FACULTAD DE INGENIERÍA

INGENIERÍA CIVIL INFORMÁTICA

Tópicos en ráfaga en aplicaciones web de gran escala

Tesis para optar al grado de ingeniero civil en informática

Autor:

Omar Opazo Rojas

Profesor guía: **Carlos Gómez-Pantoja**

Santiago de Chile, Chile.

Septiembre, 2015

Capítulo 1

Resumen

La web como se conoce hoy en día, se compone de un sin número de sitios o *websites*. Al año 2012, la estimación de *websites* fue de 634 millones, 51 de los cuales fueron creados el mismo año. El número de usuarios, en el mismo periodo, se estimó en 2,4 billones [21].

Existen puntos de convergencia de usuarios, como un subconjunto de todos los sitios de la web; *websites* con alta concurrencia de usuarios, manejo de grandes volúmenes de datos y requerimientos de eficiencia en sus desempeños. Estos *websites* se conocen como aplicaciones web de gran escala.

Las aplicaciones web de gran escala, internamente son soportadas por un conjunto de máquinas o *nodos* que responden a las solicitudes de los usuarios. El desempeño de las aplicaciones web de gran escala, depende entre otros factores, del correcto balance de carga entre estos *nodos*.

Dentro de los eventos que merman el desempeño de las aplicaciones web de gran escala, están las denominadas "consultas en ráfaga". En [18] se define el comportamiento ráfaga como *una alta frecuencia inusual para una consulta*. Lo importante de las consultas en ráfaga es que son consultas que **deben** ser detectadas antes de alcanzar su cota superior de frecuencias, debido a que no hacerlo podría inhabilitar los *nodos* que responden a dichas consultas por la sobrecarga que implica la resolución de tantas consultas en un intervalo de tiempo tan reducido.

Las consultas en ráfaga, en el contexto de análisis de *outliers*, pueden verse como un tipo de *anomalía colectiva* [5] , ya que el aumento de la frecuencia se da de forma gradual (aunque acelerada), lo que hace factible realizar predicciones de ráfagas con modelos no necesariamente complejos.

Las consultas en ráfaga suelen referirse a un tópico en particular que genera un inusitado interés por parte da la comunidad. Sin embargo, no todas las consultas que hablan sobre un tópico en ráfaga, llegan a convertirse en una consulta en ráfaga.

Este trabajo se focaliza en generar un método eficiente en tiempo y espacio, capaz de detectar tópicos en ráfagas y mediante esos tópicos detectados como ráfagas, encontrar las consultas en ráfaga asociadas a ellos.

Abstract

Agradecimientos

Índice general

Índice de figuras

Índice de cuadros

Índice de algoritmos

Capítulo 2

Introducción

2.1. Motivación

En la actualidad, acceder a los servicios que otorga la web (o *world wide web*) parece ser una tarea básica pero fundamental para las personas. La sociedad asume el poder acceder a los recursos provistos por la web, ya sea por trabajo, de manera informativa o simplemente por ocio, como una actividad tácitamente disponible 24/7.

Dentro de los mismos servicios que otorga la web, existen puntos de convergencia de usuarios, a los cuales muchas personas acceden, como lo son las redes sociales (ej: *Facebook, Twitter*), motores de búsqueda (ej: *Google, Yahoo!*) o comunidades de pregunta/respuesta (ej: *Yahoo! answers, Microsoft comunity*). Estos servicios deben lidiar con un alto número de usuarios que concurren a sus *websites*, lo que requiere que sus arquitecturas estén diseñadas de forma que sean capaces de resolver las peticiones de los usuarios de manera rápida. Es decir, cada componente de estos *websites*, debe estar orientado a aplicaciones web de gran escala.

Las aplicaciones web de gran escala, internamente se distribuyen en múltiples máquinas o *nodos* que deben lidiar con las peticiones de los usuarios, resolverlas y entregar los resultados de forma rápida. En ese sentido, el desempeño de una aplicación de gran escala para una determinada petición, está sujeto al estado de la máquina a la cual se envió la petición a ser resuelta. Entonces, mientras menos recursos del nodo en cuestión estén siendo utilizados al momento de resolver la petición, mejores serán los resultados en cuanto a rapidez de respuesta. Asimismo, si la petición de un usuario es direccionada a un nodo sobrecargado, el proceso de resolución y envio de resultados se torna menos eficiente.

Normalmente las consultas de los usuarios, tienden a tener un comportamiento sesgado, donde la gran masa de usuarios tiende a utilizar una gamma reducida de palabras en sus consultas, las aplicaciones web de gran escala deben lidiar con este comportamiento. Además, existe un fenómeno que es transversal al contexto, conocido como *Consultas en ráfaga*, el cual requiere de un tratamiento especial.

Una consulta en ráfaga es un tipo especial de consulta, la cual en un periodo de tiempo acotado, sufre un alza abrupta en su frecuencia. Las consultas en ráfaga, en general, obedecen a eventos aislados, pero de interés popular, por ejemplo, la muerte de alguna celebridad (o un hecho en especial asociado a ellas), el lanzamiento de un nuevo producto al mercado, etc. El problema principal de estas consultas tiene que ver con el desbalance que producen en los *nodos* que responden a las peticiones en una aplicación web, ya que por elevar desmesuradamente la frecuencia de una misma consulta, la cantidad de recursos que requiere el *nodo* que responde a la consulta (en ráfaga) aumenta (generalmente de forma drástica), lo que, en el peor caso, puede inhabilitar los servicios soportados por esa máquina.

Lo ideal es detectar una consulta en ráfaga, antes de que ésta provoque la inhabilitación de un *nodo*, para así levantar una advertencia sobre esa consulta y realizar medidas de balance de carga, o priorización de la consulta en un sistema de respuestas precomputadas.

Un problema es que los puntos de convergencia de usuarios pueden ser vistos como contextos distintos, lo que produce que tengan características distintas tanto en la manera de como reciben peticiones desde los usuarios, como en la forma en la cual entregan las respuestas a las peticiones. Sin embargo, como se menciona anteriormente, las consultas en ráfaga obedecen a eventos o hechos en especial, es decir, hablan de un **tópico** en particular.

Este trabajo, tiene por finalidad, definir un método de detección temprana y validación de tópicos en ráfaga, orientado a aplicaciones web de gran escala. La detección se orienta a los tópicos puesto que esto quiebra la limitante de los distintos contextos existentes en las aplicaciones web de gran escala: un tópico por si solo es independiente del contexto. Además, se debe considerar que al enforcase a un contexto de aplicaciones web de gran escala, los mecanismos generados deben ser eficientes en tiempo y espacio.

Finalmente, este trabajo de tesis busca responder las siguientes preguntas:

1. ¿Es posible definir un método de capaz de hacer **detección** de tópicos en ráfaga sobre una masa de tópicos con distintos comportamientos en sus series de fecuencias, eficiente en tiempo y espacio?

2. ¿Es posible definir un método de capaz de **validar** a los tópicos detectados como ráfaga, basado en los intereses de los usuarios en torno a esos tópicos en un instante de tiempo determinado, eficiente en tiempo y espacio?

3. ¿Es posible definir un método que identifique las consultas ráfagas asociadas a los tópicos declarados como ráfaga (y descarte las consultas no ráfagas), eficiente en tiempo y espacio?

2.2. Antecedentes

2.3. Contribución de la tesis

2.4. Estructura de la tesis

Este trabajo se divide en 6 capítulos.

El capítulo 1 presenta la motivación del trabajo, donde se explica, la necesidad de contar con un método de detección de ráfagas eficiente en tiempo y espacio. Se explica además, la orientación del trabajo y se justifica esta orientación.

El capítulo 2 se centra en definir el problema que motiva la realización de la tesis. En este capítulo se define el contexto de trabajo donde se emplean los métodos propuestos para la detección de ráfagas. Se realiza la contextualización del problema con la finalidad de enmarcar el trabajo. Los objetivos (general y específicos) se muestran en este capítulo, junto con la hipótesis que se baraja en la investigación. Finalmente, se define la metodología de trabajo, la cual es una modificación al método científico clásico y se expone un apartado sobre los entornos de trabajo y herramientas utilizadas en la investigación, para cerrar con el alcance del trabajo, donde se termina de estructurar la investigación.

El capítulo 3 comprende únicamente la especificación del marco teórico, definiendo conceptos clave utilizados en el trabajo. De aquí, lo más importante es la sección 6.1 donde se definen los tipos de consultas como “señales”, puesto que es fundamental esta definición para comprender de

buena forma los métodos empleados en la etapa experimental.

El capítulo 4, comprende la revisión del estado del arte, donde se hace una revisión de la literatura existente y se destacan las ideas más importantes planteadas por otros autores. Este trabajo de revisión, sienta las primeras bases para el desarrollo de este trabajo. Esto resulta importante para cualquier trabajo de tesis.

El capítulo 5 presenta el desarrollo de la solución. Los subcapítulos 5.1, 5.2 y 5.3 son de gran importancia, debido a que muestran el desarrollo de los métodos creados para cada contexto de trabajo. En el subcapítulo 5.1 se muestra el desarrollo del contexto "Comunidad de pregunta/respuesta", donde se emplean métodos de detección de ráfagas orientado a tópicos y validación de ráfagas orientado a actividad de clicks. El subcapítulo 5.2 muestra el desarrollo del contexto "Log de consultas" donde se prueba el método de detección de ráfagas con orientación a tópicos. Finalmente, el subcapítulo 5.3 trata sobre la detección de ráfagas en el contexto de *Twitter* y la validación de estas mediante la actividad de *retuits*.

El capítulo 6 muestra las conclusiones final del trabajo, luego de revisar los resultados de la experimentación y se describe el trabajo futuro.

Capítulo 3

Descripción del problema

Como se menciona en la sección 1, las aplicaciones web de gran escala internamente son soportadas por un conjunto de máquinas o nodos, los cuales se encargan de resolver las peticiones de los usuarios y entregar las respuestas a dichas peticiones. El estado de los nodos se relaciona directamente con el desempeño de la aplicación soportada, por lo tanto, el comportamiento deseable para una aplicación de gran escala, es mantener un balance de carga eficiente entre los nodos subyacentes.

Las consultas en ráfaga, son un fenómeno que impacta directamente en el desbalance de los nodos, puesto que al aumentar la frecuencia de una consulta de forma abrupta en un instante de tiempo acotado, las peticiones al nodo que responde a la consulta también aumentan, generando una sobrecarga a dicho nodo, o en el peor caso, la inhabilitación del nodo, lo que haría reducir el desempeño de la aplicación o inhabilitarla.

Desde el punto de vista de *caché*, las consultas en ráfaga también se tornan relevantes. Por ahora se puede decir que el *caché* en el contexto de motores de búsqueda, se utiliza para mantener respuestas precomputadas a consultas altamente demandadas, generando respuestas rápidas y sin la necesidad de obligar a los nodos a buscar las respuestas a estas consultas cada vez que se requieran, por lo cual se evita la sobrecarga a los nodos. El caché será revisado en el capítulo 4. Las consultas en ráfaga en este contexto, son de vital importancia, puesto que normalmente **no** se consideran en el *caché* ya que estas consultas de existir, son muy poco consultadas por los usuarios antes de tornarse ráfagas. Al aumentar la frecuencia de una consulta que se transformará en ráfaga, esta debiese estar, mientras dure su "intervalo de importancia" (alta frecuencia), en *caché*.

La figura 3.1, muestra el diagrama causa/efecto para simplificar la comprensión del problema.

No manejar las consultas en ráfaga podría provocar las causas expuestas, es decir, sería una de las causas de las causas expuestas.

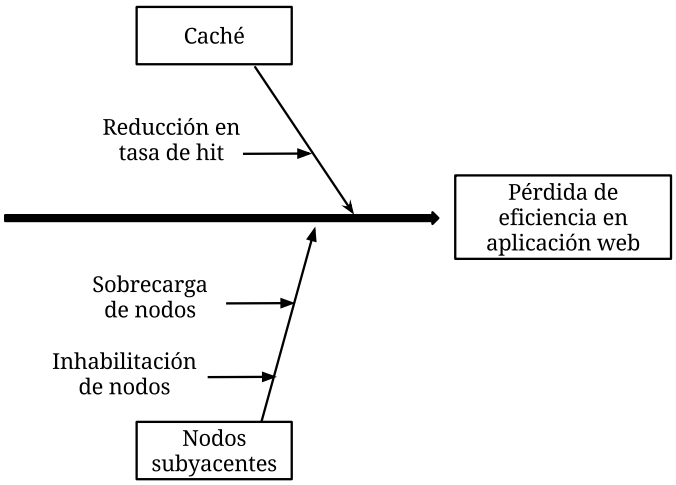

Figura 3.1: Diagrama causa/efecto

- **Caché**: No considerar las consultas en ráfaga produciría una reducción en la tasa de hit, puesto que en algún momento, las consultas en ráfaga son altamente demandadas por los usuarios. No almacenarlas en *Caché*, produciría fallos o *miss* realizar la búsqueda de la consultas.
- **Nodos subyacentes**: No considerar las consultas en ráfaga podría producir una sobrecarga de los nodos subyacentes a las aplicaciones o en el peor caso, la inhabilitación de los nodos.

Este trabajo no se enfoca en las causas expuestas en el diagrama causa efecto, puesto que como antes se describe, el tratamiento de las consultas en ráfaga obedece a una de las causas previas a las causas del diagrama.

3.1. Contextualización

Este trabajo se enfoca en diseñar e implementar un mecanismo de detección de tópicos en ráfaga y un mecanismo de validación de tópicos en ráfaga. La contexto de estos mecanismos es *online*, lo que implica que ambos mecanismos deben ser eficientes en tiempo y espacio.

Los algoritmos presentados en este documento, emulan un comportamiento *online*, la preparación de los datos a analizar no representa el núcleo de los mecanismos, por lo cual, la preparación de los datos no es relevante al momento de analizar la calidad de los algoritmos.

Los resultados experimentales de los algoritmos, se comparan a métodos encontrados en la revisión del estado del arte, con la finalidad de conocer el desempeño de los mecanismos propuestos versus lo existente.

Dependiendo del contexto sobre el cual se trabaja, los algoritmos están sujetos a variaciones en el sentido de la obtención de los datos que se desean analizar, sin embargo, el mecanismo para declarar o no una ráfaga, se basa en el uso del coeficiente de variación para todos los contextos. Esto es invariable y representa el enfoque de este trabajo.

El coeficiente de variación se utiliza para observar la relación entre la media de la muestra (una serie de frecuencias) y la variabilidad de la variable observada. Esta medida es utilizada debido a que utiliza la desviación estándar, lo que otorga el grado de dispersión de los datos y además utiliza la media, lo que puede ser más preciso al separar series ráfagas versus permanentes.

La orientación de la detección, es hacia los tópicos, por sobre las consultas, teniendo esto por finalidad, crear un mecanismo independiente del contexto sobre el cual se trabaja y romper las limitantes propias de algunos contextos (p.e. uso de lenguaje natural en CQA).

Los contextos de trabajo son:

- Comunidad de pregunta respuesta (CQA), en específico *Yahoo! Answers*.
- Motor de búsqueda (mediante *logs* de consultas de Yahoo!).
- Red social (*Twitter*).

Además, se busca definir un mecanismo capaz de validar los tópicos en ráfaga, considerando dos aspectos:

- Es realmente el tópico, un tópico en ráfaga.
- Si lo es, ¿cuales de las consultas asociadas al tópico, corresponden a consultas en ráfaga?.

3.2. Enunciado del problema

En este trabajo, se aborda el problema de definir un método independiente del contexto para la detección de tópicos en ráfaga y un método de validación de tópicos en ráfaga, ambos eficientes tanto en tiempo como en espacio.

3.3. Objetivos

3.3.1. Objetivo general

El objetivo de este trabajo es generar un mecanismo de detección temprana de tópicos en ráfaga, eficiente en tiempo y espacio, válido para los contextos de comunidades de pregunta/respuesta, motores de búsqueda y redes sociales. Además se busca generar mecanismos de validación de tópicos declarados como ráfagas, mediante distintas características, de acuerdo al contexto estudiado.

3.3.2. Objetivos específicos

Los objetivos específicos se detallan a continuación:

- Generar un método eficiente en tiempo y espacio para detectar ráfagas, considerando tópicos por sobre consultas completas.
- Generar y validar estrategias computacionalmente aceptables en contextos de aplicaciones web de gran escala, para realizar validaciones automáticas de detecciones de ráfagas.

3.4. Hipótesis

Las hipótesis que se barajan en este trabajo, son la siguiente:

- Un método de detección de tópicos en ráfaga, podría hacer más eficiente (más temprana) la detección que los métodos orientados a las consultas.
- La actividad de clicks asociada a las consultas y el contador retuits asociado a un tuit, serviría como medio de validación de ráfagas.

3.5. Metodología de trabajo

A pesar de que este trabajo se desarrolla sobre el marco del método científico, se plantea una metodología de trabajo más robusta que el método científico clásico. La finalidad de esto, es man-

tener una estructura básica que otorgue orden y trazabilidad sobre los distintos algoritmos o ideas que se desarrollarán.

La figura 3.2 muestra la metodología empleada en este trabajo. La línea superior, muestra la etapa clásica del método científico. La diferencia se hace sobre la etapa de especificación de metodología de trabajo, haciendo hincapié sobre esta tarea.

Primeramente, se realiza una planificación, donde por un lado se estiman los plazos de desarrollo y por otro se definen las herramientas de control, manipulación de datos y también se define un entorno de trabajo. Posteriormente, se pasa a la etapa de desarrollo, una etapa iterativa que comprende por sí sola análisis, diseño, codificación y pruebas. Esta etapa se realiza para los 3 contextos abordados en este problema.

Para este trabajo resulta fundamental establecer una etapa de análisis del contexto sobre el cual se trabajará, debido a que cada uno de los contextos a abordar, tienen características diferentes. Posteriormente, con el análisis realizado, se diseña el método a proponer y el (o los) método comparativo, todo esto a nivel de ideas y pseudocódigo. Adaptar los datos a los distintos métodos, se toma como una etapa aparte, ya que esto muchas veces dista de ser algo trivial. Luego de esto, se codifican los métodos pensados en etapas anteriores y se pasa a la etapa de cierre donde se preparan las salidas de los algoritmos a ser procesadas. Finalmente estos resultados son pasados a la etapa de procesamiento y análisis de resultados que compone la penúltima etapa del método científico clásico. Por último, los resultados de cada contexto se condensan en la etapa final de cierre y concusiones.

3.5.1. Herramientas y entorno de trabajo

Para el desarrollo de este trabajo, se utilizaron las siguientes herramientas:

- Lenguaje de programación C++, g++ 4.8.
- Lenguaje de programación Java 1.7.0_79.
- Procesador de texto Awk 4.0.1.
- Herramienta de gráficos Gnuplot 4.6 patchlevel 4.
- Framework para aplicaciones distribuídas Hadoop 2.6.0.

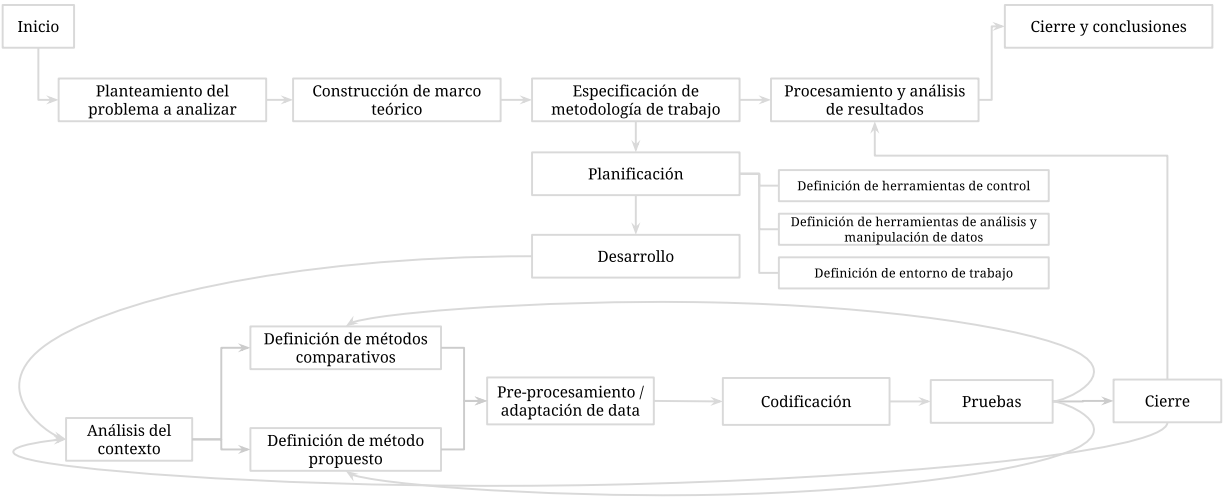

Figura 3.2: metodología de trabajo

- PdfTeX 3.1415926-2.5-1.40.14

El entorno de trabajo se especifica a continuación:

- Servidor Debian 3.2.63-2 x86_64 Intel(R) Xeon(R) CPU E7- 4830 @ 2.13GHz, 32 núcleos , 64 GB ram. Compartido.
- Servidor Ubuntu 14.04.2 LTS (GNU/Linux) x86_64, Intel(R) Xeon(R) CPU E5-2637 v2 a 3.50GHz, 16 núcleos, 256 GB ram. Compartido.

3.6. Alcance

Este trabajo, comprende la especificación e implementación de métodos de detección de tópicos en ráfaga en tres contextos de aplicaciones web de gran escala. Además contempla métodos de validación de tópicos declarados como ráfaga, utilizando como referencia, estudios previos revisados en la inspección del estado del arte.

Todos los mecanismos propuestos en este trabajo, pretenden sentar una base en cuanto a métodos de deteción de tópicos en ráfaga y validación de estos **en un contexto *online***.

Como se menciona anteriormente, este trabajo pretende abarcar tanto la especificación de mecanismos, como la implementación de cada uno de ellos. Esto es beneficioso del punto de vista del análisis de los resultados, ya que utilizando distintas técnicas, se podrá revisar el desempeño de cada uno de los métodos propuestos.

Capítulo 4

Marco Teórico

4.1. Aplicaciones web de gran escala

Las aplicaciones web de gran escala son servicios web con alta concurrencia de usuarios y manejo de grandes volúmenes de datos. Este tipo de aplicaciones requiere que tanto su *back-end* como su *fron-end* sean soportados por algoritmos eficientes en tiempo y espacio. Además requiere de una infraestructura capaz de soportar la carga de usuarios y capaz de entregar alta disponibilidad, por lo que en este tipo de aplicaciones, predominan las arquitecturas distribuídas.

4.1.1. Motores de búsqueda

Según [11] los motores de búsqueda web "son sistemas computacionales que 'almacenan' la web y en los que los usuarios realizan búsquedas sobre esta información almacenada. La respuesta del motor de búsqueda web a una búsqueda es una serie de documentos que la responden de mejor manera siguiendo una función de *ranking*". La infraestructura que soporta los motores de búsqueda web, se compone de múltiples nodos que trabajan de forma distribuída por conjuntos. Cada conjunto de nodos es utilizado para cubrir ciertas funcionalidades.

Independiente de las funcionalidades que cada conjunto de nodos deba cubrir, lo importante es mantener el balance entre la carga de dichos nodos, puesto que no hacerlo, puede desencandenar la inhabilitación de alguno de ellos y con esto, el sistema completo puede ver reducido su desempeño.

4.1.2. Caché

El *Caché* es un servicio fundamental para un motor de búsqueda. La idea de utilizar el *caché* es almacenar consultas y sus respuestas precomputadas, para así hacer más rápida la respuesta a estas

consultas. Este tipo de tecnología es limitada en cuanto a espacio (lo que también incide en que sea veloz), por lo que es necesario que los datos almacenados dentro de *caché*, realmente sean de utilidad.

El servicio de *caché*, debiese estar optimizado para almacenar respuestas precomputadas a aquellas consultas que más se realicen, ya sea de manera histórica o de manera temporal. El *caché* es de alto acceso, ya que al llegar una consulta de un usuario al motor de búsqueda, ésta pasa por un servicio llamado, *servicio de fron-end* (FS) (el cual se encarga de gestionar la respuesta a una consulta recibida) e inmediatamente busca en *caché* la consulta del usuario. Si esta consulta se encuentra en *caché*, la respuesta precomputada es devuelta inmediatamente al usuario, mientras que si no está en *caché*, se debe buscar la respuesta en otro servicio (llamado servicio de índice (IS)) lo que torna el proceso mucho menos eficiente.

4.1.3. Índice invertido

El índice invertido es una estructura de datos utilizada para indexación de documentos. Dada una consulta q, se genera un vocabulario para q extrayendo las palabras T distintas de la consulta. Luego, se forma una lista invertida, para cada palabra T_i, con los documentos donde aparece la palabra [29]. Además, el índice invertido puede traer consigo más información que una indicación binaria sobre si una palabra está o no en un documento, como por ejemplo, la frecuencia de apariciones de la palabra [6]. Esta lista de ocurrencias de cada T_i se denomina *Posting list*.

Esta estructura de datos está orientada al alto desempeño. Empresas como Google utilizan índice invertido para organizar los resultados de las búsquedas de los usuarios [3].

El principal problema de esta estructura de datos, es que puede consumir demasiados recursos en memoria. En estos casos, se hace uso de *índices parciales*, guardando en disco el índice obtenido hasta un momento determinado, lo que permite vaciar la memoria y comenzar el proceso nuevamente. Finalmente, los resultados de los índices parciales pasan por un proceso de *merge*, para obtener el índice invertido completo.

En este trabajo, se hace uso de índice invertido en el sentido de encontrar las consultas que contienen a un determinado término declarado como ráfaga.

4.1.4. Outliers

Un *outlier* puede ser definido como un dato o punto que es muy diferente del resto de los datos o puntos de la muestra [1]. Bajo este marco, la detección de *outliers* se ha utilizado con diversas finalidades, como lo son: detección de fraudes financieros, diagnósis médica, etc. Además se ha empleado el análisis de *outliers* para predecir eventos naturales, como terremotos o maremotos.

Las consultas o tópicos en ráfaga pueden verse como un tipo de anomalía colectiva [5] , es decir, una colección de puntos de la serie de una consulta o tópico puede presentar un comportamiento anómalo con respecto a la "normalidad" de la muestra (la media). Esto introduce, implicitamente, los conceptos de *media* y *varianza* como variables a considerar.

4.1.5. Comunidades de pregunta/respuesta

Las comunidades de pregunta/respuesta, son portales de intercambio de información entre usuarios, donde los usuarios tienen la posibilidad de ser activos en la comunidad haciendo consultas sobre temas de interés y esperando a que otros usuarios puedan responder en base a sus propios conocimientos u otras fuentes de información. Del mismo modo, un usuario puede ser activo en la comunidad respondiendo preguntas de otros usuarios. Según [14], las comunidades de pregunta/respuesta "se están convirtiendo rápidamente en una fuente de conocimiento sobre temas que no están bien atendidos por los motores de búsqueda generales", es decir, estas comunidades se hacen valiosas en el sentido de que existen preguntas las cuales los motores de búsqueda no resuelven de la mejor manera.

Uno de los mayores problemas al procesar datos provenientes de comunidades de pregunta/respuesta, es que estos están escritos en lenguaje natural, lo que hace más complejo el análisis.

La orientación de este trabajo (tópicos) puede lidiar con el problema del lenguaje natural, ya que, si bien es cierto, al procesar los datos provenientes de comunidades de pregunta/respuesta se tendrán verbos con distintas conjugaciones, jergas, etc., los tópicos tienden a ser invariables en ese sentido.

4.1.6. Logs

Uno de los enfoques de este trabajo es el procesamiento de *logs* de consultas de usuarios. Estos archivos contienen información acerca de las búsquedas de los usuarios en un motor de búsqueda (el *string* de la consulta y el *timestamp*). El procesamiento de estos archivos no debe lidiar con el

problema del análisis en las comunidades pregunta/respuesta (lenguaje natural), puesto que son búsquedas realizadas directamente sobre el buscador, donde predominan las preguntas concisas (2-3 términos) [25].

4.1.7. Redes sociales: *Twitter*

En [8] se define redes sociales como "los servicios basados en la web que permiten a los individuos (i) construir un perfil público o semi-público dentro de un sistema acotado, (ii) articular una lista de otros usuarios con los que comparten una conexión y (iii) ver y recorrer su lista de conexiones y las hechas por otros dentro del sistema".

Las redes sociales conforman hoy en día, los puntos de convergencia más importantes en cuanto cantidad de usuarios, según [26], la red social con mayor cantidad de usuarios activos (*facebook*) supera el billón de usuarios registrados. Por su parte *Twitter* tiene la no despreciable suma de 288 millones de usuarios activos al año 2015.

Según [31] *Twitter* "es una red social o servicio de microblogging que permite a sus usuarios registrados difundir mensajes cortos llamados *twits (tweets)*". En *Twitter*, los usuarios pueden *seguir* a otros usuarios y así recibir los mensajes o *twits* que estos emiten. *Twitter* se ha proyectado como una red social informativa, llegando incluso a ser medio de información fiable para noticieros.

Twitter ofrece a los usuarios la opción de *retuitear* (*retweet*) un post de otro usuario que sea de su agrado o interés, de esta forma, las noticias relevantes se mueven por los *nodos* (que pueden ser ciudades, países e incluso continentes): el *retuiteo* funciona como arista entre *nodos* parcial o totalmente desacoplados.

4.2. Comportamiento de usuario

4.2.1. Zipf

En base a sus estudios, George Kingsley Zipf formuló una ley denominada "Ley de Zipf" [37][38]. Esta ley empírica establece que la tendencia humana es utilizar un conjunto reducido de palabras, mientras que el conjunto mayor de palabras son utilizadas muy pocas veces. Esta ley se ha validado en numerosos estudios sobre los motores de búsqueda y la web, diseños de sistemas de caché eficientes, etc. [4][19][23][33]. Es decir, a la hora de utilizar la web, los usuarios, en su gran mayoría, tienden a utilizar un conjunto reducido de palabras y rara vez utilizan palabras fuera de ese conjunto.

Lo anterior debiese ser considerado a la hora de procesar los textos, ya sean consultas de usuarios, *tweets* u otros. Existen palabras que no aportan a un procesamiento de los tópicos ráfagas, por ser palabras de uso frecuente en los usuarios. Este tipo de palabra también se denomina como *stopwords* o "palabras vacias".

4.2.2. Categorías de consultas, tipos de señales

La serie de frecuencias que genera una consulta o tópico a través del tiempo, puede ser entendida como una señal, es decir, dependiendo del tipo de consulta o tópico (en el sentido del interés del usuario), la señal que genera las frecuencias de apariciones puede ser de distinto tipo:

Permanentes

Este tipo de señal, corresponde a señales con altas frecuencias a través del tiempo. Consultas como *Facebook , Google o Twitter* son las que generan este tipo de señal.

Desde el punto de vista de un sistema de *caché*, este tipo de consultas debiese estar **siempre** dentro del *caché*, puesto que se sabe de antemano que este tipo de consultas es altamente demandada por los usuarios, además los resultados de estas consultas no debiesen conllevar un proceso de *indexación*.

Periódicas

Este tipo de señal, corresponde a señales con altas frecuencias por instantes de tiempo no muy prolongados, y luego un decaimiento en las frecuencias. Este comportamiento se repite por periodos de tiempo definidos. Consultas como *Fifa world cup* o *presidential election* son las que generan este tipo de señal.

Desde el punto de vista de un sistema de *caché*, este tipo de consultas, no debiese estar siempre almacenadas en *caché*, puesto que al ser el *caché* una memoria reducida, se podría quitar espacio a una consulta permanente o incluso a una ráfaga, estando en un instante de tiempo donde la frecuencia de la consulta periódica fuese baja. Una política de admisión/desalojo de *caché*, debiese considerar el desalojo de las consultas periódicas en sus periodos de baja frecuencia.

Ráfagas

Este tipo de señal, corresponde a señales con frecuencias casi nulas en su comportamiento normal y luego, en periodos de tiempo muy acotados sufren alzas abruptas en sus frecuencias para posteriormente exponer un decaimiento gradual en la frecuencia de su señal. Consultas como *Osama Bin Laden death* o *MH17* (derribo de avión de Malaysia Airlines), son consultas que generan este tipo de señal.

Desde el punto de vista de un sistema de *caché*, este tipo de consultas no debe estar en *caché* hasta detectarse que corresponde a un evento en ráfaga, donde debiese priorizarse su ingreso a la memoria. Sin embargo, es necesario que la detección de este tipo de consultas, se realice mucho antes de alcanzar su máxima frecuencia, ya que probablemente requieran de un tratamiento especial en cuanto a balance de carga.

Capítulo 5

Estado del arte

A continuación, se describen los principales estudios extraídos en la revisión del estado del arte, con respecto a los métodos de detección de ráfagas:

En [27] Subasic y Castillo estudian el comportamiento de los usuarios durante un evento ráfaga, contrastando este comportamiento con uno normal, es decir, antes (pre-episodio) y después (post-episodio) de la ráfaga. En dicho estudio, se propone un método que considera la frecuencia de una *query* en el instante actual, versus la frecuencia de ella en el pasado. El método clasifica la duración de los episodios para asociar los distintos "tipos de ráfaga" a unas métricas predefinidas por ellos mismos. Además, se concluye que **durante los episodios de ráfagas, la actividad de clicks tiende a ser más concentrada que en los pre-episodios y post-episodios**. Esto último presenta una arista a considerar, ya que la actividad de clicks puede ser utilizada como medio de validación de ráfagas, es decir, si se tiene un tópico T declarado como posible ráfaga, la actividad de clicks se puede utilizar como método de validación de T cómo ráfaga. Otra arista a abordar es la posibilidad de realizar un *mapeo* de T a las *querys* (que contienen a T) que "aportan" a que T sea declarado como ráfaga (basándose en la actividad de clicks). Lo importante en estos enfoques, es idear los mecanismos de tal forma que sean aplicables a un contexto *online*.

Los mismos autores en [28] indican que las ráfagas pueden ser vistas como *outliers* sobre su propia serie de frecuencias previas cuando la frecuencia en el instante actual es desde 1,5 a 2 veces la desviación estándar del promedio de frecuencias previas. Este punto es importante del punto de vista que se otorga información acerca de la utilidad de la desviación estándar como medio de detección de anomalías sobre una serie de frecuencias, sin embargo, utilizar la desviación estándar otorga una perspectiva acerca de la variabilidad de una serie, mas no si esta tiene altas o bajas

frecuencias en su comportamiento normal: puede darse el caso en que una serie con alta frecuencia tenga altas variaciones, pero no por esto la serie corresponderá a una ráfaga.

En [30] Vlachos *et. al* propone un mecanismo *offline* para detección de ráfagas basado en el uso de un *threshold* (el cual es calculado en base al promedio y desviación estándar de la serie de frecuencias de una consulta). Posteriormente este *threshold* es comparado con el promedio móvil de las frecuencias de la serie de la misma consulta. Este método es impracticable en un contexto *online*, debido a que implicaría la necesidad de almacenar las frecuencias de las series de cada una de las consultas, pero sirve como método comparativo, puesto que otorga veracidad sobre dichas series.

Kleinberg en [16], propone un método basado en un autómata finito de dos estados que clasifica ráfagas basado en frecuencias y considerando el comportamiento pasado. Utiliza además una intesidad de ráfaga y con ese parámetro forma una estructura jerárquica de ráfagas basada en la intensidad. Finalmente se prueba que un autómata finito de dos estados es suficiente para identificar las ráfagas. Dejando de lado la debilidad de considerar el comportamiento pasado de las consultas (lo mismo que ocurre con el método de *Vlachos*), este enfoque clasifica las ráfagas basado en las frecuencias, por lo que podría detectar ráfagas pero también sería "poco sutil" a la hora de separar las ráfagas de las consultas permanentes (léase *facebook* o *twiter*). Sin embargo, aquí se da cuenta de otro aspecto a considerar; el uso de frecuencias por sí solo no es un enfoque tan robusto, debido a que se ajusta al comportamiento de las consultas permanentes.

En [12], Qi *et. al* propone un método para representar ráfagas como un *feature* en un clasificador SVM. Ellos basan sus estudios en la propuesta de [16] de utilizar un AFD de dos estados. Los autores proponen que para cada *feature f* en el estado final del AFD (es decir, en el estado que declara el documento como ráfaga), se conserven dos datos relevantes: el promedio de las frecuencias del documento en una ventana de tiempo y la frecuencia más alta del documento. Con esto, los autores encuentran en algunos casos, multiples ráfagas para los distintos *features*. Este punto de vista puede ser útil desde el punto de vista de un método basado en técnicas de inteligencia artificial, lo cual no se ajusta al enfoque de este trabajo.

En [36] Zhu y Shasha proponen un método que utiliza ventanas de tiempo elásticas y una estructura de datos eficiente en tiempo para detectar ráfagas. Ellos proponen que cada ventana de tiempo tenga su propio *treeshold* para declarar ráfagas. Utilizan **promedio y desviación estándar** para como medida de declaración de ráfagas. La relación promedio/desviación estándar utilizada en este trabajo, es un enfoque considerado en este trabajo, puesto que otorga la ventaja de detectar

anomalías en una serie (desviación estándar) pero además permite tener control sobre el comportamiento de la serie en cuanto a frecuencias, por lo que puede aislar las posibles ráfagas de las consultas permanentes.

Klan *et. al* [15] realizan un estudio sobre el trabajo llevado a cabo en [36]. Estos autores se focalizan en el *treshold*, ya que mencionan que hacerlo estático puede no ser lo mejor, puesto que si es muy alto, no detectará ráfagas y si es muy bajo, será demasiado permisivo en la predicción. Los métodos propuestos son *simple smoothing* y *double smoothing*, los cuales hacen más complejo los modelos. Sin embargo, definen su método como lineal en tiempo y espacio en función de las entradas. Este punto de vista no es relevante para este trabajo, puesto que la definición de parámetros de detección (umbrales) se hará en base a experimentación y se asume estático basado en el trabajo propuesto en [11].

En otra perspectiva, el trabajo de Yunliang *et. al* [13], utiliza un método simple para detección de ráfagas en logs de consultas, basado en frecuencias en un instante y la diferencia de esta con respecto al promedio de frecuencias de instantes pasados. Focalizan su trabajo en el uso de bigramas (todas las combinaciones de dos términos de una frase) y unigramas (cada término de una frase). Pese a que se puede utilizar un enfoque de programación dinámica para hacer eficiente el método en términos de espacio (realizando por ejemplo un promedio ponderado en vez de acumular todos los promedios), el enfoque utilizado en este trabajo considera la desviación estándar como medio de detección del grado de anomalía de una serie, más que la diferencia de frecuencias, basado principalmente en la regla práctica del intervalo [1].

En [34] Junjie *et. al* proponen una mejora a los modelos convencionales de detecciones de ráfagas, trabajando sobre un contexto de medios sociales. Los autores proponen que el texto por si sólo, no es un indicador completo a la hora de detectar ráfagas. Entre otros *features*, los autores proponen "metadata frequency" (frecuencia de los metadatos), "topic coverage" (cobertura del tema) y "user attractiveness" (atractivo del usuario). Concluyen que robustizar el modelo (agregando más *features*), aumenta la efectividad de las detecciones. Este enfoque se orienta a métodos de inteligencia artificial, sin embargo, la idea de agregar *features* para robustizar el modelo, se puede homologar a la idea de utilizar la actividad de clicks para validar ráfagas como una manera de mejorar estas detecciones.

[1] La regla práctica del intervalo define que para muchos conjuntos de datos, cerca del 95 % de los valores muestrales se ubican dentro de dos desviaciones estándar de la media [2]

A modo de cierre, los trabajos revisados en este apartado impulsan a la utilización de métodos matemáticos para la detección de anomalías sobre una serie de frecuencias. Se encuentra que utilizar un análisis sobre las frecuencias no es suficiente debido a la imposibilidad de separar las consultas ráfagas de las permanentes. Utilizar la desviación estándar, por si sola, tiene la debilidad de mostrar sólo la variabilidad de una muestra, sin embargo, no indica si dicha variabilidad es ascendente o descendente. Por otro lado, utilizar un enfoque basado en desviación estándar y frecuencias a la vez, permite utilizar lo mejor de ambos métodos, es decir, se puede detectar variabilidad en una muestra y además se puede conocer la intensidad de la consulta en cada instante. Este enfoque requiere que las series sean compactas para ser viable en un contexto *online*, por lo que el uso de ventanas de tiempo móvil es adoptado en este trabajo. En conclusión, este trabajo utiliza un umbral basado en frecuencias y en el "coeficiente de variación" de una ventana de tiempo móvil [2], idealmente compacta.

[2]El coeficiente de variación se define como la razón entre la desviación estándar y la media de un conjunto de datos [2]

Capítulo 6

Desarrollo de la solución

6.1. Comunidades de pregunta/respuesta

6.1.1. Modelo de detección de tópicos en ráfagas

El modelo propuesto en este trabajo para realizar la detección de ráfagas, se enfoca en la detección de un tópico, más que los métodos de detección en ráfaga enfocados en la pregunta. Este enfoque se debe a dos factores principalmente:

1. En una comunidad pregunta/respuesta, predomina el lenguaje natural, por lo que es dificil encontrar *strings* que sean exactamente iguales y esto genera que al momento de mapear frecuencias para las distintas preguntas, estas tiendan a ser bajas.

2. Una pregunta se compone de una serie de términos o tópicos. En un instante determinado, un tópico puede tornarse relevante para los usuarios, por lo que las preguntas acerca de ese tópico pueden tener un alza abrupta. Sin embargo, para un mismo tema de interés, la gramática de las preguntas puede darse de infinitas formas.

Considerando el punto 2, este trabajo busca realizar el proceso inverso en cuanto a la detección de preguntas en ráfaga, es decir, partiendo de un tópico declarado como ráfaga T, mapear a todas aquellas preguntas que "aportaron" a que T fuese declarado como ráfaga.

Contextualización de tópicos

Una limitante importante de orientar la detección de ráfagas a tópicos es que al hacer el procesamiento sobre un tópico, este carece de contexto. Por ejemplo, la palabra *fantasy*, por sí sola no puede entregar la información de si obedece o no a una ráfaga, al contrario de *"fantasy football"* o *"sexual fantasy"* (lo primero es un evento en particular, mientras que lo segundo obedece a una pregunta frecuente).

Este trabajo considera "la señal" producida por un tópico, como el contexto al cual pertenece, es decir, es la serie de frecuencias a través del tiempo lo que sirve como herramienta para declarar o no un tópico como ráfaga.

Basado en lo anterior, para un tópico T, existen n señales, provenientes de n preguntas, que por composición generan la forma de la señal de T. Se identifican por lo menos 3 tipos de señales:

- Señales permanentes.
- Señales periódicas.
- Señales ráfagas.

Si se sabe que una señal se puede componer de la suma de otras señales, el método propuesto toma las señales de tipo ráfaga que aportaron a que T fuese declarado como ráfaga.

Una de las utilidades que brinda el considerar la señal asociada a T como el contexto, es la validación manual de T como ráfaga. El uso de señales representativas también se utiliza en el método MPC, definido en la sección 6.1.5

La figura 6.1, muestra la composición de señales periódicas, permanentes y ráfagas. La señal compuesta, obedece a la señal de un tópico T declarado como ráfaga. La idea del método propuesto es obtener las señales ráfagas que aportaron a la composición de la señal de T como ráfaga.

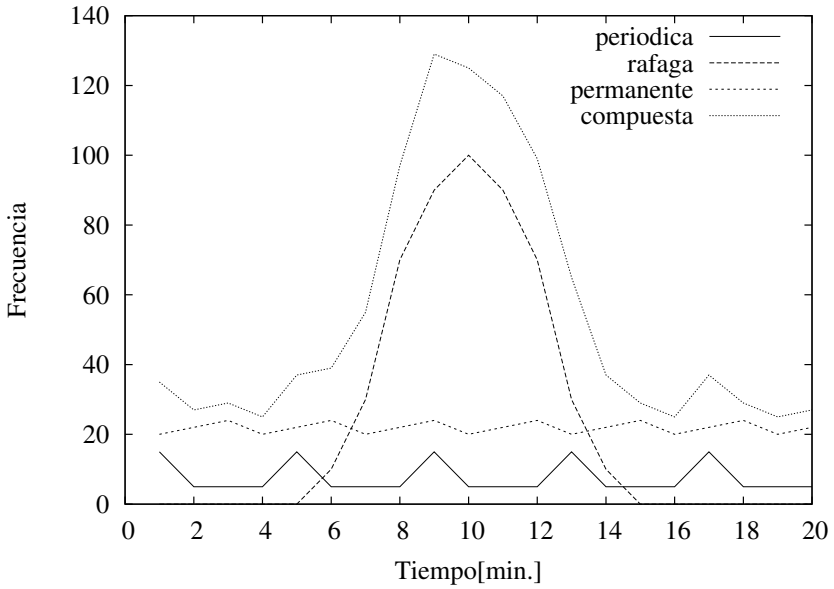

Figura 6.1: Composición de señales. Fuente: elaboración propia.

6.1.2. Características de la entrada

Los algoritmos generados para este contexto, trabajan sobre un cuerpo de datos con las siguientes características:

- Preguntas de usuarios en la comunidad de pregunta/respuesta de Yahoo! Answers. Muestra entre los años 2006 y 2013.
- Datos privados.
- Datos mayoritariamente no estructurados.
- Volúmen: 156 MB, texto plano.
- Cantidad de preguntas: 6,043,460.

La etapa de pre-procesamiento de los datos para este contexto, realiza una serie de filtros, con el fin de estructurar en un cierto grado los datos de entrada de los algoritmos.

Los principales filtros empleados son:

- Quitar aquellas consultas escritas en alfabetos distintos del alfabeto latino (se aceptan pequeñas modificaciones, como letras propias del alfabeto portugués).
- Quitar "stopwords".

El cuerpo de datos se compone de 2 partes; una *Qid*, que corresponde al identificador de la consulta (la cual se compone de la fecha de creación de la consulta concatenada con una clave *hash*) y la consulta en sí.

Las *Qids* son importantes, ya que indican la fecha en la cual se crea la consulta.

6.1.3. Detección de tópicos en ráfagas

El algoritmo 1 muestra la detección de tópicos en ráfaga. Este algoritmo emula un contexto online. Cada línea leída por el algoritmo, representa una consulta con su respectiva *Qid*.

Algoritmo 1 Algoritmo de detección de tópicos en ráfaga

```
1: Define map < termino , map < instante , frecuencia >> mapa_terminos
2: Define TW← 20, Umbral← 20 , X← param, R← param
3: Procedure Detección de tópicos en ráfaga
4:     /* Formar las series de frecuencias para todos los términos de la muestra */
5:     CrearSeries()
6:     /* Procesar las series de frecuencias de todos los términos de la muestra */
7:     Bursty()
```

```
1: Procedure CrearSeries()
2:     instante ←ObtenerInstante( query )
3:     for each token ∈ query do
4:         /* Suma 1 a la frecuencia del término en el instante */
5:         insertar( token )
6:     endFor
7:     /* Cambio instante actual a instante posterior */
8:     if nuevo_instante then
9:         actualizar_instante( )
10:    endIf
```

```
1: Procedure BURSTY( )
2:     Define vector < frecuencias > series
3:     for each termino ∈ mapa_terminos do
4:         i ← 0
5:         while i ≤ TW do
6:             series.insert( i )
7:             i++
8:         endWhile
9:         avg ← promedio( series ) /* Promedio */
10:        sd ← desviacion_estandar( series ) /* Desviación estándar */
11:        cv ← sd/avg /* Coeficiente de variación */
12:        /*Primer filtro: Frecuencia del último elemento de la serie supera UMBRAL*/
13:        if ultimo( serie ) ≥ UMBRAL then
14:            /*Segundo filtro: Considera coeficiente de variación, promedio,
15:            desviación estándar y frecuencia actual de la serie*/
16:            if ultimo( serie ) ≥ ( avg + X∗sd ) && ( cv ≥ R ) then
17:                return true
18:            endIf
19:        endIf
20:        /* Remueve un elemento para una nueva entrada */
21:        series.erase( primer_elemento )
22:    endFor
23:    /* Cambio instante actual a instante posterior */
24:    if nuevo_instante then
25:        actualizar( )
26:    endIf
```

6.1.4. Resultados de detección de tópicos en ráfagas

Experimentalmente, considerando un umbral de detección 20 y una ventana de tiempo del mismo tamaño, se obtienen buenos resultados, por lo que, el modelo se ajusta a ser parametrizable sólo en términos de X y R. No obstante, variar los parámetros X y R del modelo, genera resultados diversos.

El parámetro X indica si la frecuencia del instante actual está X veces la desviación estándar de la serie sobre el promedio de la serie, mientras que el parámetro R indica si la desviación estándar de la serie es R veces el promedio móvil.

La evaluación de los parámetros X y R se hace sobre un intervalo expuesto en [11]. En ese trabajo, se menciona que los mejores valores de X y R, varían entre $[0,5$ - $2,0\ ]$ para X y $[0,1 - 2,0]$ para R. Las tablas 6.1 y 6.2 muestran los resultados obtenidos para distintos valores de X y R, en cuanto a cantidad de tópicos encontrados.

Tabla 6.1: *Cantidad de tópicos candidatos a ráfaga, parte I*

	R									
X	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0
$0,5$	870	870	869	831	575	401	313	286	264	244
$0,6$	870	870	869	830	574	401	313	286	264	244
$0,7$	869	869	868	829	573	400	312	285	263	243
$0,8$	868	868	867	828	571	398	310	284	262	242
$0,9$	868	868	867	826	571	397	310	284	262	242
$1,0$	868	868	867	824	569	396	310	284	262	242
$1,1$	868	868	867	824	568	396	310	284	262	242
$1,2$	868	868	867	820	566	396	310	283	261	241
$1,3$	866	866	865	815	565	395	310	283	259	239
$1,4$	866	866	865	811	562	395	310	283	259	239
$1,5$	866	866	865	806	556	393	309	282	259	239
$1,6$	866	866	865	803	555	392	309	282	259	238
$1,7$	864	864	863	796	549	391	308	280	257	236
$1,8$	863	863	862	782	544	388	306	279	257	235
$1,9$	861	861	860	777	534	381	302	276	255	234
$2,0$	861	861	860	763	529	378	300	273	252	231

Tabla 6.2: *Cantidad de tópicos candidatos a ráfaga, parte II*

	R									
X	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9	2,0
$0,5$	235	213	206	186	169	158	144	134	124	120
$0,6$	235	213	206	186	169	158	144	134	124	120
$0,7$	234	212	205	185	168	157	143	133	123	119
$0,8$	233	211	204	184	167	156	142	132	122	118
$0,9$	233	211	204	184	167	156	142	132	122	118
$1,0$	233	211	204	184	167	156	142	132	122	118
$1,1$	233	211	204	184	167	156	142	132	122	118
$1,2$	231	210	203	183	166	155	141	131	121	117
$1,3$	229	208	201	181	164	153	139	129	119	115
$1,4$	229	208	201	181	164	153	139	129	119	115
$1,5$	229	208	201	181	164	153	139	129	119	115
$1,6$	228	207	200	181	164	153	139	129	119	115
$1,7$	227	206	199	180	163	152	138	128	118	114
$1,8$	226	206	199	180	163	152	138	128	118	114
$1,9$	225	205	198	179	161	150	137	128	118	114
$2,0$	222	202	195	175	157	146	133	124	114	110

Los resultados muestran que a mayor restrictividad en los parámetros X y R (mayores valores), menor es la cantidad de tópicos declarados como ráfaga, es decir, la cantidad de tópicos declarados como ráfaga, es inversamente proporcional a la restrictividad de los parámetros X y R.
Lo que las tablas 6.1 y 6.2 muestran, es la cantidad de tópicos declarados como ráfagas bajo distintas configuraciones de X y R, sin embargo, estos resultados ameritan una segunda revisión, puesto que obtener más tópicos declarados como ráfaga, no indica que los parámetros son mejores.

El análisis *precision/recall*, es utilizado como medio de validación de la calidad de los parámetros X y R.

Raghavan en [22], define *Precisión* como "El ratio entre el número de elementos relevantes recuperados, dividido por el número de elementos recuperados" y *Recall* como "El ratio entre el número de elementos relevantes recuperados, sobre el número total de documentos relevantes".

Entonces, en el contexto actual *Precision* responde a la pregunta: ¿Qué % de tópicos declarados como ráfaga, son realmente tópicos en ráfaga?, mientras que *Recall* responde a la pregunta: ¿Qué % de tópicos en ráfaga es recuperado? Los resultados del análisis, se muestran en las tablas 6.3 6.4 6.5 6.6

Tabla 6.3: *Precision, parte I*

	R									
X	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0
$0,5$	$0,1689$	$0,1689$	$0,1691$	$0,1768$	$0,2556$	$0,364$	$0,4568$	$0,5$	$0,534$	$0,5614$
$0,6$	$0,1689$	$0,1689$	$0,1691$	$0,1771$	$0,256$	$0,364$	$0,4568$	$0,5$	$0,534$	$0,5614$
$0,7$	$0,168$	$0,168$	$0,1682$	$0,1761$	$0,2547$	$0,3625$	$0,4551$	$0,4982$	$0,5323$	$0,5596$
$0,8$	$0,167$	$0,167$	$0,1672$	$0,1751$	$0,2539$	$0,3618$	$0,4548$	$0,4964$	$0,5305$	$0,5578$
$0,9$	$0,167$	$0,167$	$0,1672$	$0,1755$	$0,2539$	$0,3627$	$0,4548$	$0,4964$	$0,5305$	$0,5578$
$1,0$	$0,167$	$0,167$	$0,1672$	$0,1759$	$0,2548$	$0,3636$	$0,4548$	$0,4964$	$0,5305$	$0,5578$
$1,1$	$0,167$	$0,167$	$0,1672$	$0,1759$	$0,2552$	$0,3636$	$0,4548$	$0,4964$	$0,5305$	$0,5578$
$1,2$	$0,167$	$0,167$	$0,1672$	$0,1768$	$0,2561$	$0,3636$	$0,4548$	$0,4946$	$0,5287$	$0,556$
$1,3$	$0,1674$	$0,1674$	$0,1676$	$0,1779$	$0,2566$	$0,3645$	$0,4548$	$0,4946$	$0,525$	$0,5523$
$1,4$	$0,1674$	$0,1674$	$0,1676$	$0,1787$	$0,258$	$0,3645$	$0,4548$	$0,4946$	$0,525$	$0,5523$
$1,5$	$0,1674$	$0,1674$	$0,1676$	$0,1799$	$0,2607$	$0,3664$	$0,4563$	$0,4964$	$0,525$	$0,5523$
$1,6$	$0,1674$	$0,1674$	$0,1676$	$0,1805$	$0,2612$	$0,3673$	$0,4563$	$0,4964$	$0,525$	$0,5546$
$1,7$	$0,1666$	$0,1666$	$0,1668$	$0,1809$	$0,2622$	$0,3657$	$0,4545$	$0,4964$	$0,5252$	$0,555$
$1,8$	$0,1668$	$0,1668$	$0,167$	$0,1841$	$0,2647$	$0,3685$	$0,4575$	$0,4982$	$0,5252$	$0,5574$
$1,9$	$0,166$	$0,166$	$0,1662$	$0,184$	$0,2677$	$0,3727$	$0,4602$	$0,5$	$0,5254$	$0,5555$
$2,0$	$0,166$	$0,166$	$0,1662$	$0,1874$	$0,2703$	$0,3756$	$0,4633$	$0,5054$	$0,5277$	$0,5584$

Tabla 6.4: *Precision, parte II*

	R									
X	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9	2,0
$0,5$	$0,5787$	$0,5962$	$0,6067$	$0,6236$	$0,6508$	$0,6772$	$0,7152$	$0,7388$	$0,75$	$0,75$
$0,6$	$0,5787$	$0,5962$	$0,6067$	$0,6236$	$0,6508$	$0,6772$	$0,7152$	$0,7388$	$0,75$	$0,75$
$0,7$	$0,5769$	$0,5943$	$0,6048$	$0,6216$	$0,6488$	$0,6751$	$0,7132$	$0,7368$	$0,7479$	$0,7478$
$0,8$	$0,5751$	$0,5924$	$0,6029$	$0,6195$	$0,6467$	$0,673$	$0,7112$	$0,7348$	$0,7459$	$0,7457$
$0,9$	$0,5751$	$0,5924$	$0,6029$	$0,6195$	$0,6467$	$0,673$	$0,7112$	$0,7348$	$0,7459$	$0,7457$
1	$0,5751$	$0,5924$	$0,6029$	$0,6195$	$0,6467$	$0,673$	$0,7112$	$0,7348$	$0,7459$	$0,7457$
$1,1$	$0,5775$	$0,5924$	$0,6029$	$0,6195$	$0,6467$	$0,673$	$0,7112$	$0,7348$	$0,7459$	$0,7457$
$1,2$	$0,5757$	$0,5904$	$0,6009$	$0,6174$	$0,6445$	$0,6709$	$0,7092$	$0,7328$	$0,7438$	$0,7435$
$1,3$	$0,572$	$0,5865$	$0,597$	$0,6132$	$0,6402$	$0,6666$	$0,705$	$0,7286$	$0,7394$	$0,7391$
$1,4$	$0,572$	$0,5865$	$0,597$	$0,6132$	$0,6402$	$0,6666$	$0,705$	$0,7286$	$0,7394$	$0,7391$
$1,5$	$0,572$	$0,5865$	$0,597$	$0,6132$	$0,6402$	$0,6666$	$0,705$	$0,7286$	$0,7394$	$0,7391$
$1,6$	$0,5745$	$0,5893$	$0,6$	$0,6132$	$0,6402$	$0,6666$	$0,705$	$0,7286$	$0,7394$	$0,7391$
$1,7$	$0,5726$	$0,5873$	$0,5979$	$0,6111$	$0,638$	$0,6644$	$0,7028$	$0,7265$	$0,7372$	$0,7368$
$1,8$	$0,5752$	$0,5873$	$0,5979$	$0,6111$	$0,638$	$0,6644$	$0,7028$	$0,7265$	$0,7372$	$0,7368$
$1,9$	$0,5733$	$0,5853$	$0,5959$	$0,6089$	$0,6397$	$0,66$	$0,7007$	$0,7265$	$0,7372$	$0,7368$
2	$0,5765$	$0,5891$	$0,6$	$0,6171$	$0,6496$	$0,6712$	$0,7142$	$0,7419$	$0,7543$	$0,7368$

Tabla 6.5: *Recall, parte I*

	R									
X	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0
$0,5$	1	1	1	1	1	$0,9931$	$0,9727$	$0,9727$	$0,9591$	$0,9319$
$0,6$	1	1	1	1	1	$0,9931$	$0,9727$	$0,9727$	$0,9591$	$0,9319$
$0,7$	$0,9931$	$0,9931$	$0,9931$	$0,9931$	$0,9931$	$0,9863$	$0,9659$	$0,9659$	$0,9523$	$0,9251$
$0,8$	$0,9863$	$0,9863$	$0,9863$	$0,9863$	$0,9863$	$0,9795$	$0,9591$	$0,9591$	$0,9455$	$0,9183$
$0,9$	$0,9863$	$0,9863$	$0,9863$	$0,9863$	$0,9863$	$0,9795$	$0,9591$	$0,9591$	$0,9455$	$0,9183$
1	$0,9863$	$0,9863$	$0,9863$	$0,9863$	$0,9863$	$0,9795$	$0,9591$	$0,9591$	$0,9455$	$0,9183$
$1,1$	$0,9863$	$0,9863$	$0,9863$	$0,9863$	$0,9863$	$0,9795$	$0,9591$	$0,9591$	$0,9455$	$0,9183$
$1,2$	$0,9863$	$0,9863$	$0,9863$	$0,9863$	$0,9863$	$0,9795$	$0,9591$	$0,9523$	$0,9387$	$0,9115$
$1,3$	$0,9863$	$0,9863$	$0,9863$	$0,9863$	$0,9863$	$0,9795$	$0,9591$	$0,9523$	$0,9251$	$0,8979$
$1,4$	$0,9863$	$0,9863$	$0,9863$	$0,9863$	$0,9863$	$0,9795$	$0,9591$	$0,9523$	$0,9251$	$0,8979$
$1,5$	$0,9863$	$0,9863$	$0,9863$	$0,9863$	$0,9863$	$0,9795$	$0,9591$	$0,9523$	$0,9251$	$0,8979$
$1,6$	$0,9863$	$0,9863$	$0,9863$	$0,9863$	$0,9863$	$0,9795$	$0,9591$	$0,9523$	$0,9251$	$0,8979$
$1,7$	$0,9795$	$0,9795$	$0,9795$	$0,9795$	$0,9795$	$0,9727$	$0,9523$	$0,9455$	$0,9183$	$0,8911$
$1,8$	$0,9795$	$0,9795$	$0,9795$	$0,9795$	$0,9795$	$0,9727$	$0,9523$	$0,9455$	$0,9183$	$0,8911$
$1,9$	$0,9727$	$0,9727$	$0,9727$	$0,9727$	$0,9727$	$0,9659$	$0,9455$	$0,9387$	$0,9115$	$0,8843$
2	$0,9727$	$0,9727$	$0,9727$	$0,9727$	$0,9727$	$0,9659$	$0,9455$	$0,9387$	$0,9047$	$0,8775$

Tabla 6.6: *Recall, parte II*

	R									
X	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9	2,0
$0,5$	$0,9251$	$0,8639$	$0,8503$	$0,7891$	$0,7482$	$0,7278$	$0,7006$	$0,6734$	$0,6326$	$0,6122$
$0,6$	$0,9251$	$0,8639$	$0,8503$	$0,7891$	$0,7482$	$0,7278$	$0,7006$	$0,6734$	$0,6326$	$0,6122$
$0,7$	$0,9183$	$0,8571$	$0,8435$	$0,7823$	$0,7414$	$0,721$	$0,6938$	$0,6666$	$0,6258$	$0,6054$
$0,8$	$0,9115$	$0,8503$	$0,8367$	$0,7755$	$0,7346$	$0,7142$	$0,687$	$0,6598$	$0,619$	$0,5986$
$0,9$	$0,9115$	$0,8503$	$0,8367$	$0,7755$	$0,7346$	$0,7142$	$0,687$	$0,6598$	$0,619$	$0,5986$
1	$0,9115$	$0,8503$	$0,8367$	$0,7755$	$0,7346$	$0,7142$	$0,687$	$0,6598$	$0,619$	$0,5986$
$1,1$	$0,9115$	$0,8503$	$0,8367$	$0,7755$	$0,7346$	$0,7142$	$0,687$	$0,6598$	$0,619$	$0,5986$
$1,2$	$0,9047$	$0,8435$	$0,8299$	$0,7687$	$0,7278$	$0,7074$	$0,6802$	$0,653$	$0,6122$	$0,5918$
$1,3$	$0,8911$	$0,8299$	$0,8163$	$0,7551$	$0,7142$	$0,6938$	$0,6666$	$0,6394$	$0,5986$	$0,5782$
$1,4$	$0,8911$	$0,8299$	$0,8163$	$0,7551$	$0,7142$	$0,6938$	$0,6666$	$0,6394$	$0,5986$	$0,5782$
$1,5$	$0,8911$	$0,8299$	$0,8163$	$0,7551$	$0,7142$	$0,6938$	$0,6666$	$0,6394$	$0,5986$	$0,5782$
$1,6$	$0,8911$	$0,8299$	$0,8163$	$0,7551$	$0,7142$	$0,6938$	$0,6666$	$0,6394$	$0,5986$	$0,5782$
$1,7$	$0,8843$	$0,8231$	$0,8095$	$0,7482$	$0,7074$	$0,687$	$0,6598$	$0,6326$	$0,5918$	$0,5714$
$1,8$	$0,8843$	$0,8231$	$0,8095$	$0,7482$	$0,7074$	$0,687$	$0,6598$	$0,6326$	$0,5918$	$0,5714$
$1,9$	$0,8775$	$0,8163$	$0,8027$	$0,7414$	$0,7006$	$0,6734$	$0,653$	$0,6326$	$0,5918$	$0,5714$
2	$0,8707$	$0,8095$	$0,7959$	$0,7346$	$0,6938$	$0,6666$	$0,6462$	$0,6258$	$0,585$	$0,5714$

Para comprender de mejor manera los resultados expuestos en las tablas 6.3, 6.4, 6.5 y 6.6, se emplea la medida "$F1 - Score$". *Raghavan* en [22] presenta la medida como "El ratio entre el producto de *precision* y *recall*; y la suma de los mismos. Todo esto amplificado por 2".

$$F1\text{-}Score = 2\,\frac{precision * recall}{precision + recall}$$

La medida $F1 - Score$ realiza una ponderación entre *precision* y *recall*. Los valores que tienden a 1, son los mejores $F1 - Scores$, mientras que los tendientes a 0, son los peores.

Las tablas 6.7 y 6.8 muestran los resultados para las distintas configuraciones de X y R, considerando la medida $F1 - Score$.

Tabla 6.7: *F1-Score, parte I*

	R									
X	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0
$0,5$	$0,289$	$0,289$	$0,2893$	$0,3006$	$0,4072$	$0,5328$	$0,6217$	$0,6605$	$0,6861$	$0,7007$
$0,6$	$0,289$	$0,289$	$0,2893$	$0,3009$	$0,4077$	$0,5328$	$0,6217$	$0,6605$	$0,6861$	$0,7007$
$0,7$	$0,2874$	$0,2874$	$0,2876$	$0,2991$	$0,4055$	$0,5301$	$0,6187$	$0,6574$	$0,6829$	$0,6974$
$0,8$	$0,2857$	$0,2857$	$0,2859$	$0,2974$	$0,4038$	$0,5284$	$0,617$	$0,6542$	$0,6797$	$0,694$
$0,9$	$0,2857$	$0,2857$	$0,2859$	$0,298$	$0,4038$	$0,5294$	$0,617$	$0,6542$	$0,6797$	$0,694$
1	$0,2857$	$0,2857$	$0,2859$	$0,2986$	$0,405$	$0,5303$	$0,617$	$0,6542$	$0,6797$	$0,694$
$1,1$	$0,2857$	$0,2857$	$0,2859$	$0,2986$	$0,4055$	$0,5303$	$0,617$	$0,6542$	$0,6797$	$0,694$
$1,2$	$0,2857$	$0,2857$	$0,2859$	$0,2998$	$0,4067$	$0,5303$	$0,617$	$0,6511$	$0,6764$	$0,6907$
$1,3$	$0,2862$	$0,2862$	$0,2865$	$0,3014$	$0,4073$	$0,5313$	$0,617$	$0,6511$	$0,6699$	$0,6839$
$1,4$	$0,2862$	$0,2862$	$0,2865$	$0,3027$	$0,409$	$0,5313$	$0,617$	$0,6511$	$0,6699$	$0,6839$
$1,5$	$0,2862$	$0,2862$	$0,2865$	$0,3043$	$0,4125$	$0,5333$	$0,6184$	$0,6526$	$0,6699$	$0,6839$
$1,6$	$0,2862$	$0,2862$	$0,2865$	$0,3052$	$0,4131$	$0,5343$	$0,6184$	$0,6526$	$0,6699$	$0,6857$
$1,7$	$0,2848$	$0,2848$	$0,2851$	$0,3054$	$0,4137$	$0,5315$	$0,6153$	$0,651$	$0,6683$	$0,684$
$1,8$	$0,2851$	$0,2851$	$0,2854$	$0,31$	$0,4167$	$0,5345$	$0,6181$	$0,6525$	$0,6683$	$0,6858$
$1,9$	$0,2837$	$0,2837$	$0,284$	$0,3095$	$0,4199$	$0,5378$	$0,6191$	$0,6524$	$0,6666$	$0,6824$
2	$0,2837$	$0,2837$	$0,284$	$0,3142$	$0,423$	$0,5409$	$0,6219$	$0,6571$	$0,6666$	$0,6825$

Tabla 6.8: *F1-Score, parte II*

	R									
X	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9	2,0
$0,5$	$0,712$	$0,7055$	$0,7082$	$0,6966$	$0,6962$	$0,7016$	$0,7079$	$0,7046$	$0,6863$	$0,6741$
$0,6$	$0,712$	$0,7055$	$0,7082$	$0,6966$	$0,6962$	$0,7016$	$0,7079$	$0,7046$	$0,6863$	$0,6741$
$0,7$	$0,7086$	$0,7019$	$0,7045$	$0,6927$	$0,692$	$0,6973$	$0,7034$	$0,7$	$0,6814$	$0,6691$
$0,8$	$0,7052$	$0,6983$	$0,7008$	$0,6888$	$0,6878$	$0,693$	$0,6989$	$0,6953$	$0,6765$	$0,6641$
$0,9$	$0,7052$	$0,6983$	$0,7008$	$0,6888$	$0,6878$	$0,693$	$0,6989$	$0,6953$	$0,6765$	$0,6641$
1	$0,7052$	$0,6983$	$0,7008$	$0,6888$	$0,6878$	$0,693$	$0,6989$	$0,6953$	$0,6765$	$0,6641$
$1,1$	$0,7071$	$0,6983$	$0,7008$	$0,6888$	$0,6878$	$0,693$	$0,6989$	$0,6953$	$0,6765$	$0,6641$
$1,2$	$0,7037$	$0,6946$	$0,6971$	$0,6848$	$0,6837$	$0,6887$	$0,6944$	$0,6906$	$0,6716$	$0,659$
$1,3$	$0,6968$	$0,6873$	$0,6896$	$0,6768$	$0,6752$	$0,68$	$0,6853$	$0,6811$	$0,6616$	$0,6488$
$1,4$	$0,6968$	$0,6873$	$0,6896$	$0,6768$	$0,6752$	$0,68$	$0,6853$	$0,6811$	$0,6616$	$0,6488$
$1,5$	$0,6968$	$0,6873$	$0,6896$	$0,6768$	$0,6752$	$0,68$	$0,6853$	$0,6811$	$0,6616$	$0,6488$
$1,6$	$0,6986$	$0,6892$	$0,6916$	$0,6768$	$0,6752$	$0,68$	$0,6853$	$0,6811$	$0,6616$	$0,6488$
$1,7$	$0,6951$	$0,6855$	$0,6878$	$0,6727$	$0,6709$	$0,6755$	$0,6807$	$0,6763$	$0,6566$	$0,6436$
$1,8$	$0,697$	$0,6855$	$0,6878$	$0,6727$	$0,6709$	$0,6755$	$0,6807$	$0,6763$	$0,6566$	$0,6436$
$1,9$	$0,6935$	$0,6818$	$0,684$	$0,6687$	$0,6688$	$0,6666$	$0,676$	$0,6763$	$0,6566$	$0,6436$
2	$0,6937$	$0,6819$	$0,6842$	$0,6708$	$0,671$	$0,6689$	$0,6785$	$0,6789$	$0,659$	$0,6436$

Por experimentación, entonces, se definen los parámetros ideales como:

- ventana de tiempo: $TW = 20$
- umbral de frecuencia: $UMBRAL = 20$
- $X = 0,7$
- $R = 1,1$

Con lo anterior se obtiene que existen 234 tópicos ráfaga, mientras que un procesamiento por inspección manual, entrega 190 tópicos ráfaga. La diferencia se debe a que existen series permanentes, que contienen ráfagas dentro de sí mismas. Estas series podrían llegar a ser tomadas como ráfaga por el método propuesto, mientras que la inspección manual es más estricta en ese sentido. Este problema no es relevante desde el punto de vista de que la detección de ráfagas se hace pensando en que este tipo de consultas puede ser perjudicial para el desempeño de un motor de búsqueda

y por tanto debiese estar contemplada por un sistema de *caché* como prioritaria, del mismo modo que las consultas permanentes, es decir, para un sistema de *caché*, tanto las consultas en ráfaga como las permanentes, debiesen ser prioritarias.

La figura 6.2 expone una muestra de la señal de un tópico permanente con ráfagas dentro de sí.

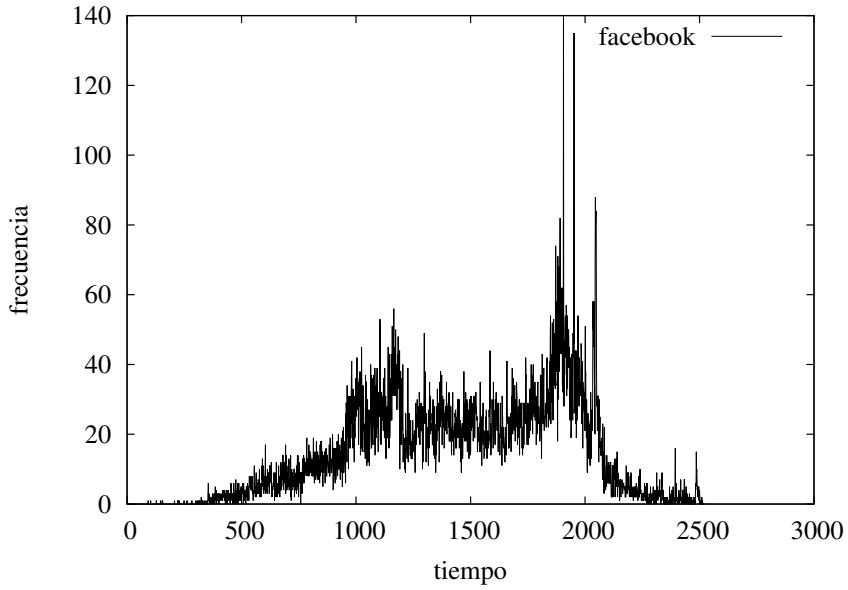

Figura 6.2: tópico permanente-ráfaga

Las ráfagas se producen en torno a los días 1800 - 2100 aprox., que según la data con la cual se cuenta, significa desde el mes de Mayo del año 2011, hasta Marzo del año 2012.

Según datos provistos en [9], a finales del año 2011 se lanza *facebook messenger for windows*. Este nuevo producto se correlaciona con una de las ráfagas apreciadas en la figura; como se describe antes, es un evento ráfaga dentro de una señal permanente.

A finales del mes de Septiembre, Facebook lanza *timeline* [10] (un servicio que emula una biografía digital). Esto justificaría otra de las ráfagas.

Los dos ejemplos anteriores, se exponen a modo de justificación de ráfagas sobre señales permanentes. Como se menciona anteriormente, este tipo de eventos no son relevantes en este trabajo, puesto que al ser señales permanentes, por sí solas debiesen ser tratadas de forma diferente.

Instantes de detección

Este apartado, muestra de forma gráfica, la detección de algunos tópicos en ráfaga, utilizando el método propuesto en este trabajo.

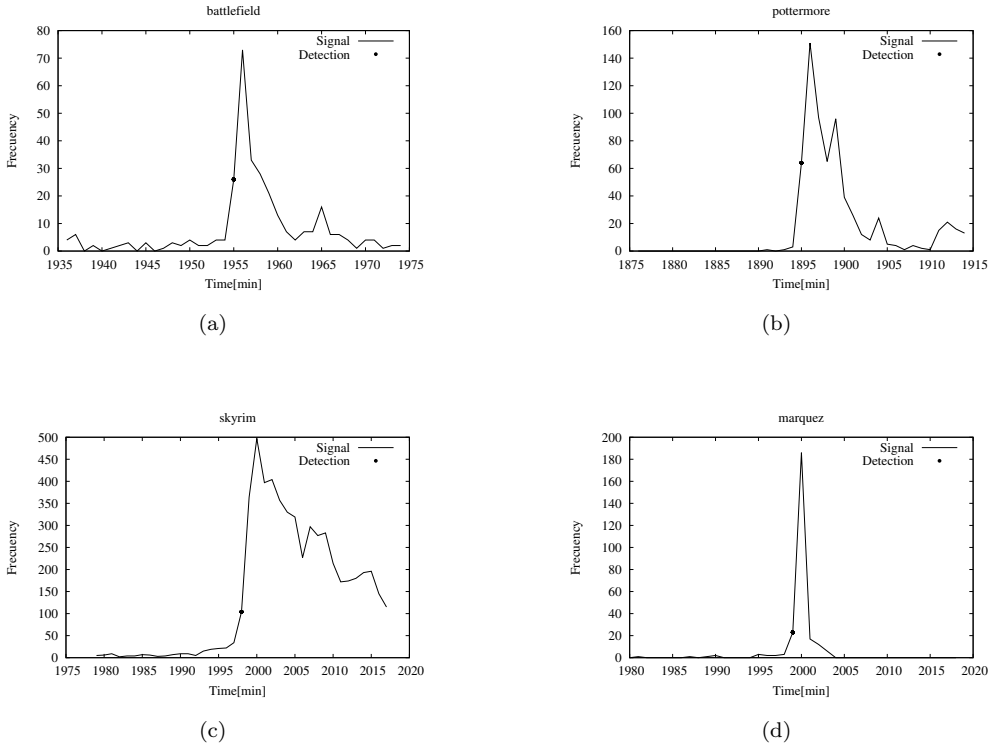

Figura 6.3: Muestras de series de frecuencias e instantes de detección de ráfagas. Fuente: elaboración propia

La figura 6.3 muestra cómo el método propuesto se adelanta a la máxima frecuencia de la serie, declarándo las ráfagas mucho antes de alcanzar la cota superior.

Resultados obtenidos contra método comparativo

El método a comparar con el mecanismo propuesto, es el método propuesto por *Vlachos et. al* [30].

En dicho trabajo, los autores plantean un mecanismo *offline* para la detección de ráfagas, basándose en el uso de *threeshold* calculado de forma *offline*. Este *treeshold* se calcula como el promedio de los promedios móviles de la serie de tamaño tw. Con este umbral ya encontrado, se realiza un segundo paso por la serie de frecuencias. El criterio de detección de ráfagas se da cuando un promedio móvil (también de tamaño tw) supera en X unidades la desviación estándar de los promedios móviles de la serie (calculada al momento de obtener el *treeshold*) más el promedio de promedios móviles.

Este método resulta no ser eficiente a la hora de detectar ráfagas, puesto que suele detectarlas cuando éstas ya han alcanzado su máxima frecuencia, pero si sirve como un método para compararse, debido a que almacena la información de toda la serie de frecuencias y por ende, sus predicciones suelen ser verdaderos positivos (sobretodo al trabajar con valores altos de X).

Para este experimento se utiliza tw=20 (para guardar relación con la ventana de tiempo del método propuesto) y X=2, para evitar caer en falsos positivos (se aumenta restrictividad de detección).

Como se observa, el método de Vlacho [30], no es preciso a la hora de hacer detecciones tempranas de ráfagas, sino que puede incluso detectar anomalías, una vez pasado el momento de *peak* de frecuencias.

6.1.5. Tópicos en ráfaga y actividad de clicks

La actividad de clicks asociada a una Qid otorga otra perspectiva acerca del comportamiento de un tópico t. Si se sabe un tópico declarado como ráfaga proviene de un conjunto de preguntas (representadas por sus respectivas $Qids$), se puede generar un método eficiente en tiempo y espacio, capaz de analizar la actividad de clicks del conjunto de $Qids$ y por medio de esto validar a t como un tópico emergente. La figura 6.8 da cuenta de la idea general del método de validación.

Sin embargo, experimentalmente se observa que la cantidad de $Qids$ asociadas a un término T, puede llegar al orden de miles y la actividad de clicks para cada Qid es una serie de valores enteros que podría llegar a ser muy grande, (dependiendo del intervalo de tiempo durante el cual

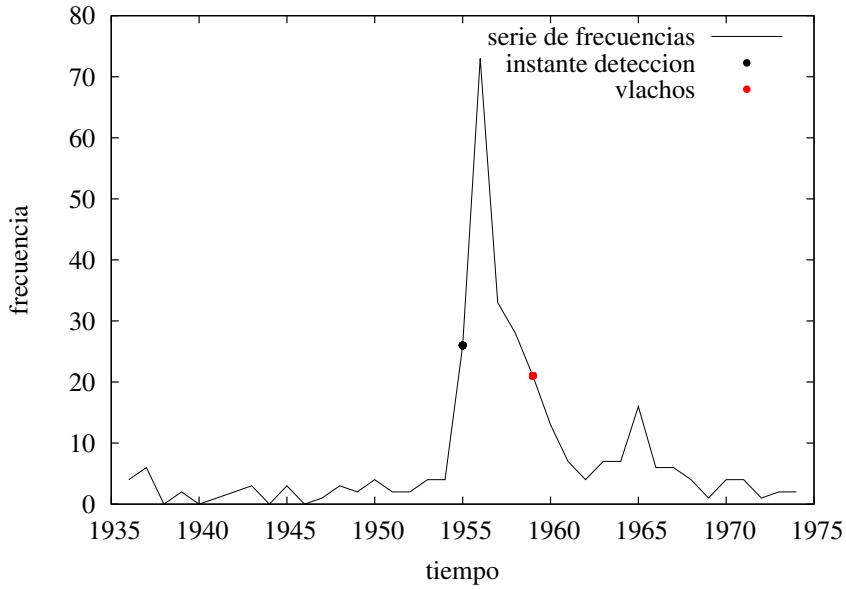

Figura 6.4: tópico ráfaga battlefield, método propuesto vs Vlachos

se almacenen los clicks de los usuarios) por lo que realizar un análisis sobre las series completas de de clicks todas las $Qids$, resulta ser inviable en un contexto online.

Se plantea entonces, un método capaz de utilizar las actividades de clicks asociadas a las $Qids$ asociadas a su vez a T, eficiente en tiempo y espacio y enfocado a dos puntos claves:

1. Validar a un término T como un tópico en ráfaga.
2. Encontrar la(s) pregunta(s) que más aporta(n) a que T sea declarado como ráfaga, o sea, las preguntas relevantes.

El primer punto es importante puesto que si se encuentra que T corresponde a un tópico en ráfaga, se puede levantar un *flag* que indique un especial cuidado con las nuevas preguntas que contengan a T. El segundo punto resulta importante desde el punto de vista del motor de búsqueda, puesto que, como se plantea en [35] , al tener identificadas las preguntas relevantes, internamente un conjunto de consultas Cq que contengan a T, pueden ser asociadas por el motor de búsqueda a las consultas relevantes y de ese modo, modificar el orden de despliegue de los resultados.

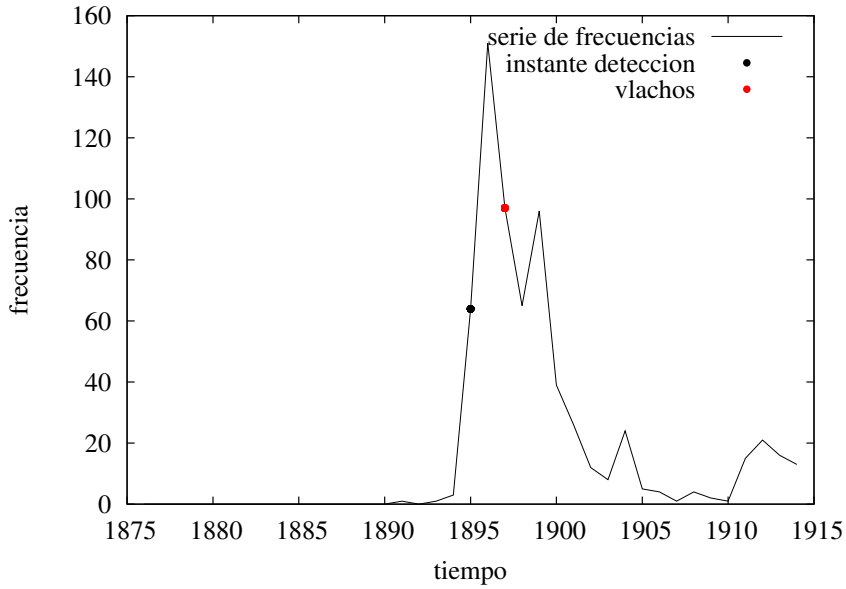

Figura 6.5: tópico ráfaga pottermore, método propuesto vs Vlachos

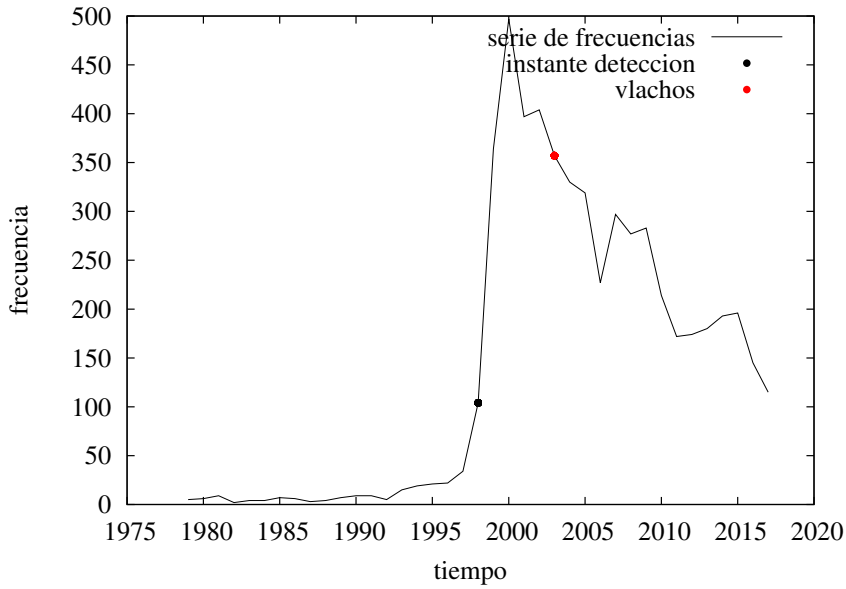

Figura 6.6: tópico ráfaga skyrim, método propuesto vs Vlachos

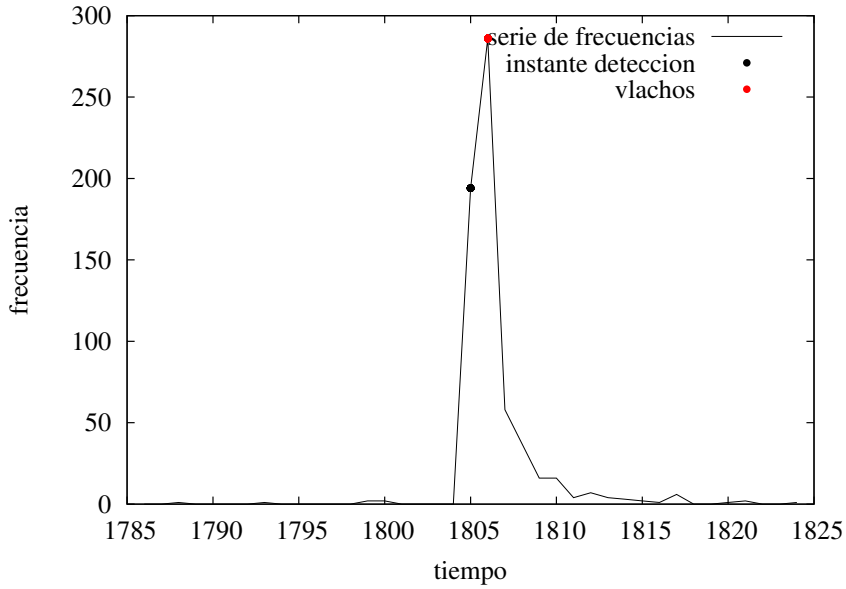

Figura 6.7: tópico ráfaga bin, método propuesto vs Vlachos

Método de procesamiento de clicks: MPC

Las ráfagas son señales con frecuencias casi nulas en su comportamiento normal y luego, en periodos de tiempo muy acotados sufren alzas abruptas en sus frecuencias para posteriormente exponer un decaimiento gradual en la frecuencia de su señal . Considerando esto, siempre que se hace una detección de ráfagas, se tiene un instante de tiempo t_i que marca el punto de inflexión entre declarar o no una ráfaga (en este trabajo, ese punto de inflexión se considera al sobrepasar un umbral de frecuencias mínimo predefinido). Entonces se tiene:

$$\forall\ T\ \textit{declarado como ráfaga}, \exists\ t_i \mid T.\text{frec}(t_i) \geq \textit{umbral}$$

El método propuesto MPC, considera la existencia de t_i para analizar las series de clicks de las $Qids$ asociadas a T. Además, se considera el uso de una ventana de tiempo reducida tw para procesar las series de clicks de cada Qid. La ventana de tiempo a analizar es estática (a diferencia de la ventana de tiempo móvil empleada en el método 6.1.3), y vá desde t_i hasta $t_i - tw$, siendo tw un valor pequeño. La figura 6.9 muestra la lógica del método MPC.

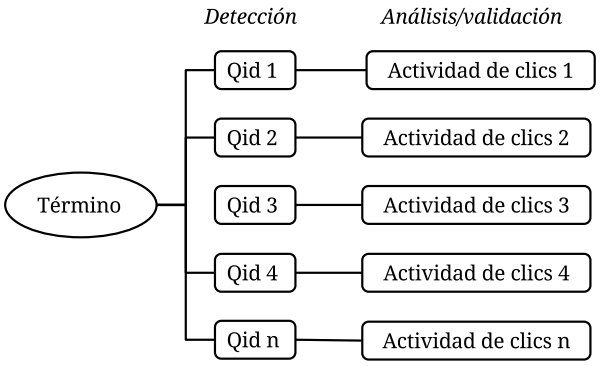

Figura 6.8: idea general método de validación

La forma de hacer eficiente este mecanismo, es utilizar un índice invertido, para asociar a cada tópico T declarado como ráfaga, todas las $Qids$ donde T es un $Token$ de la pregunta completa. De esta manera, se obtienen las preguntas asociadas a T de forma rápida.

Luego, se debe acceder a la actividad de clicks asociada a cada Qid **considerando el instante** t_i, puesto que la revisión de la actividad de clicks se hace sobre una ventana reducida desde el punto de detección de la ráfaga, hacia atrás (en un entorno real, es imposible obtener la actividad de clicks desde t_i hacia adelante). Lo anterior, está pensado considerando el planteamiento expuesto en [27], donde, según la experimentación de los autores, la actividad de clicks se aglomera durante los episodios de ráfagas, mas que antes y después de estos.

En la figura 6.9, las $Qids$ 1, 2 y n son tomadas en cuenta por el método, debido a que la creación de dichas consultas es anterior a t_i (cuando se declara T como ráfaga). Por otro lado, la Qid m, no es considerada por el método, debido a que la creación de la consulta ocurre en un instante de tiempo posterior a t_i, es decir, dada la orientación de este trabajo de detectar y validar un término declarado como ráfaga en un entorno *online*, la pregunta asociada a la Qid m nisiquiera existiría en t_i, por lo cual es imposible que m haya aportado a la señal de T.

Entonces, dado T un tópico declarado como ráfaga en el instante t_i y $CQids$ el conjunto de $Qids$ asociado a T que cumplen la restricción de contener a t_i; el método MPC, se define:

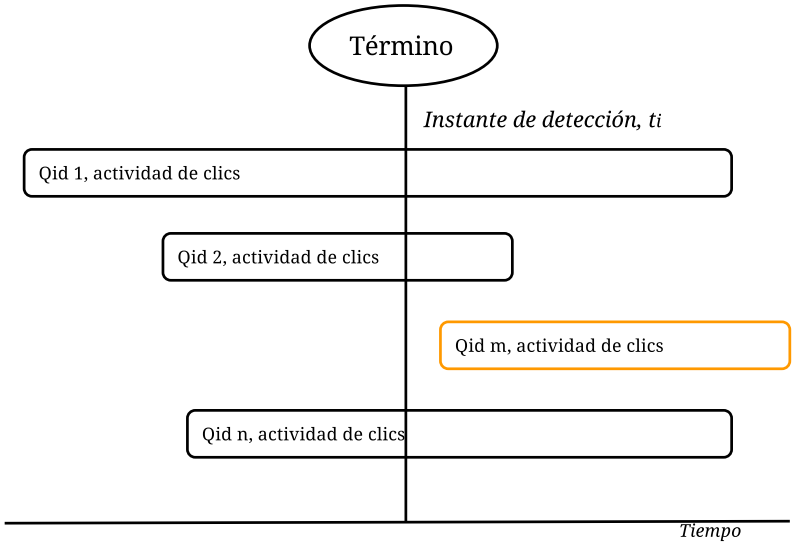

Figura 6.9: método MPC

$\forall\ CQids_j$, con $j = 1, 2, .., size(Cqids)$.
Obtener una serie de clicks desde t_i hasta $t_i - tw$.
Procesar cada serie de clicks.
Obtener (máxima frecuencia) de clicks de $CQids_j$.
Generar una serie de máximas frecuencias de clicks.
Procesar la serie de máximas frecuencias de clicks.

El método MPC, brinda información en dos sentidos, definidos como *ganancia vertical* y *ganancia horizontal*.

La *ganancia vertical*, es la información que se puede extraer a través de la serie generada a partir de la máxima frecuencia de clicks para todas las $Qids$ asociadas a T. Esta serie puede ser vista como la señal representativa de T y sirve para validar a T o no como un tópico en ráfaga.

El método *naive*, sería marcar todas las $Qids$ asociadas a T como importantes, sin embargo, la *ganancia horizontal* permite definir cuales $Qids$ son las que aportan a que la señal de T sea de tipo ráfaga, es decir, la *ganancia horizontal* se emplea como método para descomponer la señal de T en sus correspondientes señales y extraer las señales ráfagas de éstas (orientado a la actividad

de clicks). la *ganancia horizontal* se define como la ganancia que entrega el procesamiento de la serie de cada Qid asociada a T. Como estas series son pequeñas (tamaño tw), es factible realizar el procesamiento de cada una de ellas.

En los análisis experimentales, se utiliza tw=10. El algoritmo empleado para revisar la *ganancia vertical*, realiza un recorrido de la señal generada en orden n (O(n)), puesto que, la cantidad de datos a procesar puede llegar al orden de miles, dependiendo de la cantidad de $Qids$ asociadas a T, por lo que se requiere un método eficiente en ese sentido. Lo que se realiza es buscar **al menos un dato que supere un umbral predefinido**. Si se da el caso, significa que existe al menos una serie de clicks para una Qid que se mostró un aglutinamiento en torno al episodio ráfaga.

Para obtener la *ganancia horizontal*, se emplea un método simple, similar a lo propuesto en [30] por *Vlachos et. al*, puesto que, lo importante no es hacer una detección temprana, sino que encontrar las $Qids$ que aportan a que T sea un tópico ráfaga. Además, como se menciona antes, se utiliza una ventana reducida tw=10, por lo que emplear este método, no implica un impacto en el desempeño.

El algoritmo 2, muestra el funcionamiento del método de procesamieno de clicks. En este algoritmo, se supone la data ya filtrada.

Algoritmo 2 MPC

```
1: Define multimap < termino , qid > mapa_terminos
2: Define tw← 10, Umbral← 20 , X← param, R← param
3: Procedure MPC
4:     /* Generar índice invertido [termino , qids] */
5:     indice_invertido( )
6:     /* Procesar las qids asociadas a los términos, obtener ganancia horizontal y vertical */
7:     Procesar( )
```

```
1: Procedure INDICE_INVERTIDO()
2:     Define map < terminos_ráfaga , instante_detección > validados
3:     for each Qid do
4:         for each token ∈ Query( Qid ) do
5:             if iterator::iter ← validados.find( token ) then
6:                 mapa_terminos.insert( iter , Qid )
7:             endIf
8:         endFor
9:     endFor
```

```
1:  Procedure PROCESS()
2:      Define vector < frecuencias > serie
3:      Define vector < frecuencias > serie_max
4:      Define ganancia_horizontal, ganancia_vertical
5:      for each  t ∈ validados  do
6:          serie ← clear( )
7:          serie_max ← clear( )
8:          ganancia_horizontal ← 0
9:          ganancia_vertical ← 0
10:         instante ← ObtenerInstante( t )
11:         termino ← t
12:         for each q ∈ mapa_terminos.find( termino ) do
13:             /* Obtengo la serie de largo tw desde el instante hacia atrás */
14:             serie ← obtener_tw__desde( instante )
15:             /* Inserción del máximo número de la serie en la serie de máximos (G. Vertical) */
16:             max ← max( instante , tw )
17:             serie_max.insert( max )
18:             /* Calculo la ganancia horizontal */
19:             avg ← promedio( serie )
20:             de ← desviacion_estandar( serie )
21:             /* Se define un umbral de frecuencias ínfimo */
22:             if  max>10 && max > avg + 2*de  then
23:                 ganancia_horizontal++
24:             endIf
25:         endFor
26:         /* Calculo la ganancia vertical */
27:         for each  dato ∈ serie_max  do
28:             if  dato > threshold then
29:                 return true
30:             endIf
31:         endFor
32:     endFor
```

Resultados de método propuesto

Para hacer más comprensible el método MPC, la tabla 6.9 ilustra la lógica del método.

Tabla 6.9: Método MPC, tipos de ganancias

Qid	serie de clicks (ganancia horizontal)	serie de máximos (ganancia vertical)
Qid_1	0 0 0 0 0 0 0 0 0 0 2	2
Qid_2	0 0 0 0 0 0 0 0 0 0 1	1
Qid_3	0 0 0 0 0 0 0 0 0 0 3	3
Qid_4	0 0 0 0 0 0 0 0 0 0 4	4
Qid_5	0 0 1 0 3 4 3 1 8 34	34
Qid_6	0 0 0 1 0 0 0 0 0 82	82
Qid_7	0 0 0 0 1 0 7 6 6 34	34
Qid_8	0 1 0 2 1 0 1 0 1 36	36
Qid_9	0 0 0 0 0 1 0 0 0 2	2
Qid_10	0 0 0 0 0 0 0 0 0 2	2
Qid_11	0 0 0 1 0 0 0 0 0 99	99
Qid_12	0 0 0 0 0 0 1 0 1 56	56
Qid_13	0 0 0 0 0 2 1 1 1 27	27
Qid_14	0 0 0 1 1 3 2 2 2 246	246
Qid_15	0 0 0 0 0 0 0 0 0 3	3

La tabla 6.9 expone **una muestra** de series de 15 *Qids* asociadas al tópico "activity". En este caso, las *Qids* 5, 6, 7, 8, 11, 12, 13 y 14, otorgan ganancia horizontal, puesto que las series cumplen con la restricción propuesta por el método. Esto significa que esas *Qids*, son las que hay que supervisar, por tener relación con la ráfaga de "activity". Además, la ganancia vertical, valida a T como ráfaga, puesto que la serie de máximos que se genera, cumple con las restricciones del método.

De este modo, el método cumple con el objetivo de (i) validar a un tópico como ráfaga y (ii) mapear el tópico a las preguntas que provocan que sea ráfaga.

Los resultados de la ejecución del algoritmo MPC, se detallan en la tabla 6.10. La cantidad de tópicos ráfagas procesados para esta prueba, se reduce, debido a que no se cuenta con la data necesaria para utilizar todos los tópicos declarados como ráfaga en etapas previas.

Debido a que no se cuenta con la totalidad de los datos de clicks asociados a las $Qids$, el cuerpo de tópicos se reduce de los 190 validados.

El primer filtro, es obtener los tópicos declarados como ráfaga después del año 2011. Aquí, el conjunto se reduce a 147.

El segundo, es obtener los tópicos declarados como ráfaga después del cuarto mes del año 2011. Este filtro sólo deja fuera 1 tópico, reduciendo el conjunto a 146.

Tabla 6.10: Resultados MPC, con serie de máximos

Ítem	Resultado	% de validados
Cantidad de tópicos totales	146	100
Cantidad de validados por G.V	139	95,20
Cantidad de validados por G.H (G.H > 0)	119	81,50
Promedio de G.H	21,55	
Máxima G.H	323	

Posiblemente, los resultados pueden ser mejores, si se logra contar con un mayor volumen de data. No obstante, el método MPC entrega un mecanismo eficiente en tiempo y espacio, capaz de validar a T como un tópico emergente en el 95,20 % de los casos. Además, en el 81,05 % de los casos, es posible mapear el tópico en ráfaga a, por lo menos, una consulta relevante, siendo en promedio, más de 21 consultas las que se detectan como potenciales ráfagas asociadas a un tópico T.

6.1.6. Conclusiones CQA

Dado un contexto online, la detección de tópicos ráfagas, se puede realizar de forma eficiente en tiempo y espacio. Los experimentos realizados en este trabajo, confirman además, que la detección de este tipo de señales, se puede llevar a cabo de forma temprana, es decir, antes de que se alcance el *peak* de frecuencias.

Sobre si la detección de ráfagas orientada a tópicos presenta una mejora con respecto al uso de *querys* a la hora de hacer la detección, la respuesta es que en este contexto (CQA), la detección

orientada a *querys* no puede competir contra la detección orientada a tópicos, por una razón muy sencilla: Es muy poco probable que las *querys* (el *string*) se repitan, puesto que, como se menciona en este trabajo, en las comunidades de pregunta/respuesta predomina el lenguaje natural. Un experimento llevado a cabo sobre esto, confirma que de toda la data disponible (aproximadamente 6,000,000 de *querys*), la máxima frecuencia para la misma *query* es de 6.

La comparación entre tópicos vs *querys*, se aborda en el capitulo 6.2.

Este capítulo, además, confirma la hipótesis sobre la detección de tópicos y la posibilidad de, mediante estos tópicos, encontrar las potenciales consultas ráfagas. Este método *MPC*, es capaz de utilizar la estructura "índice invertido" para realizar esta tarea de forma eficiente. Además, se validan los estudios expuestos en [27], sobre el aglutinamiento de la actividad de clicks, en torno a un episodio ráfaga.

6.2. Logs de consultas

El segundo contexto a abordar en este trabajo, son los Logs de consultas.

Como se menciona en la sección 4.1.6, un Log de consultas es un archivo con consultas de usuarios realizadas sobre un motor de búsqueda (para más detalle, léase la sección 6.2.1).

En [25], se menciona que las búsquedas de los usuarios suelen ser concisas y "planas" (carecen de conjugaciones, por lo menos, más que en una comunidad CQA). Sin embargo, debido al volúmen de los datos a procesar, es necesario realizar la etapa de pre-procesamiento de éstos.

6.2.1. Características de la entrada

Los algoritmos generados para este contexto, trabajan sobre un cuerpo de datos con las siguientes características:

- Consultas de usuarios al motor de búsqueda de Yahoo!. Muestra durante 3 meses consecutivos del año 2011.
- Datos privados.
- Datos parcialmente estructurados.

- Volúmen: 36 GB, texto plano, separado en 3 archivo de 12 GB cada uno.
- Cantidad de consultas: 146,634,807.

El cuerpo de datos se compone de 2 partes; un *timestamp*, que corresponde al número de segundos transcurridos desde el 1 de enero de 1970, hasta el momento en que se creó dicho *timestamp* (*timestamp* UNIX) y la consulta en sí.

La etapa de pre-procesamiento de los datos es un tanto más compleja para este contexto que el contexto CQA. El detalle se muestra en la siguiente sección.

6.2.2. Monitorización de consultas

A lo largo de este trabajo, se da cuenta de que para detectar ráfagas, ya sea orientado a tópicos o consultas, se debe analizar la señal de éstos a través del tiempo. Al contar con un gran volúmen de datos y un alto número de consultas, generar series de tiempo con las frecuencias de aparición en los Logs de consultas para cada tópico o cada consulta se torna lento e impracticable según los recursos computacionales con los que se cuenta (principalmente, en términos de memoria). Esto pareciera más ser una estrategia *naive* a un problema que nisiquiera representa el núcleo del problema.

Debido a lo anterior, es que se utiliza una estrategia más sofisticada en la etapa de pre-procesamiento, basado en el trabajo propuesto en [24], estrategia también utilizada en [11].

En dichos trabajos, se emplea una estructura de monitorización, eficiente en términos espaciales y temporales, donde, por medio de una política LFU (emulando una memoria caché), se pueden obtener las k consultas más relevantes de una muestra de tamaño n de forma rápida. El trabajo propuesto, mejora el orden de las funciones de inserción, borrado y acceso a orden constante ($O(1)$), mediante la implementación de funciones de la política LRU y listas doblemente enlazadas, mientras que la extracción de las k consultas es en orden n ($O(n)$).

Una vez implementada la idea, la extracción de las k consultas más relevantes se realiza considerando intervalos de tiempo, es decir, cada un cierto intervalo de tiempo, se extraen las k consultas más relevantes de la caché emulada, se vacía la caché y se continúa con la monitorización. Este proceso se realiza hasta completar los 3 archivos de datos con los que se cuenta.

La salida de este algoritmo (al cual llamaremos "monitorización de primer nivel"), es un archivo con las $k * n$ consultas más relevantes de la muestra total (donde n corresponde a la cantidad de extracciones de la memoria caché emulada).

Esta salida no presta mayor utilidad si no es tratada de forma indicada, puesto que entrega las k consultas más relevantes de la muestra, pero lo que se busca encontrar son las consultas candidatas a ráfagas para generar las series de frecuencias asociadas a ellas. Entonces, dadas las n salidas de k consultas cada una, se busca encontrar las consultas permanentes para luego re-utilizar la estructura de monitorización (llamaremos a esta acción, "monitorización de segundo nivel") para nuevamente extraer las k consultas más relevantes **pero esta vez, sin considerar las consultas marcadas como permanentes anteriormente**. Las consultas consideradas permanentes serán aquellas con alta ocurrencia en las n salidas.

Como la finalidad principal del contexto Logs de consultas es comparar la detección orientada a tópicos versus la detección orientada a consultas, la monitorización se realizará también con esta orientación.

El algoritmo 3, muestra la estructura de monitorización en sus dos niveles.

Algoritmo 3 Monitorización de consultas

```
1:  Procedure MONITORIZACIÓN()
2:      instante ←ObtenerInstante( query )
3:      for each token ∈ query do
4:          /* Suma 1 a la frecuencia del término en el instante */
5:          insertar( token )
6:      endFor
7:      /* Cambio instante actual a instante posterior */
8:      if nuevo_instante then
9:          actualizar_instante( )
10:     endIf
```

La monitorización se realiza a nivel de un minuto, es decir, cada vez que transcurre un minuto en la muestra (medido en base al *timestamp*), se obtienen las top-k consultas más relevantes. Se considera $k = 100$.

Luego de realizar la monitorización, se obtiene una serie de consultas relevantes, las cuales se

deben procesar nuevamente. En primer lugar se descartan aquellas consultas relevantes en todas (o casi todas) las mediciones, puesto que por lógica, estas consultas son permanentes. Este filtro deja consultas como *youtube*, *facebook* o *google* fuera de la muestra de interés. Se podría decir que las consultas relevantes en la mayoría de las ocaciones, corresponden a *stopwords* para este análisis actual.

El segundo nivel de monitorización, se realiza de forma manual sobre la muestra de interés (consultas relevantes de toda la muestra - *stopwords*). Como se trabaja con consultas completas (no términos), se obtienen consultas candidatas a ráfagas de forma sencilla, ya que éstas sí tienen contexto y se sabe a lo que se refieren. Cabe recordar que las consultas en ráfaga suelen obedecer a consultas sobre acontecimientos de interés (nuevos productos, celebridades, etc.).

La tabla 6.11 muestra algunas consultas candidatas a ráfaga obtenidas de forma sencilla con las monitorizaciones realizadas.

Tabla 6.11: Consultas ráfagas

Id	Consultas candidatas
1	lindsay lohan
2	kirstie alley
3	natalie portman
4	wwe wrestlemania
5	world war z

Como se aprecia, se obtienen consultas sobre celebridades (1, 2 y 3), acontecimientos de interés popular (probablemente periódico) (4) y eventos de interés como el estreno de una película (5).

La tabla 6.12 resume la cantidad de consultas en ráfaga detectadas para cada mes de consultas.

6.2.3. Detección de ráfagas

Sobre el contexto actual, el principal objetivo es comparar la detección de tópicos en ráfaga contra la detección de consultas en ráfaga. Se busca validar una de las hipótesis de este trabajo, en específico validar si la detección de ráfagas orientado a tópicos puede hacer más eficiente (más

Tabla 6.12: Salida monitorización

Mes	Cantidad de consultas
1 (Abril 2011)	219
2 (Mayo 2011)	182
3 (Junio 2011)	228

temprana) la detección de ráfagas.

En el capítulo 6.1.5 se ha propuesto (y validado) un mecanismo capaz de encontrar las consultas en ráfaga asociadas a un tópico en específico declarado como ráfaga utilizando la actividad de *clicks*, es decir, sobre este contexto se puede realizar lo mismo, extrapolando una idea ya validada.

Volviendo a la idea principal de este contexto (comparar la eficiencia de la orientación a tópicos versus la orientación a consultas), una vez obtenidas las consultas candidatas a ráfagas luego de la monitorización, se obtienen los términos que componen esas consultas. Finalmente se crean las series de frecuencias de los términos (señales representantes). Vale remarcar que tantos las series de las consultas, como las de los términos se crean en unidades de frecuencia por minuto, puesto que se cree lo suficientemente preciso para detectar diferencias entre detecciones.

Tabla 6.13: Salida monitorización términos

Mes	Cantidad de términos
1 (Abril 2011)	480
2 (Mayo 2011)	402
3 (Junio 2011)	487

La tabla 6.13 muestra la cantidad de términos para los cuales se crearán series de frecuencias.

6.2.4. Resultados de detección

La detección de ráfagas se realiza mediante el algoritmo 1 propuesto en la sección 6.1.3. Los parámetros se mantienen fijos, puesto que entregan buenos resultados (las ráfagas se detectan de forma temprana).

El algoritmo de detección de ráfagas propuesto en este trabajo, es aplicable para detectar ráfagas en el contexto de Logs de consultas, sin embargo, el objetivo de este contexto es determinar si la detección orientada a términos es más eficiente que la detección orientada a consultas. Los siguientes puntos abordan los distintos análisis realizados.

Tabla 6.14: Tiempos de detección (promedio)

Mes	Instante de detección (min.)	
	términos	consultas
1 (Abril 2011)	13.524,11	18.730,03
2 (Mayo 2011)	15.041,80	17.254,34
3 (Junio 2011)	12.207,45	19.740,47

6.2.5. Conclusiones Logs de consultas

6.3. *Twitter*

Estado del arte

Aunque el estudio de las consultas en ráfaga en redes sociales (o más específico en *Twitter*) suele estar basado en extrapolaciones de estudios descritos en la revisión del estado del arte de este trabajo (sección 5), la detección de ráfagas en *Twitter* amerita un segundo análisis, debido en gran parte a la complejidad (o riqueza) de los datos. Se realiza una pequeña revisión sobre los trabajos realizados hasta el momento en el contexto de *Twitter* y consultas en ráfaga, con la finalidad de conocer los principales mecanismos empleados.

En [20] se propone un mecanismo de detección de tópicos ráfagas y análisis de tendencias. Los autores utilizan un algoritmo no descrito pero se explica que simula una cola de tópicos ráfagas. Los tópicos ráfagas son declarados en base a la frecuencia por minuto, es decir, se buscan cambios en las frecuencias de forma abrupta entre minutos cercanos. Luego utilizando una estrategia *greedy* se agrupan los tópicos ráfagas detectados en grupos denominados *trends* (tendencias). Una vez obtenidos las tendencias, se monitorean los *tuits* asociados a cada *trend* para obtener información más compleja. Este mecanismo podría en parte servir para lo que se busca en este trabajo (detectar ráfagas), pero no se especifíca sobre que campo del *tuit* se trabaja realmente ni cómo es en realidad el algoritmo. Sin embargo el enfoque de la monitorización es interesante debido al gran volúmen de

datos provenientes de *Twitter*.

En [32] se propone un mecanismo basado en dos pasos principales: primero una monitorización sobre el flujo de *tuis* cuya finalidad es extraer los *tuits* más relevantes en un periodo de tiempo dado y luego la segunda parte, donde se realiza un procesamiento sobre los tópicos y pares de tópicos (bigramas) para conocer los tópicos relevantes. Se utilizan técnicas de *hashing* para mejorar el desempeño. Se propone un método matemático basado en *aceleración* derivando la velocidad (de aparición) con respecto al tiempo para obtener el cambio en las apariciones del número de *tuits* por unidad de tiempo, las palabras por unidad de tiempo y los pares de palabras por unidad de tiempo. Esta medida es utilizada para declarar ráfagas.

Lo más relevante del trabajo es la idea de la monitorización del flujo de *tuits*, puesto que indica que se toma en consideración lo impracticable de mantener cada *tuit* en memoria.

En [7] se propone un mecanismo basado en el trabajo propuesto por Kleinberg en [16], además se comparan distintos modelos utilizando series de frecuencias las cuales representan distintas cosas dependiendo del modelo empleado, siempre basado en el mecanismo de *Kleinberg* y los dos estados (de un autómata finito) que plantea. El trabajo utiliza un número predeterminado de temas a analizar y está orientado a un contexto *offline*, pero de la misma forma en la cual se adapta el algoritmo propuesto en este trabajo de tesis para la detección de ráfagas se puede pasar una idea como la propuesta en este estudio. La preclasificación de los temas a analizar es otro aspecto que se puede manejar, mediante por ejemplo, la estructura de monitorización de consultas relevantes utilizada en la sección 6.2.2

Modelo de detección de tópicos en ráfagas

Twitter es una red social con distintos fines, que van desde el ocio, hasta fines informativos. En el trabajo presentado en [17], se concluye que más del 85 % de los *trending topics*[1] de una masa de *tuits* estudiados corresponden a titulares de noticias.

Twitter corresponde a un contexto distinto a los otros dos estudiados en este trabajo. Se conoce que el cuerpo de un *tuit* puede ir acompañado múltiples campos importantes del punto de vista del procesamiento de los datos de *Twitter* (llamaremos a estos campos de interés *entidades*), a diferencia de las comunidades de pregunta/respuesta y las consultas de usuarios, donde el tema

[1] Los *trending topics* corresponden a las tendencias o temas más repetidos en un instante en particular en *Twitter*

central es una parte del *string* completo, pero no se identifica de ninguna forma en especial.

Dentro de las entidades posibles, un *tuit* puede componerse de uno (o múltiples) *hashtag*[2] lo que generalmente identifica el tema sobre el cual trata el *tuit*

Además del *hashtag*, existe una entidad llamada *retweet count* (o contador de *retuiteo*). Esta entidad se utiliza para conocer la cantidad de veces que un *tuit* ha sido *retuiteado* o desplegado por otro usuario, distinto a quién lo generó en un principio. Se puede decir que un mismo *tuit* se expande través de la red mediante los *retuiteos* de los usuarios y más importante aún, un tema de interés se disemina por la red (local o mundialmente).

Desde la perspectiva de grafos, cada usuario corresponde a un nodo del grafo, mientras que las aristas corresponden a las relaciones entre estos usuarios (en *Twitter* el *follow*[3] entre usuarios). Entonces un *tuit* se puede mover por las aristas cuando un nodo *retuitea* un *tuit* creado (o *retuiteado*) por un nodo vecino. Cada uno de los movimientos del *tuit* por las aristas del grafo aumentan el contador de *retuiteo* asociado al *tuit*.

[2] *Hashtag* es una etiqueta que resalta una parte de un *tuit*, generalmente referido al tema central sobre el cual se habla. Se caracteríza por comenzar con el símbolo #, p.e. #FifaWorldCup

[3] La relación *follow* entre usuarios es la manera de que en twitter un usuario "siga" a otro, lo que implica que los *tuits* del usuario "seguido" son visibles para el usuario "seguidor". A puede seguir a B, como B puede seguir a A

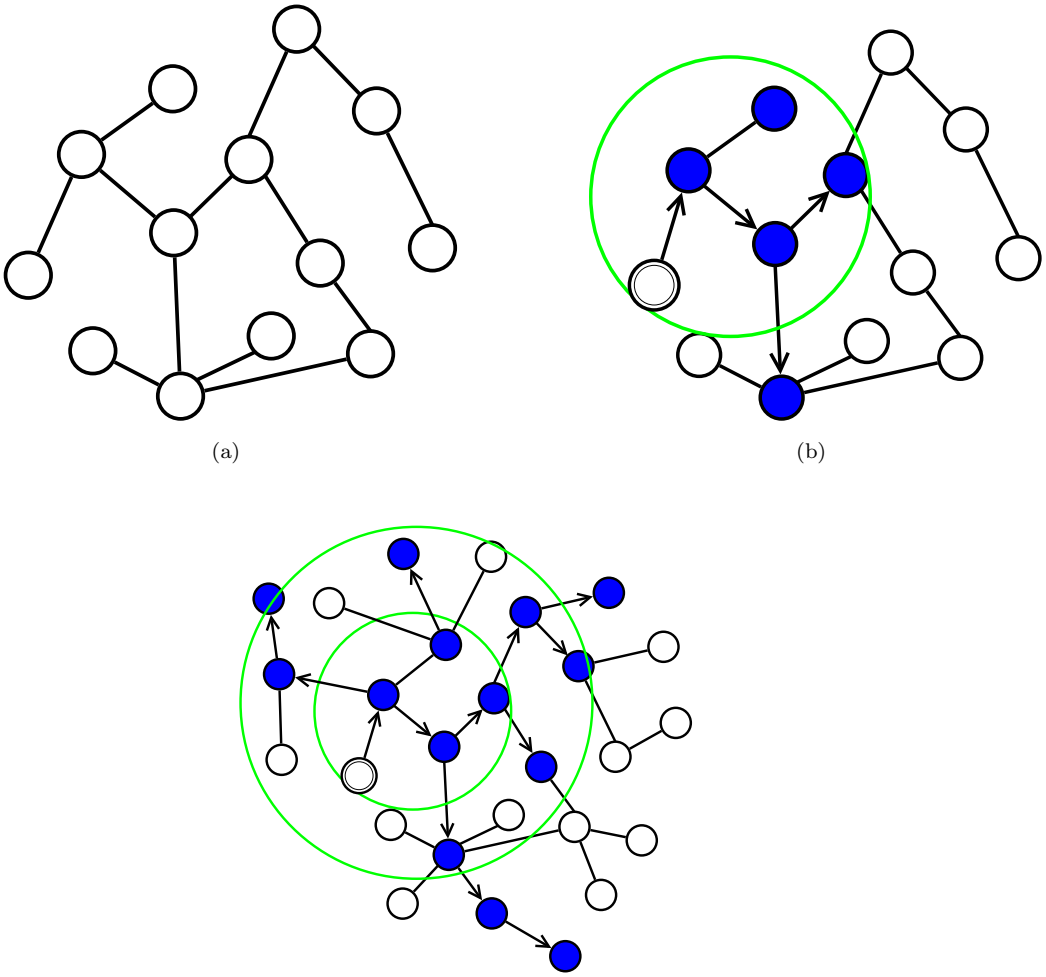

Figura 6.10: Movimiento de *tuit* por red de usuarios mediante *retuiteos*. Fuente: elaboración propia

La figura 6.10 representa una forma de diseminación de un *tuit* por la red. El nodo con doble círculo en las partes (b) y (c) representa el origen del *tuit*. Las flechas representan el movimiento del *tuit* por la red. Los nodos rellenos representan usuarios que han *retuiteado* el *tuit*. Se aprecia que un *tuit* se disemina por la red cubriendo zonas del grafo. Esto no necesariamente es así, puesto que se pueden dar comportamientos donde un *tuit* tiene alto impacto en una zona del grafo, pero la repercusión más grande se produce en el otro extremo del grafo, por ejemplo, un *tuit* generado en Europa con alta repercusión en *Chile*.

En las redes sociales existen usuarios con alta popularidad (en *Twitter* altamente seguidos por otros usuarios) los cuales mantienen su popularidad de distintas formas. En el ámbito informativo, las cuentas asociadas a noticieros o revistas de espectáculo desean ser los primeros en publicar

acerca de hechos de interés. Es en este punto donde una detección temprana de tópicos relevantes se vuelve importante. Si se puede detectar que un tema, un *tuit* o un *hashtag* se tornará *trending topic*, entonces se puede publicar sobre este tema antes de que alcance dicho estado.

6.3.1. Características de la entrada

Los algoritmos generados para este contexto, trabajan sobre un cuerpo de datos con las siguientes características:

- *tuits* en formato JSON, descargados mediante API *Twitter4j* de forma continua desde XXXX hasta XXXX.
- Datos públicos
- Tuplas estructuradas pero no idénticas (admiten campos nulos).
- Volúmen: XXXX GB, texto en formato JSON.
- Cantidad de *tuis* $XXXX$.

La preparación de los datos de entrada es un punto relevante, el cual se revisará en la sección 6.3.2

6.3.2. *Hadoop*

Debido a la masividad de los datos obtenidos en este contexto de estudio, surge la necesidad de emplear mecanismos sofisticados para el procesamiento de éstos. En primer lugar, la compresión de los datos fue necesaria por no contar con arquitectura para no hacerlo. La estrategia empleada fue descargar un cierto volumen de datos y direccionarlos a un paquete comprimido. Con esto, se redujeron los paquetes de *tuits* a paquetes comprimidos de un 10 % del peso original aproximadamente utilizando la compresión *bzip2*.

Esto trajo consigo dos problemas principalmente: (i) La necesidad de descomprimir los paquetes para redireccionarlos a un programa para procesar los datos, lo que produce un retardo en el desempeño (y considerando la masividad de los datos, esto no es irrelevante) y (ii) Los datos distribuidos en paquetes independientes, lo que implicaría tener que procesar cada paquete como una instancia aislada.

Debido a las razones antes expuestas, se decide realizar el procesamiento previo de los datos bajo el *framework Hadoop*. *Hadoop* es un framework (y software ibre) que permite procesamiento de grandes volúmenes de datos de forma distribuida (o pseudodistribuida, explotando el multi-procesamiento). *Hadoop* trabaja sobre la lógica *MapReduce*, la cual realiza dos tareas principales; asociatividad mediante claves (*Map*) y consolidación de los datos de salida (*Reduce*). Hadoop entre otras cosas permite:

- Procesar múltiples archivos distribuidos lógica o físicamente.
- Procesar archivos comprimidos bajo distintos algoritmos (incluído *bzip2*).
- Procesamiento tolerante a fallos y auto-arrancable en caso de errores.

Hadoop es utilizado para obtener las series de frecuencias de los tópicos que se requieran a través de todo el tiempo de monitoreo.

No obstante, existen dos perspectivas a la hora de proponer un mecanismo completo para la detección de ráfagas. Una parte es realizar la generación de las series de tiempo para detectar ráfagas utilizando *Hadoop* trabajando sobre datos previamente descargados. La segunda derivada es la detección de ráfagas en tiempo real, donde se debe emplear un mecanismo eficiente para lidiar con el alto volumen de datos. Para llevar a cabo esto, se propone la estructura utilizada en la monitorización de consultas en la sección 6.2.2 debido a las características que ahí se exponen.

Como ya se sabe que la estructura de monitorización funciona y es eficiente, lo que se debe probar es que la estrategia para generar las series de tiempo es la adecuada. En la sección 6.3 se menciona que un *tuit* se compone de múltiples *entidades*. En este trabajo se propone el uso de la entidad *contador de retuit (retweet count)* asociada a un *tuit* para detectar ráfagas, debido a que un tema de interés general se mueve entre los nodos de la red de *Twitter* mediante el *retuiteo* de los usuarios (figura 6.10). En específico se analiza cómo temas de interés conocidos entre los periodos de datos que se tienen (*trending topics*) se tornan ráfagas y si es posible detectarlos antes de su *peak*.

Entonces, dado un conjunto de *trending topics* conocidos C_{tt} , entre XXXX y XXXX:

Obtener un conjunto de *tuits* t que traten sobre algún elemento de C_{tt}.
$\forall\ t_i \in C_{tt}$.
Generar series de frecuencias del contador de *retuit* asociado a t_i.
Procesar las series de frecuencias, utilizando el algoritmo propuesto.

Capítulo 7

Conclusiones

Bibliografía

[1] Charu C Aggarwal and Philip S Yu. Outlier detection for high dimensional data. In *ACM Sigmod Record*, volume 30, pages 37–46. ACM, 2001.

[2] L.E.P. Ayala. *Probabilidad y estadística*. Area : Matemáticas. Pearson Educación, 2004.

[3] Luiz André Barroso, Jeffrey Dean, and Urs Hölzle. Web search for a planet: The google cluster architecture. *Micro, Ieee*, 23(2):22–28, 2003.

[4] Berkant Barla Cambazoglu, Flavio P Junqueira, Vassilis Plachouras, Scott Banachowski, Baoqiu Cui, Swee Lim, and Bill Bridge. A refreshing perspective of search engine caching. In *Proceedings of the 19th international conference on World wide web*, pages 181–190. ACM, 2010.

[5] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[6] Doug Cutting and Jan Pedersen. Optimization for dynamic inverted index maintenance. In *Proceedings of the 13th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 405–411. ACM, 1989.

[7] Qiming Diao, Jing Jiang, Feida Zhu, and Ee-Peng Lim. Finding bursty topics from microblogs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 536–544. Association for Computational Linguistics, 2012.

[8] Nicole B Ellison et al. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230, 2007.

[9] fayerwayer.com. Facebook messenger para windows. urlhttps://www.fayerwayer.com/2011/12/ya-puedes-descargar-oficialmente-facebook-messenger-para-windows/, 2011.

[10] genbeta.com. Facebook timeline. urlhttp://www.genbeta.com/redes-sociales-y-comunidades/timeline-el-rediseno-completo-del-perfil-de-facebook-para-mostrar-la-historia-de-tu-vida-de-manera-automatica, 2011.

[11] Carlos Gómez-Pantoja. *Servicios de cache distribuídos para motores de búsqueda web.* PhD thesis, Universidad de Chile, Abril 2014.

[12] Qi He, Kuiyu Chang, Ee-Peng Lim, and Jun Zhang. Bursty feature representation for clustering text streams. In *SDM*, pages 491–496. SIAM, 2007.

[13] Yunliang Jiang, Cindy Xide Lin, and Qiaozhu Mei. Context comparison of bursty events in web search and online media. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1077–1087. Association for Computational Linguistics, 2010.

[14] Pawel Jurczyk and Eugene Agichtein. Discovering authorities in question answer communities by using link analysis. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 919–922. ACM, 2007.

[15] Marcel Karnstedt, Daniel Klan, Christian Pölitz, Kai-Uwe Sattler, and Conny Franke. Adaptive burst detection in a stream engine. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1511–1515. ACM, 2009.

[16] Jon Kleinberg. Bursty and hierarchical structure in streams. *Data Min. Knowl. Discov.*, 7(4):373–397, October 2003.

[17] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600. ACM, 2010.

[18] Theodoros Lappas, Marcos R Vieira, Dimitrios Gunopulos, and Vassilis J Tsotras. On the spatiotemporal burstiness of terms. *Proceedings of the VLDB Endowment*, 5(9):836–847, 2012.

[19] Ronny Lempel and Shlomo Moran. Predictive caching and prefetching of query results in search engines. In *Proceedings of the 12th international conference on World Wide Web*, pages 19–28. ACM, 2003.

[20] Michael Mathioudakis and Nick Koudas. Twittermonitor: trend detection over the twitter stream. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 1155–1158. ACM, 2010.

[21] Royal Pingdom. Internet 2012 in numbers. urlhttp://royal.pingdom.com/2013/01/16/internet-2012-in-numbers/, 2012.

[22] Vijay Raghavan, Peter Bollmann, and Gwang S Jung. A critical investigation of recall and precision as measures of retrieval system performance. *ACM Transactions on Information Systems (TOIS)*, 7(3):205–229, 1989.

[23] Paricia Correia Saraiva, Edleno Silva de Moura, Novio Ziviani, Wagner Meira, Rodrigo Fonseca, and Berthier Riberio-Neto. Rank-preserving two-level caching for scalable search engines. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 51–58. ACM, 2001.

[24] Ketan Shah, Anirban Mitra, and Dhruv Matani. An o (1) algorithm for implementing the lfu cache eviction scheme. Technical report, Technical report, 2010."http://dhruvbird. com/lfu. pdf, 2010.

[25] Craig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz. Analysis of a very large web search engine query log. In *ACm SIGIR Forum*, volume 33, pages 6–12. ACM, 1999.

[26] Statista. Social network users. urlhttp://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/, 2015.

[27] Ilija Subašić and Carlos Castillo. The effects of query bursts on web search. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, volume 1, pages 374–381. IEEE, 2010.

[28] Ilija Subašic and Carlos Castillob. Investigating query bursts in a web search engine. 2013.

[29] Gabriel Hernán Tolosa and Esteban Feuerstein. Mejoras algorítmicas y estructuras de datos para búsquedas altamente eficientes. In *XIV Workshop de Investigadores en Ciencias de la Computación*, 2012.

[30] Michail Vlachos, Christopher Meek, Zografoula Vagena, and Dimitrios Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 131–142. ACM, 2004.

[31] whatsis.com. What is twitter. urlhttp://whatis.techtarget.com/definition/Twitter, 2010.

[32] Wei Xie, Feida Zhu, Jing Jiang, Ee-Peng Lim, and Ke Wang. Topicsketch: Real-time bursty topic detection from twitter. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 837–846. IEEE, 2013.

[33] Yinglian Xie and David 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. IEEE, 2002.

[34] Junjie Yao, Bin Cui, Yuxin Huang, and Xin Jin. Temporal and social context based burst detection from folksonomies. In *AAAI*, pages 21–27, 2010.

[35] Hugo Zaragoza, B Barla Cambazoglu, and Ricardo Baeza-Yates. Web search solved?: all result rankings the same? In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 529–538. ACM, 2010.

[36] Yunyue Zhu and Dennis Shasha. Efficient elastic burst detection in data streams. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 336–345. ACM, 2003.

[37] George Kingsley Zipf. Human behavior and the principle of least effort. 1949.

[38] George Kingsley Zipf. *The psycho-biology of language: An introduction to dynamic philology.* Routledge, 2013.