM Trans. Inf. Syst.*, 31(3):13:1–13:36, 2013.

[10] N. Asadi and J. Lin. Fast, incremental inverted indexing in main memory for web-scale collections. *CoRR*, abs/1305.0699, 2013.

[11] N. Asadi, J. Lin, and M. Busch. Dynamic memory allocation policies for postings in real-time twitter search. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 1186–1194, New York, NY, USA, 2013. ACM.

[12] C. Badue, R. Baeza-Yates, B. Ribeiro-Neto, and N. Ziviani. Distributed query processing using partitioned inverted files. In *In Proc. of the 9th String Processing and Information Retrieval Symposium (SPIRE*, pages 10–20. IEEE CS Press, 2001.

[13] R. Baeza-Yates, A. Gionis, F. Junqueira, V. Murdock, V. Plachouras, and F. Silvestri. The impact of caching on search engines. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 183–190, New York, NY, USA, 2007. ACM.

[14] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval - the concepts and technology behind search, Second edition*. Pearson Education Ltd., Harlow, England, 2011.

[15] M. K. Bergman. The Deep Web: Surfacing hidden value. White paper, BrightPlanet, Deep Content, 2001.

[16] E. Bortnikov, R. Lempel, and K. Vornovitsky. Caching for realtime search. In *Proceedings of the 33rd European Conference on Advances in Information Retrieval*, ECIR'11, pages 104–116, Berlin, Heidelberg, 2011. Springer-Verlag.

[17] B. E. Brewington and G. Cybenko. How dynamic is the web? In *Proceedings of the 9th International World Wide Web Conference on Computer Networks : The International Journal of Computer and Telecommunications Netowrking*, pages 257–276, Amsterdam, The Netherlands, The Netherlands, 2000. North-Holland Publishing Co.

[18] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, Sept. 2002.

[19] A. Z. Broder, D. Carmel, M. Herscovici, A. Soffer, and J. Zien. Efficient query evaluation using a two-level retrieval process. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, pages 426–434, New York, NY, USA, 2003. ACM.

[20] M. Busch, K. Gade, B. Larson, P. Lok, S. Luckenbill, and J. Lin. Earlybird: Real-time search at twitter. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ICDE '12, pages 1360–1369, Washington, DC, USA, 2012. IEEE Computer Society.

[21] S. Büttcher and C. L. A. Clarke. A document-centric approach to static index pruning in text retrieval systems. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, CIKM '06, pages 182–189, New York, NY, USA, 2006. ACM.

[22] B. B. Cambazoglu, F. P. Junqueira, V. Plachouras, S. Banachowski, B. Cui, S. Lim, and B. Bridge. A refreshing perspective of search engine caching. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 181–190, New York, NY, USA, 2010. ACM.

[23] D. Carmel, D. Cohen, R. Fagin, E. Farchi, M. Herscovici, Y. S. Maarek, and A. Soffer. Static index pruning for information retrieval systems. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 43–50, New York, NY, USA, 2001. ACM.

[24] C. Chen, F. Li, B. C. Ooi, and S. Wu. Ti: An efficient indexing mechanism for real-time search on tweets. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 649–660, New York, NY, USA, 2011. ACM.

[25] J. Cho and H. Garcia-Molina. The evolution of the web and implications for an incremental crawler. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pages 200–209, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[26] J. Choi and W. B. Croft. Temporal models for microblogs. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 2491–2494, New York, NY, USA, 2012. ACM.

[27] B. Croft, D. Metzler, and T. Strohman. *Search Engines: Information Retrieval in Practice*. Addison-Wesley Publishing Company, USA, 1st edition, 2009.

[28] S. Ding and T. Suel. Faster top-k document retrieval using block-max indexes. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 993–1002, New York, NY, USA, 2011. ACM.

[29] D. Donato, L. Laura, S. Leonardi, and S. Millozzi. Large scale properties of the webgraph. *The European Physical Journal B - Condensed Matter and Complex Systems*, 38(2):239–243, 2004.

[30] F. Douglis, A. Feldmann, B. Krishnamurthy, and J. Mogul. Rate of change and other metrics: A live study of the world wide web. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems on USENIX Symposium on Internet Technologies and Systems*, USITS'97, pages 14–14, Berkeley, CA, USA, 1997. USENIX Association.

[31] M. Efron. Information search and retrieval in microblogs. *J. Am. Soc. Inf. Sci. Technol.*, 62(6):996–1008, June 2011.

[32] D. Fetterly, M. Manasse, M. Najork, and J. Wiener. A large-scale study of the evolution of web pages. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, pages 669–678, New York, NY, USA, 2003. ACM.

[33] N. W. Francis and H. Kučera. *Frequency Analysis of English Usage: Lexicon and Grammar.*, volume 18. Houghton Mifflin, Boston, Apr. 1982.

[34] G. Golovchinsky and M. Efron. Making sense of twitter search, 2010.

[35] R. B. González. *Index Compression for Information Retrieval Systems.* PhD thesis, University of A Coruña, 2008.

[36] C. E. Grant, C. P. George, C. Jenneisch, and J. N. Wilson. Online topic modeling for real-time twitter search. In *Proceedings of The Twentieth Text REtrieval Conference, TREC 2011, Gaithersburg, Maryland, USA, November 15-18, 2011*, 2011.

[37] G. Grefenstette and P. Tapanainen. What is a word, what is a sentence? problems of tokenization. pages 79–87, 1994.

[38] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*, WWW '05, pages 902–903, New York, NY, USA, 2005. ACM.

[39] S. Heinz and J. Zobel. Efficient single-pass index construction for text databases. *J. Am. Soc. Inf. Sci. Technol.*, 54(8):713–729, June 2003.

[40] B. A. Huberman, D. M. Romero, and F. Wu. Social networks that matter: Twitter under the microscope. *CoRR*, abs/0812.1045, 2008.

[41] B. A. Huberman, D. M. Romero, and F. Wu. Social networks that matter: Twitter under the microscope. First Monday, 14(1-5). Recuperado desde `http://firstmonday.org/article/view/2317/2063`, 2009.

[42] B. J. Jansen, G. Campbell, and M. Gregg. Real time search user behavior. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, pages 3961–3966, New York, NY, USA, 2010. ACM.

[43] A. Java, X. Song, T. Finin, and B. Tseng. Why We Twitter: Understanding Microblogging Usage and Communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis*, WebKDD/SNA-KDD '07, pages 56–65, New York, NY, USA, 2007. ACM.

[44] A. N. Joinson. Looking at, looking up or keeping up with people?: Motives and use of facebook. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1027–1036, New York, NY, USA, 2008. ACM.

[45] H. A. Kautz, B. Selman, and M. A. Shah. The hidden web. *AI Magazine*, 18(2):27–36, 1997.

[46] B. Krishnamurthy, P. Gill, and M. Arlitt. A few chirps about twitter. In *Proceedings of the First Workshop on Online Social Networks*, WOSN '08, pages 19–24, New York, NY, USA, 2008. ACM.

[47] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 591–600, New York, NY, USA, 2010. ACM.

[48] G. Laboreiro, L. Sarmento, J. Teixeira, and E. Oliveira. Tokenizing micro-blogging messages using a text classification approach. In *Proceedings of the Fourth Workshop on Analytics for Noisy Unstructured Text Data*, AND '10, pages 81–88, New York, NY, USA, 2010. ACM.

[49] H. T. Lam, R. Perego, and F. Silvestri. On using query logs for static index pruning. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, volume 1, pages 167–170, Aug 2010.

[50] C. Lee, H. Kwak, H. Park, and S. Moon. Finding influentials based on the temporal order of information adoption in twitter. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 1137–1138, New York, NY, USA, 2010. ACM.

[51] N. Lester, J. Zobel, and H. Williams. Efficient online index maintenance for contiguous inverted lists. *Inf. Process. Manage.*, 42(4):916–933, July 2006.

[52] J. Lin and G. Mishne. A study of çhurnïn tweets and real-time search queries. In *Proceedings of the Sixth International Conference on Weblogs and Social Media, Dublin, Ireland, June 4-7, 2012*, 2012.

[53] H. P. Luhn. The automatic creation of literature abstracts. *IBM J. Res. Dev.*, 2(2):159–165, Apr. 1958.

[54] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[55] E. Markatos. On caching search engine query results. *Comput. Commun.*, 24(2):137–143, Feb. 2001.

[56] R. McCreadie, I. Soboroff, J. Lin, C. Macdonald, I. Ounis, and D. McCullough. On building a reusable twitter corpus. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 1113–1114, New York, NY, USA, 2012. ACM.

[57] D. McCullough, J. Lin, C. Macdonald, I. Ounis, and R. McCreadie. Evaluating real-time search over tweets. In *Proceedings of the Sixth International Conference on Weblogs and Social Media, Dublin, Ireland, June 4-7, 2012*, 2012.

[58] D. Metzler, C. Cai, and E. Hovy. Structured event retrieval over microblog archives. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 646–655,

Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[59] A. Moffat and T. A. H. Bell. In situ generation of compressed inverted files. *J. Am. Soc. Inf. Sci.*, 46(7):537–550, Aug. 1995.

[60] B. H. Murray and A. Moore. Sizing the internet. White paper, Cyveillance, Inc., 2000.

[61] S. Nepomnyachiy, B. Gelley, W. Jiang, and T. Minkus. What, where, and when: Keyword search with spatio-temporal ranges. In *Proceedings of the 8th Workshop on Geographic Information Retrieval*, GIR '14, pages 2:1–2:8, New York, NY, USA, 2014. ACM.

[62] A. Ntoulas and J. Cho. Pruning policies for two-tiered inverted index with correctness guarantee. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 191–198, New York, NY, USA, 2007. ACM.

[63] A. Ntoulas, J. Cho, and C. Olston. What's new on the web?: The evolution of the web from a search engine perspective. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 1–12, New York, NY, USA, 2004. ACM.

[64] I. Ounis, C. Macdonald, J. Lin, and I. Soboroff. Overview of the trec-2011 microblog track. In *In Proceedings of TREC 2011*, 2011.

[65] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *Proceedings of the 1st International Conference on Scalable Information Systems*, InfoScale '06, New York, NY, USA, 2006. ACM.

[66] M. Persin, J. Zobel, and R. Sacks-Davis. Filtered document retrieval with frequency-sorted indexes. *J. Am. Soc. Inf. Sci.*, 47(10):749–764, Sept. 1996.

[67] C. J. V. Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979.

[68] G. Salton. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

[69] G. Salton and M. E. Lesk. Computer evaluation of indexing and text processing. *J. ACM*, 15(1):8–36, Jan. 1968.

[70] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[71] D. Shan, S. Ding, J. He, H. Yan, and X. Li. Optimized top-k processing with global page scores on block-max indexes. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 423–432, New York, NY, USA, 2012. ACM.

[72] G. Skobeltsyn, F. Junqueira, V. Plachouras, and R. Baeza-Yates. Resin: A combination of results caching and index pruning for high-performance web search engines. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 131–138, New York, NY, USA, 2008. ACM.

[73] I. Soboroff, I. Ounis, C. Macdonald, and J. Lin. Overview of the trec-2012 microblog track. In *In Proceedings of TREC 2012*, 2012.

[74] T. Strohman and W. B. Croft. Efficient document retrieval in main memory. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 175–182, New York, NY, USA, 2007. ACM.

[75] J. Teevan, D. Ramage, and M. R. Morris. #twittersearch: A comparison of microblog search and web search. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 35–44, New York, NY, USA, 2011. ACM.

[76] G. H. Tolosa and F. R. A. Bordignon. Introducción a la Recuperación de Información. Conceptos, Modelos y Algoritmos Básicos. Material de Estudio, Universidad Nacional de Luján, División Estadísticas y Sistemas, Departamento de Ciencias Básicas, 2007.

[77] A. Tomasic, H. García-Molina, and K. Shoens. Incremental updates of inverted lists for text document retrieval. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, SIGMOD '94, pages 289–300, New York, NY, USA, 1994. ACM.

[78] N. Tonellotto, C. Macdonald, and I. Ounis. Query efficiency prediction for dynamic pruning. In *Proceedings of the 9th workshop on Large-scale and distributed informational retrieval - LSDS-IR '11*, page 3, 2011.

[79] N. Tonellotto, C. Macdonald, and I. Ounis. Efficient and effective retrieval using selective pruning. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 63–72, New York, NY, USA, 2013. ACM.

[80] D. Tunkelang. A Twitter analog to PageRank. Recuperado desde `http://thenoisychannel.com/2009/01/13/a-twitter-analog-to-pagerank`, 2009.

[81] H. Turtle and J. Flood. Query evaluation: Strategies and optimizations. *Inf. Process. Manage.*, 31(6):831–850, nov 1995.

[82] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2Nd ACM Workshop on Online Social Networks*, WOSN '09, pages 37–42, New York, NY, USA, 2009. ACM.

[83] J. Weng, E.-P. Lim, J. Jiang, and Q. He. Twitterrank: Finding topic-sensitive influential twitterers. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 261–270, New York, NY, USA, 2010. ACM.

[84] Y. Xie and D. O'Hallaron. Locality in search engine queries and its implications for caching. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1238–1247 vol.3, 2002.

[85] J. Zhang, X. Long, and T. Suel. Performance of compressed inverted list caching in search engines. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 387–396, New York, NY, USA, 2008. ACM.

[86] D. Zhao and M. B. Rosson. How and why people twitter: The role that micro-blogging plays in informal communication at work. In *Proceedings of the ACM 2009 International Conference on Supporting Group Work*, GROUP '09, pages 243–252, New York, NY, USA, 2009. ACM.

[87] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Comput. Surv.*, 38(2), July 2006.