PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

Implementación y Evaluación de Algoritmo De Indexación de Tweets

FERNANDO XAVIER FREZ MERIÑO

Profesor Guía: Wenceslao Palma Muñoz

Carrera: Ingeniería de Ejecución en Informática

INFORME FINAL DEL PROYECTO PARA OPTAR AL TÍTULO PROFESIONAL DE INGENIERO EJECUCIÓN EN INFORMÁTICA

Septiembre 2018

Esta Memoria de Título está dedicada a mi Madre, Tía y Tío, mi Familia.
En especial a mi Madre que sin ella no pudiera haber logrado ser la persona que soy y seré.
Además, le dedico este logro a todas y cada una de las personas que necesitaron y pude estar ahí para ayudarlas.
Abnegación y Constancia.
[1]

Índice

Lista de Figuras

Resumen

Desde el punto de vista de una búsqueda en tiempo real sobre microblogs (twitter) nuevos datos deben quedar disponibles para su búsqueda inmediatamente después de su creación. En bases de datos tradicionales el requerimiento antes mencionado puede ser fácilmente satisfecho creando un índice, el cual es evaluado midiendo el intervalo de tiempo que transcurre entre la inserción del nuevo dato y su disponibilidad en el índice. Sin embargo, un enfoque tradicional va degradando su rendimiento cuando se utiliza para indexar microblogs (twitter), ya que la carga de trabajo (actualización del índice y consulta) aumenta.

En el presente trabajo se implementa y evalúa TI (Tweet Index), un algoritmo para la indexación y búsqueda de tweets en tiempo real.

La intuición sobre la cual trabaja TI se basa en que se deben indexar los tweets que tienen una alta probabilidad de ser consultados mientras que los otros tweets pueden ser indexados posteriormente. Los resultados experimentales obtenidos sobre una intuición sobre la cual se basa TI.

Abstract

From the perspective of a searching on a real-time about microblogs, new data must be available for search immediately after its creation. In the traditional databases, the requirement mentioned, can be easily solved by creating an index, which is evaluated by measuring the time lapse between the input of a new data and it's availability in the index set up. However, a traditional approach degrades its performance when it is used for micrologs index (twitter), because the workload is increased due to the updating process of indexes and queries.

The present paper evaluates and implements TI (Tweet Index), an algorithm for the process of indexing and searching tweets, on real-time basis. Intuition over which TI works, is based on the need to indexing the tweets that have a high probability to be consulted, meanwhile others tweets can be indexed later. The experimental results is based on intuition TI.

Keywords: Real-Time Search, Index, Rancking

1 Introducción

Hoy en día, las redes sociales debido a su objetivo de diseño manejan una gran cantidad de información diaria. Twitter es un microblog que desde su lanzamiento en la red se estima que tiene alrededor de 200 millones de usuarios, generando 65 millones de tweets al día y con más de 800.000 peticiones de búsqueda diarias. Al momento de generar una búsqueda de un tema en especial o palabra clave, los motores de búsqueda sólo tienen la capacidad de mostrar el resultado dado el orden en que se encuentran ordenados los tweets, o sea, su antigüedad desde que fue ingresado al sistema, lo cual implica que se recuperará información de poco uso y a veces sin sentido. En mitigación de la problemática anterior, existen algoritmos que mediante la técnica de indexación (implementación de un índice invertido), y una elaborada función de ranking, responden en forma eficiente y eficaz a una consulta basada sobre una o múltiples palabras claves.

El método que proponen los algoritmos, tiene su base en la clasificación de los tweets mediante su contenido (análisis de texto) que deberá coincidir con la palabra clave de búsqueda. Dentro del conjunto de los tweets seleccionados no necesariamente todos aportarán utilidad, en virtud a lo anterior se creará un ranking, donde los primeros K elementos de ese ranking serán en términos estadísticos los más útiles y por ende los indexados en primera instancia.

Considerando la gran cantidad de datos a manejar, es importante registrar en estructuras de datos la información de cada tweet, su relación y los usuarios propiamente tal.

Debido a que el esquema de indexación es independiente del ranking, se pueden definir diferentes tipos de funciones de ranking, existen adaptaciones para los sistemas de redes sociales, donde se explotan las características del comportamiento de los usuarios, sus relaciones de amistad, logrando estadísticas que se almacenan en memoria para apoyar el sistema de elección y filtro de tweets.

En el presente trabajo se implementa y evalúa TI (Tweet Index), un algoritmo para la indexación y búsqueda de tweets en tiempo real.

La intuición sobre la cual trabaja TI se basa en que se deben indexar los tweets que tienen una alta probabilidad de ser consultados mientras que los otros tweets pueden ser indexados posteriormente.

2 Definición del Problema

Twitter, uno de los microblog con mayor demanda del presente tiempo cuenta dentro de sus funcionalidades, con un modelo de búsqueda en tiempo real que no cumple con las expectativas deseadas. Al momento de analizar un retorno a una consulta de una palabra clave, éste no siempre será de utilidad debido principalmente a la forma de administrar los tweets día a día.

Las webs en tiempo real, como muchas de las nuevas tendencias no tienen una definición exacta, pero si cuentan con las siguientes características: Es una nueva forma de comunicación, crea un nuevo estilo de contenido, es en tiempo real (inmediatez), es pública y tiene una gráfica social explícita asociada a él. Por otro lado, las búsquedas en tiempo real, funcionalidades otorgadas por Twitter para encontrar un tweet específico o relacionado a un tema en especial se vuelve una tarea costosa, lo anterior es causado por la gran cantidad de información que se sube por día a la red, siendo un aproximado de 50 millones diarios según las estadísticas de Twitter. Considerando las cifras anteriores, se deben procesar miles de datos por segundo, además de generar índices en conjunto a una base de palabras claves útiles, con la finalidad de obtener un retorno aceptable en tiempo real.

Otro problema que mantiene la búsqueda en tiempo real, es la baja efectividad de la función de ranking, función que en el actual modelo de búsqueda de Twitter ordena los tweets basados en su tiempo de ingreso al sistema, por lo tanto, el tweet más reciente presentará un mayor ranking sin considerar si la información que contiene es útil o no. De la misma forma, las actualizaciones de dichos rankings representan grandes esfuerzos de ejecución debido a la alta tasa de ingreso de nuevos tweets, por lo cual es necesario estudiar algoritmos que den solución a dicho problema.

En virtud a la problemática anterior, los algoritmos a estudiar permiten generar consultas con resultados bastante efectivos y eficientes, para ello están compuestos de distintos elementos indispensables para dar soporte al manejo de la gran cantidad de información y su clasificación, logrando así resultados en tiempo real de mayor utilidad, reduciendo los costos de procesamiento y almacenamiento los cuales son evaluados bajo métricas de interés.

3 Objetivos

En el presente título se definen los objetivos asociados al presente trabajo:

3.1 Objetivo General

- Implementar y evaluar algoritmo de indexación de Tweets
- Analizar el rendimiento del algoritmo

3.2 Objetivos Específico

- Comprender algoritmos de indexación de tweets
- Estudiar la búsqueda en tiempo real en el contexto de indexación de tweets
- Estudiar la función de Ranking en combinación de la relación entre usuario y Twitter
- Evaluar: a). - Costo de indexación b). - Ejecución del proceso de consulta.

4 Estado del Arte

En la actualidad, específicamente el presente año el problema de la búsqueda en tiempo real se ha seguido desarrollando, bajo la perspectiva de que un problema tiene innumerables soluciones. Considerando el fundamento anterior a continuación, se describirán tipos de algoritmos que dan solución a la problemática planteada.

4.1 TI: Un mecanismo eficiente de indexación en búsqueda en tiempo real sobre tweets

En el presente algoritmo, la colección de tweets debe ser de un tamaño suficiente como para poder generar estadísticas y así posteriormente evaluarlas. Dichos tweets serán ingresados al sistema de dos formas o esquemas de indexación: en tiempo real y el Batch en segundo plano, de acuerdo al modelo planteado en el paper fuente [1]. Para facilitar la rapidez de la creación de índices, búsqueda y otras funciones propias, el algoritmo guarda en memoria cierta información del tweet tales como: la hora en que fue ingresado al sistema, IDs e información relacionada al usuario, entre otros datos. A continuación, se puede apreciar en la Figura 1, la forma global de la arquitectura correspondiente al TI, siendo explicada a continuación.

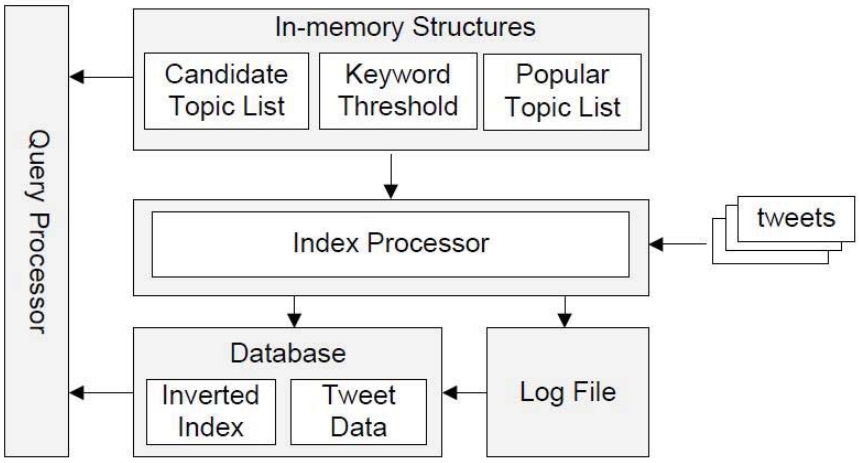

Figura 1 Arquitectura TI [1]

La Función ranking, es un elemento que otorga ventaja ante otros sistemas de clasificación, siendo ésta la encargada de generar listas de tweet que presenten una gran utilidad respecto a la consulta. Lo anterior, es provisto con las estructuras en memoria en la figura 2 referente al interior del Index Processor. Incluyendo a lo anterior, la popularidad de cada usuario

será un factor determinante mediante el campo User Page Rank, que además es utilizado por Google [4]. Dicho campo debió ser construido ya que no había posibilidad de rastrear el link entre los usuarios, por lo tanto, se generó un valor aleatorio para cada tweet.

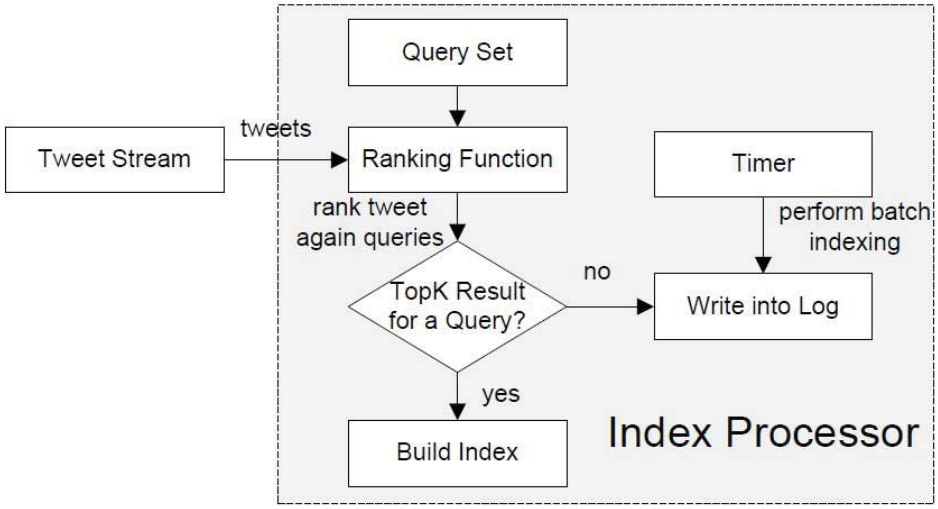

Figura 2 Proceso de Indexación [1]

Mientras son procesados los tweets, hay un conjunto de ellos que se mantiene en el Log File, conjunto de tweets que se consideran Noisy (de poco uso), para los cuales al momento en que se generen los resultados, serán agregados a la matriz de tweets para ser utilizados. Los tweets que se indexan en primera instancia (tiempo real) se clasifican como Distinguished.

El proceso índice, genera un índice invertido similar en espíritu al trabajo realizado en el índice parcial generalizado [5], y así luego de haber sido constituido el retorno de la consulta será una parte de dicho índice.

Toda consulta entrante se considera como una consulta no popular, por lo que no es insertada de inmediato al conjunto de consultas más frecuentes, sólo cuando se efectúa una actualización del incide por el proceso batch, el conjunto de consultas es actualizado, generando así, un resultado de mayor utilidad en relación a la consulta realizada.

Cabe destacar que dada la naturaleza de los tweets (elecciones presidenciales 2013) las consultas deberán ser relacionadas a ese contexto, lo cual afecta el abanico de consultas no así las discusiones que se generen dentro del DataSource.

5 Implementación Tweet Index (TI)

5.1 Stream de Tweet

Para comenzar cualquier incursión respecto a la implementación de TI, era requerimiento obtener una cantidad de datos (tweets) relevante para poder generar la implementación del algoritmo. La base de datos de tweets se encuentra contextualizada en las elecciones presidenciales del año 2013, donde el conjunto de datos cuenta con un total de 2.951.719 tweets como se ejemplifica en la Figura 3.

El conjunto de datos se encuentra materializado en un archivo de texto plano con un peso equivalente a 711,8 Megabytes, el cual cuenta con la siguiente estructuración:

- **id_articulo**: ID interno de cada tweet.
- **cuerpo**: Mensaje del tweet
- **tipo_sentido**: Positivo, negativo, neutro o sin clasificar
- **nombre**: Nombre del usuario
- **fecha_publicación**: Fecha en que se generó el tweet (TimeStamp)
- **url_artículo**: Dirección del tweet en internet
- **id_perfil**: ID de cada tema

De los registros descritos, sólo requiere la utilización de *id_artículo*, *cuerpo*, *nombre* y *fecha_publicación*. Luego del filtrado con una expresión regular codificada en lenguaje Perl, se logra la reducción de los campos llegando aproximadamente a los 430 MB.

5.2 Modelamiento de la implementación

5.2.1 Modelo General

En primer lugar, la Figura 3 muestra el modelo general del algoritmo en relación a la interfaz, DataSource, LogFile y los resultados alojados en el Archivo Resultado.

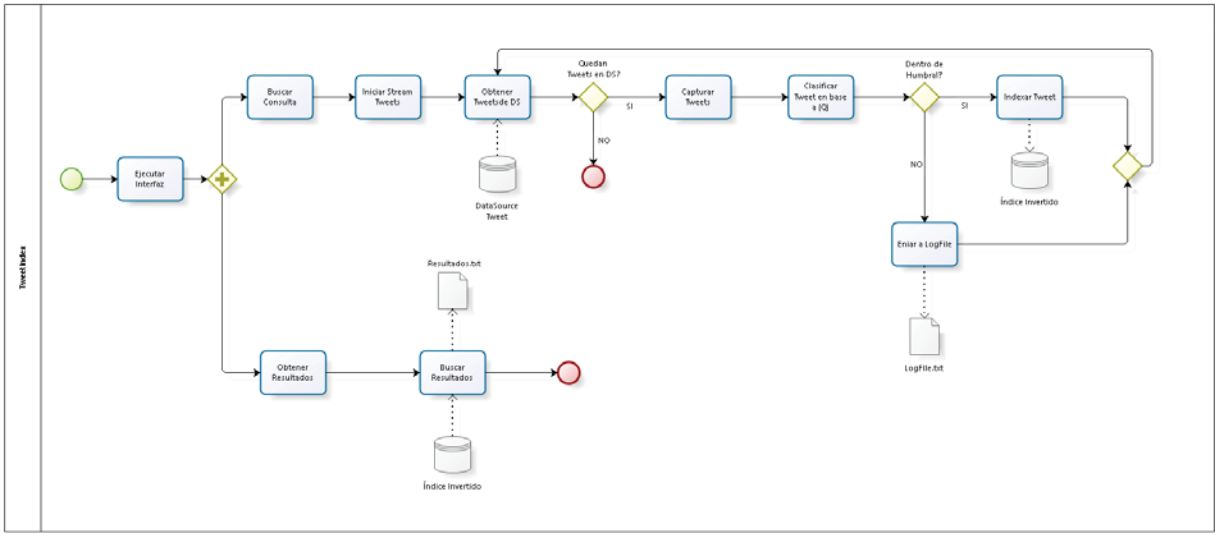

Figura 3 Modelo General

5.2.2 Modelamiento del Algoritmo

La figura 4 expone las clases principales y su relación entre ellas. A continuación, se explica su objetivo de diseño:

- **Clase Interface:** El objetivo de esta es capturar la consulta realizada por el usuario. Además de ello, da inicio al proceso interno e instancia la clase de índice invertido que a su vez generará los resultados. Todo lo anterior en relación al accionar de cada botón en ella.
- **Clase Generador:** Esta clase es esencial para el algoritmo, ya que obtiene del DataSource los tweets que luego son enviados al objeto Capturador. El objeto generador simula una conexión al Stream de Twitter.
- **Clase Capturador:** Su misión es ingresar el nuevo Tweet en la matriz de tweet, calcula variables de fechas relacionadas al tweet y luego lo envía a la Función Ranking.

- **Clase Función Ranking:** Es la que determina, en base a ciertas métricas, el valor (ranking) de un tweet en relación a cierta consulta, mientras más antiguo el tweet menor valor tendrá, por lo que se privilegia el indexar tweets mas actuales.
- **Clase Índice Invertido:** Es el encargado de generar el índice y luego imprimirlo en el archivo Resultado.
- **Clase Tweet:** Mantiene todos los atributos y métodos para poder administrarlo en base a los requerimientos del algoritmo.
- **Clase Query:** La instancia de ésta clase genera un objeto consulta, el cual contiene la consulta (el cuerpo) y su fecha de ingreso al sistema.
- **Clase User:** Información relevante respecto al usuario creador del tweet, tales como el identificador del usuario (entre otros).
- **Clase Tupla:** Al ser instanciada, crea un objeto tupla el cual es el elemento atómico en el índice invertido y conformará la lista resultado.
- **Archivo Fuente DataSource:** Archivo de texto que contiene los tweets filtrados, contiene los campos Id_Tweet, Cuerpo_tweet, user_tweet y timeStamp. Los campos están separados por el delimitador "--".
- **Archivo LogFile:** Este archivo de texto cumple la función de almacenar todos los tweets que no son indexados con el objetivo de ser un almacenamiento temporal de tweets con poco valor en contraste a las consultas ingresadas hasta el momento.
- **Archivo Resultado:** En este archivo los tweets aparecerán agrupados por cada palabra de la consulta. Sólo se imprimirá el cuerpo del tweet, por lo tanto, cada línea corresponderá a uno en particular.
- **Clase Clasificador de Tweet:** Encargada de cuantificar la utilidad de un tweet en relación a cada consulta almacenada en el conjunto de consultas. Si el valor final de utilidad del tweet está dentro de un rango o umbral, este será indexado.

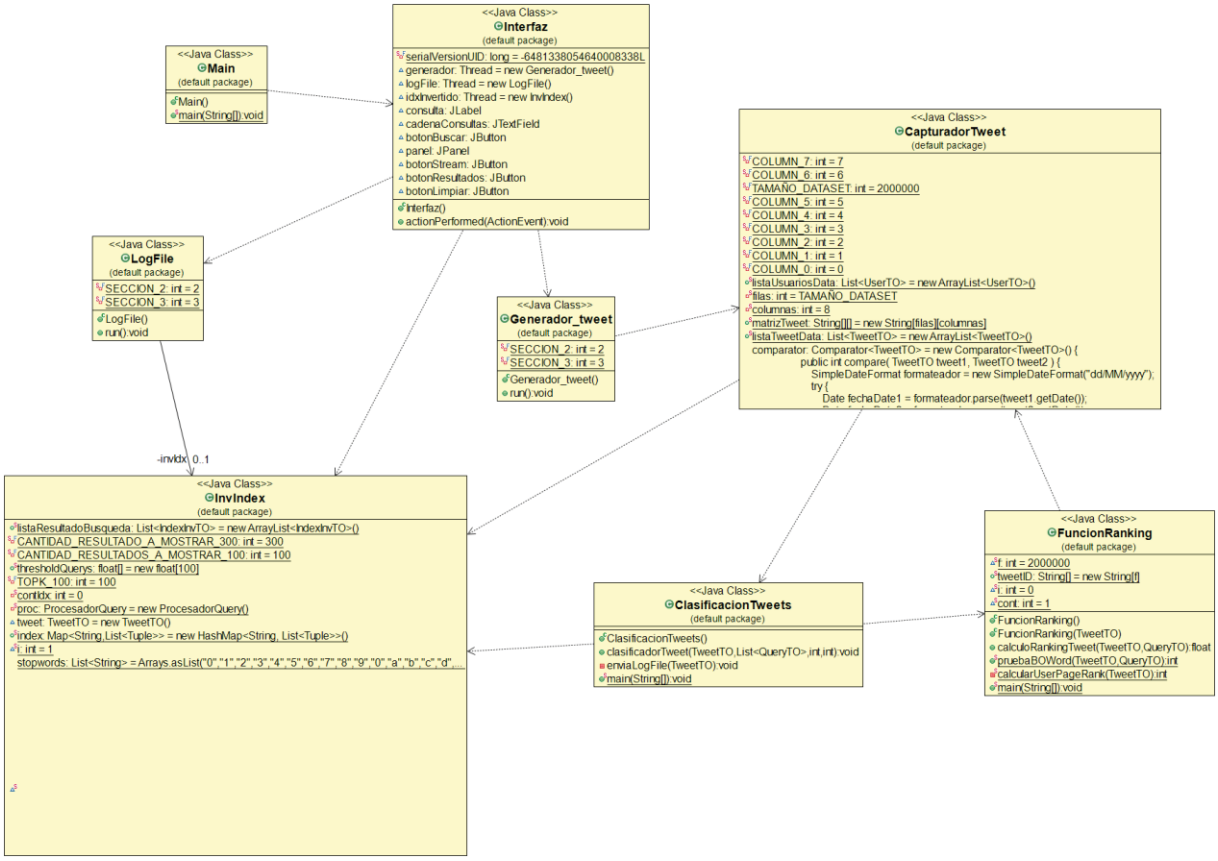

Figura 4 Diagrama de Clases

5.2.3 Clasificación de Tweet

Dentro del proceso central de evaluación de tweet, la clasificación es el proceso donde se cuantifica, por ejemplo, mediante la función Ranking la utilidad del tweet en relación al conjunto de consultas procesadas por el sistema. La estimación de la utilidad contempla principalmente, el recorrido de la matriz que relaciona las consultas y las palabras claves registradas en el sistema. Finalmente, si el valor obtenido luego de la clasificación está dentro del umbral estimado, el tweet se indexa de lo contrario se envía al Logfile.

5.2.3 Función Ranking

En lo que respecta a la esencia del algoritmo, la Función Ranking es el componente más importante de todos. La relevancia radica en la capacidad de poder cuantificar el valor de utilidad de un Tweet contra una consulta, o en otras palabras un Ranking (figura 7). En base a dicho valor y la comparación a un umbral (valor numérico definido a conveniencia) se determinará su inmediata indexación o, su envío al archivo LogFile para ser devuelto posteriormente al flujo principal, una vez que se cumpla la condición de la obtención del resultado.

Interiormente, se efectúan cálculos en relación a la fecha y hora (TimeStamp) de la consulta y el tweet, se recopila información respecto al usuario, calcula la similaridad consulta/tweet y finalmente, se obtiene el valor del Ranking.

La similaridad Tweet/Consulta, es un cálculo realizado en base a la implementación del algoritmo Bag of Word el cual se describe de la siguiente forma:

- Se toma las cadenas de consulta y cuerpo del tweet por separado y se cuentan las palabras distintas
- Se conforma un diccionario con dichas palabras
- Se generan dos vectores, uno para la consulta y otro el tweet del largo del diccionario.
- En base al índice del diccionario, cada posición del vector corresponderá a un contador de palabras.

Con los dos vectores resultantes y la fórmula siguiente (Figura 5), se realiza el cálculo de la similaridad:

$$sim(q,t) = \frac{q \times t}{|q||t|}$$

Figura 5 Ecuación Similaridad Tweet y Consulta

Luego, para calcular el valor del Ranking se utilizarán las siguientes variables de la forma definida en la Figura 6:

- **User Page-Rank:** Valor que tiende a calificar a un usuario más popular que otro.
- **Time Stamp:** Tiempo en que ingresó al sistema el tweet.
- **Similarity:** Similitud entre el tweet y la consulta.

De los cuales, los dos último son datos usados para comparación.

$$\mathcal{F}(q,t) = \frac{w_1 \times t.UPageRank + w_2 \times sim(q,t)}{q.timestamp - t.timestamp} + \frac{w_3 \times tree.popularity}{q.timestamp - tree.timestamp}$$

Figura 6 Ecuación Función Ranking

De la fórmula anterior, *q* corresponde a la consulta y *t* al tweet. La ecuación es afectada por el tiempo, por lo cual es más relevante un tweet más nuevo respecto de uno de mayor antigüedad.

5.2.4 Índice Invertido

Esta sección se desarrollará en dos ámbitos, primero el qué es un índice invertido y luego, cómo fue implementado en el presente algoritmo.

El propósito del índice invertido es generar una rápida y completa búsqueda de texto, el cual dependerá de la cantidad de datos nuevos añadidos. Está formado por dos elementos, el vocabulario (tweet) y las listas de ocurrencias (para cada término, la lista de documentos donde este aparece) como aparece en la Figura 7. En esencia, se propone la idea de que este modelo es semejante al algoritmo Bag of Word.

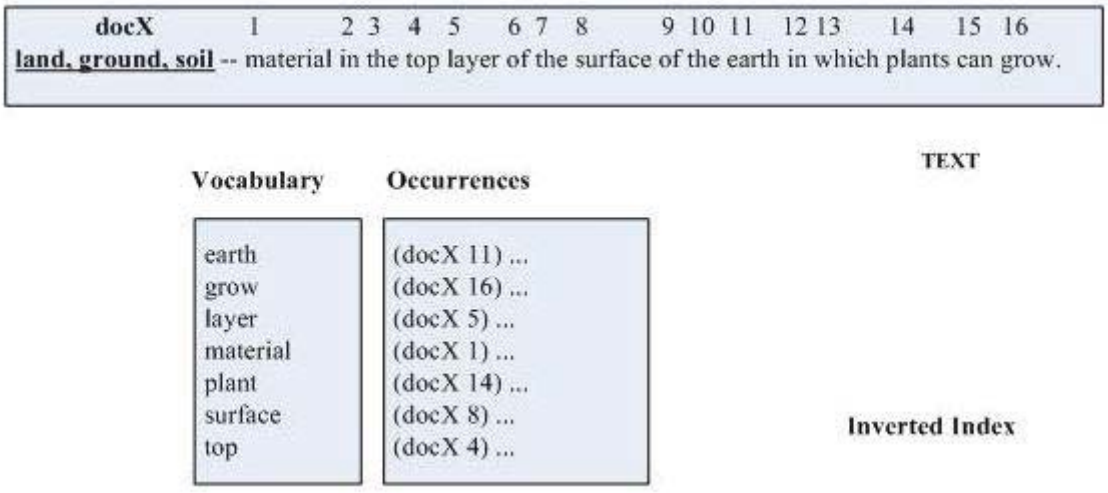

Figura 7 Ejemplo Índice Invertido

En segundo lugar, el vocabulario corresponde al cuerpo de la consulta y el documento a los Tweets. Por cada tweet que es relevante, se genera una instancia de la clase de índice invertido. Del Tweet, lo relevante es su cuerpo, por lo tanto, sólo ese campo se indexa y se añade

a una lista. Al momento de buscar en el índice, se llama al método “search” donde se le pasa como parámetro una lista, la cual contendrá la consulta palabra por palabra (aplicando Split (“ ”)). En el método anterior, es donde se genera el archivo “Resultado” donde por cada palabra de la consulta, se generará una lista de los cuerpos de los tweets.

5.2.5 Interfaz

En lo que respecta a la interfaz del algoritmo, se ha diseñado a semejanzas de la propuesta encontrada en el paper fuente [1], pero se han agregado botones adicional como lo muestra la figura 8, el botón “Inicia Stream” será quien ejecute el stream de tweets, el botón “Limpiar Campos” setea en blanco el campo de la consulta y finalmente, el botón “Resultados” el que ejecuta el índice invertido. Una vez generado el índice invertido, lo datos recuperados en virtud a la consulta serán impresos en el archivo de texto de resultados

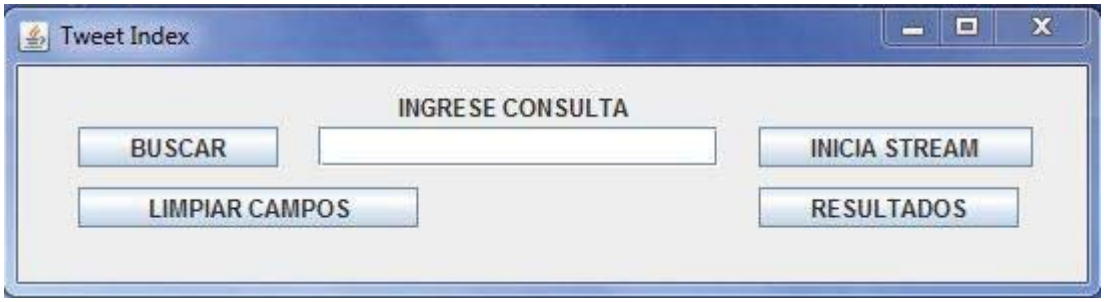

Figura 8 Interfaz Algoritmo

6 Evaluación Experimental

6.1 Criterios a Evaluar

En relación a la evaluación, se estimaron tres criterios representativos para dicho efecto, los cuales darán testimonio de la optimización planteada en [1]. Se definen así, los tres criterios:

6.1.1 Costo de Indexación

La indexación como tal, es un método diseñado para clasificar y buscar información de forma más eficiente y eficaz. La constante problemática será el volumen de datos lo cual es proporcional al tiempo en que se tardará en generar el índice. Por lo tanto, el planteamiento de generar una clasificación previa para indexar lo más probable a utilizar es esencial.

En ésta evaluación, se medirán los tiempos que tardan en indexar el conjunto de datos completo. Para efectos de la presente métrica, el DataSet es acotado a diez mil tweets.

En el gráfico generado se realiza la comparación entre la cantidad de tweets indexados en tiempo real dataset full y parcial, de acuerdo a la clasificación en la función ranking, lo anterior bajo el umbral de cien tweets indexados.

La Figura 9, muestra los valores obtenidos en la evaluación de la presente métrica. A diferencia de lo planteado en [1], la indexación realizada por el modelo TI presenta una singularidad a los 60 tweet indexados, lo cual se justifica en ciertas irregularidades presentes en el dataset luego de los variados filtros y posterior método de graficado. Restando la situación de la singularidad, se aprecia un incremento en los valores de manera muy semejante a lo apreciado en el grafico anterior, de igual forma, el tiempo en promedio que gasta en la indexación full es semejante a los resultados presentados en [1], pero con un leve incremento en el tiempo, lo cual se traduce en el tipo de almacenamiento de datos utilizado. El tiempo fue tomado desde que se inicia el stream de tweets, lo que implica que está sumado el tiempo que demora en guardar el tweet en la matriz para luego ser procesado, que a diferencia de [1] se trataría sólo de una consulta a la base de datos.

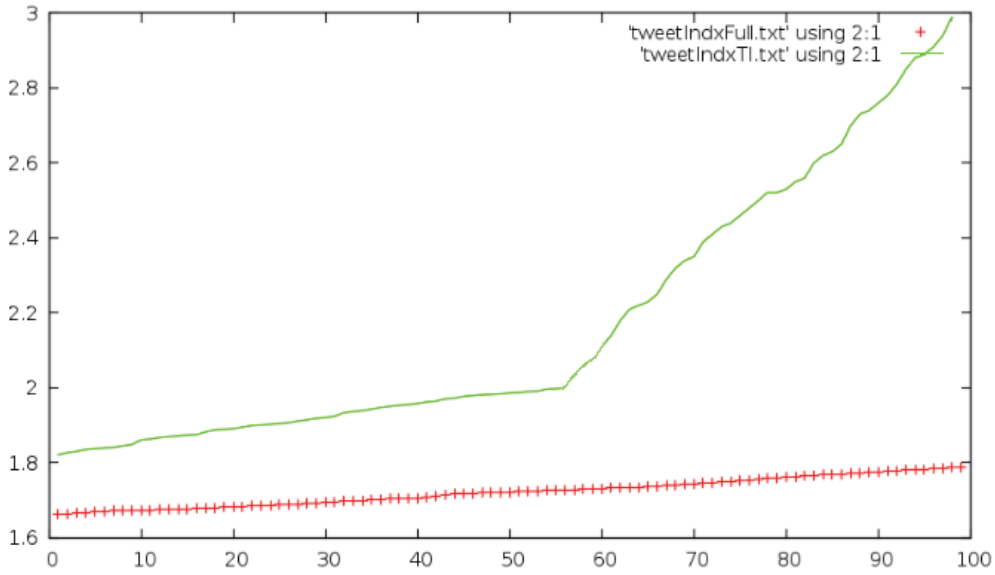

Figura 9 Métrica Obtenida

6.1.2 Ejecución del Proceso de Consulta

La Función Ranking combina la comparación de la consulta en relación al cuerpo de cada tweet. Para su evaluación, se implementará una clasificación de tweets basada en el tiempo de ingreso al sistema, tal estrategia es adoptada por Twitter y Google para la búsqueda en tiempo real.

Uno de los factores más importante es la antigüedad del tweet en el sistema. El esquema basado en el tiempo o antigüedad se comportará en forma casi lineal, por el contrario, el algoritmo implementado aumentará en tiempo mientras más tweets se vean involucrados.

Analizando la Figura 10, se puede apreciar que TI alcanza la indexación de los tweets en mayor tiempo que el modelo que indexa en base a la antigüedad del tweet, lo anterior en virtud al proceso de clasificación más exhaustivo en TI. El modelo basado en tiempo, sólo checkea que el tweet no tenga una antigüedad mayor a un parámetro definido a conveniencia para ser indexado. Es importante destacar que el data set no se encuentra en orden producto de los reiterados filtros, por lo tanto, la consulta deberá recorrer la totalidad del dataset (acotado a ochenta mil tweets) para determinar que tweets a indexar, además la evaluación fue realizada en un minuto.

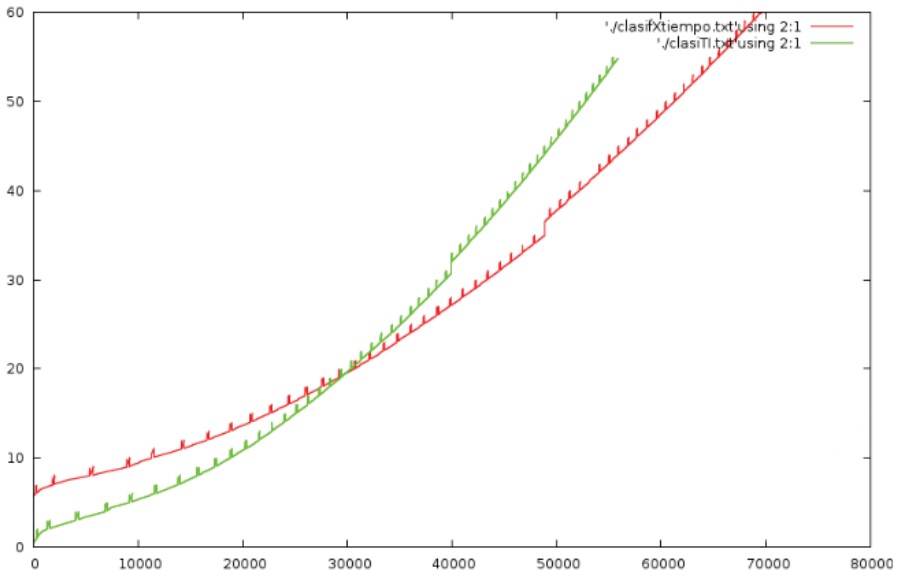

Figura 10 Métrica Generada Tweet/Tipo de Clasificación

Para efectos de todas las métricas evaluadas la consulta utilizada fue “bachelet”, que era uno de las palabras más recurrentes en el DataSource.

6.2 Entorno de Implementación

La implementación del algoritmo se desarrolló en un ambiente con necesarias las siguientes características:

6.2.1 Hardware

- Notebook procesador I5 64 Bit S.G., 4 GB Ram, 1 TB Disco Duro

6.2.2 Software

- SO Linux Debian 7.8 Wheezy 64 bit
- IDE Eclipse Kepler 64 bit
- JDK 1.8
- Gnuplot

7 Conclusiones

En el contexto del uso eficiente y eficaz de datos sumado a la creciente adherencia a las redes sociales, el algoritmo Tweet Index [1] es concebido con la finalidad de poder resolver una falencia que Twitter tenía en su funcionalidad de búsqueda. Sin embargo, la necesidad de una categorización apropiada, más bien adaptable, era necesario para poder establecer prioridades en base al contenido o cuerpo del tweet, de esa manera podría seleccionar los tweets más relacionados a la consulta y de menor antigüedad en el sistema, considerando que estadísticamente se prefieren los resultados más nuevos en menos precio de los más antiguos.

El someter el algoritmo a una serie de evaluaciones, representaba la mayor parte de esta memoria, por tanto, la elección de cada métrica a evaluar se analizó y eligió en necesidad de evaluar los aspectos más relevantes del algoritmo.

Al comprender las implicancias de trabajar con grandes volúmenes de datos suministrados en este caso por las redes sociales, y el requerimiento de una eficiencia y eficacia para el diseño e implementación de una solución a una problemática con creciente complejidad, se llegó a las siguientes conclusiones:

- Requerimientos del sistema: Si bien en Tweet Index no especifica ningún tipo de hardware mínimo, se estimó que es necesario contar con un sistema base para la presente memoria desarrollada, el cual consta de un procesador Core I5 de segunda generación, 4 Gigabytes de Memoria RAM y un disco duro de 500 Gigabytes. Este resultado fue obtenido luego de correr el mismo proceso en dos sistemas de menor capacidad.
- Control de actualizaciones: Por cada Tweet entrante se producirá el cálculo de su ranking, si dicho valor está dentro del rango establecido este será indexado, la problemática surge al actualizar el índice invertido que es proporcional a la cantidad de tweet indexados. Por lo tanto, el correcto cálculo de si un tweet debe ser indexado es primordial, ya que evitará que aumenten la cantidad de veces que se tenga que actualizar completamente el índice.
- Función Ranking: El gran fuerte del algoritmo TI es su función Ranking, la cual reduce el campo de búsqueda a menos del 50% del DataSet, sin embargo, se explica en [1] que pueden existir tantas funciones Ranking como se requieran. De lo anterior, él se constata el aporte de la función Ranking, sin embargo, el problema radica en el aumento del tiempo de ejecución dado el mayor tamaño del conjunto de datos, por lo tanto, es necesario generar un nuevo procedimiento que sea adaptable a medida que ingresan más datos y así mantenga una performance estable durante el tiempo.

8 Bibliografía

- [1]C. Chen, F. Li, B. C. Ooi, and S. Wu,"TI: An efficient indexing mechanism for real-time search on tweets," in SIGMOD, 2011, pp. 649-660.
- [2]Lingkun Wu, Wenqing Lin, Xiaokui Xiao, Yabo Xu, "LSII: An indexing structure for exact real-time search on microblogs," icde, pp.482-493, 2013 IEEE 29th International Conference on Data Engineering (ICDE), 2013
- [3]M. Busch, K. Gade, B. Larson, P. Lok, S. Luckenbill, and J. Lin, "Earlbird: Real-time search at twitter," in ICDE, 2012, pp. 1360-1369.
- [4]L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. In Technical Report, Stanford University, 1998.
- [5]P. Seshadri and A. N. Swami. Generalized partial index. In ICDE, pages 420-427, 1995.
- [6] Learning Perl Third Edition 2001, Randal L. Schwartz and Tom Phoenix, published by O'Reilli & Associates, Inc., 101 Morris Street, Sebastopol, CA 95472.