UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

IDENTIFICACIÓN DE CONTENIDO MULTIMEDIA RELEVANTE A PARTIR DE EVENTOS UTILIZANDO SU INFORMACIÓN SOCIAL

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

MAURICIO DANIEL QUEZADA VEAS

PROFESORA GUÍA:
BÁRBARA POBLETE LABRA

MIEMBROS DE LA COMISIÓN:
SERGIO OCHOA DELORENZI
MAURICIO MARÍN CAIHUAN

SANTIAGO DE CHILE
DICIEMBRE 2012

Resumen

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Una dedicatoria corta.

Agradecimientos

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Índice general

Índice de figuras

Capítulo 1

Introducción

Al igual que en el buffet de un restaurante, por mucho que se quisieran comer todos los platos favoritos, es imposible comer todo lo que uno quisiera por razones obvias. Una posibilidad es probar un poco de cada comida, para así saber qué es lo más delicioso y comer hasta hartarse.

Pero, ¿qué hacer si hay demasiados platos y no se conocen todos? de alguna manera hay que saber cuáles hay que probar, si el objetivo es comer lo mejor posible. Un amigo puede recomendar una u otra comida, lo cual puede servir para orientarse. Entonces se pueden escoger pequeñas muestras de acuerdo a las recomendaciones.

Complicando más el escenario, qué pasa si este restaurant tiene además música en vivo, y por alguna razón, se tiene el privilegio de escoger qué escuchar. En este caso, ya no es posible "probar" un poco de cada tipo existente, no sólo por la cantidad, sino porque no es posible juzgar un grupo musical por una canción o un extracto de ella. Si se quiere tener la mejor velada, pudiendo disfrutar de cada uno de los panoramas que ofrece, es necesario tener algo de información para poder escoger.

Pasando a un contexto diferente, supóngase que este gran buffet es la Web y los distintos platos corresponden a contenido publicado en ella. Por lo tanto, dada la gran cantidad de información disponible, se hace necesario poder encontrar lo más atractivo de acuerdo a la preferencia del usuario o de los usuarios. Nótese que se está haciendo otra suposición importante con esta analogía, y es que se está considerando que la información es íntegramente para ser *consumida*, y no, por ejemplo, para generar más contenido, conocimiento, o para ser utilizada por máquinas, etc. Dentro de este contexto se plantea la pregunta de cómo seleccionar el contenido más atractivo dentro de todo lo que hay disponible en un momento dado.

Siguiendo el razonamiento de la analogía, una manera de poder seleccionar sólo el contenido más "atractivo" (de acuerdo a las preferencias del usuario), es probar un poco de cada uno. O bien, generar un *resumen* de cada documento y presentarlo al usuario.

Sin embargo, tal como se dijo antes, no es posible hacer lo anterior para contenido que no puede ser resumido directamente para ser consumido (la música como tal, o bien, vídeos o imágenes). En esta perspectiva, sólo las recomendaciones pueden ayudar a determinar lo más conveniente

de acuerdo al usuario.

Este trabajo consistió en el desarrollo de una primera aproximación que permite generar resúmenes automáticos de eventos bien definidos a partir de los documentos Web que hablan de éstos. Los documentos son considerados *multimedios*: texto, imágenes, vídeos, sonidos; es decir, no necesariamente texto. El contenido es filtrado o seleccionado de acuerdo a indicadores sociales: los objetos más *tomados en cuenta* en la Web son considerados más importantes.

El sistema implementado consideró dos tipos de eventos: noticias y conciertos musicales. Para obtenerlos, se utilizó el servicio de Google News[1] y Last.fm[2]. Para medir la relevancia de los documentos y obtener los mismos se utilizó la red social Twitter[3], que provee una *Aplication Programming Interface* (o API) para realizar búsquedas y obtener información sobre los *tweets* o mensajes cortos que publican los usuarios de la red.

La estructura de este informe es como sigue: en este capítulo se comenta el contexto dentro del cual se desarrolló este sistema, las contribuciones realizadas, los objetivos y una descripción general de la solución; en el siguiente capítulo se discute el estado del arte y el marco teórico del cual se desprende este trabajo; el capítulo 3 describe más en detalle el problema a resolver, su relevancia y sus dificultades, para luego, en el capítulo 4, describir la solución implementada, más un par de casos de estudio sobre los resultados obtenidos, para finalizar con las conclusiones de este trabajo en el capítulo 5.

1.1. Contexto y Motivación

La tasa de crecimiento de la cantidad de datos en la Web, y en particular, en las *redes sociales online* (OSN, *Online Social Networks*), es de tal magnitud que se vuelve necesario encontrar formas de filtrar y buscar sólo la información relevante dentro de todas las fuentes que hablan del mismo tópico o tema.

En el contexto de las redes sociales online, cada día se publican millones de *actualizaciones de estado* (mensajes breves sobre el estado actual del usuario) con respecto a distintos tópicos, ya sean conversacionales, personales o sobre algún evento en particular[4]. Además, el auge de los teléfonos inteligentes o *smartphones* con mayor capacidad de procesamiento e integrados con todo tipo de sensores (cámaras fotográficas, de vídeo, acelerómetro, barómetro, osciloscopio, etc.), hace posible el generar aun más información y e incluso en tiempo real sobre lo que acontece en el mundo, en internet, o bien sobre el estado particular de cada usuario.

Este aumento y evolución de la generación de datos no sólo influye en la riqueza en la variedad de éstos, sino también en el comportamiento de los usuarios a lo largo del tiempo. Actualmente, una gran parte de los usuarios valora más el contenido de tipo multimedia (imágenes y videos) en las redes sociales online[5]. Se hace entonces necesario encontrar formas para satisfacer estas

[1] http://news.google.com
[2] http://last.fm
[3] http://twitter.com
[4] Pear Analytics. Twitter Study http://es.scribd.com/doc/18548460/Pear-Analytics-Twitter-Study-August-2009
[5] The Rise of Visual Social Media http://www.fastcompany.com/3000794/rise-visual-social-media. En el artículo

necesidades de los usuarios, las cuales ya han sido abordadas en parte, como la generación de resúmenes automáticos orientado a motores de búsqueda, o la determinación de la relevancia tanto de documentos en la Web como de actualizaciones de estado en las redes sociales.

Surge como motivación el poder identificar y extraer contenido relevante de la Web, a partir de eventos, y además avanzar un paso más arriba en el nivel de abstracción: considerar los documentos no por su contenido textual, lo que permite abarcar imágenes, vídeos, sonidos y multimedios. Algunas aplicaciones directas de esto son, entre otras:

- Ayudar al trabajo periodístico mediante una colección de contenido multimedia relacionado a un evento noticioso. Por ejemplo, la versión online de Radio Biobío[6] frecuentemente publica breves artículos sobre sucesos que tienen impacto en las redes sociales, mostrando un pequeño conjunto de mensajes con comentarios de la gente[7]. Una aplicación directa involucraría considerar además contenido multimedia, y organizar este contenido de acuerdo a la relevancia que tiene dentro de las redes.
- Enriquecer la búsqueda en la Web a través de contenido multimedia. Una persona buscando información sobre un concierto podría obtener imágenes y vídeos de éste fácilmente una vez identificado el concierto.
- Siguiendo lo anterior, un grupo musical podría obtener toda la información multimedia asociada a su concierto, tanto para sus fans como para ellos mismos, potenciando su popularidad.
- Poder distinguir entre eventos similares rápidamente. Por ejemplo, un usuario que desee obtener información sobre "Gaza", puede referirse tanto a la banda de música como al conflicto en Israel. El poder distinguir rápidamente mediante una imagen o un vídeo acelera mucho el proceso. *Una imagen vale más que mil palabras.*

El sistema implementado es una primera aproximación que puede satisfacer los ejemplos mencionados.

1.2. Objetivos

1.2.1. Objetivo general

El objetivo principal de este trabajo fue el de poder evaluar e implementar en la práctica un sistema de extracción de contenido multimedia basado en la información social asociada a este contenido.

---

se menciona un estudio sobre comportamiendo y preferencias de los usuarios en las redes sociales hecho por ROI Research: http://www.slideshare.net/performics_us/performics-life-on-demand-2012-summary-deck

[6] http://www.biobiochile.cl/

[7] Como muestra: http://www.biobiochile.cl/2012/12/01/aporte-de-lustrabotas-de-santiago-a-la-teleton-provoca-admiracion-en-redes-sociales.shtml, y http://www.biobiochile.cl/2012/12/01/rechazo-provocan-condicionamientos-de-compra-de-ripley-y-unimarc-para-donar-a-la-teleton.shtml

1.2.2. Objetivos específicos

- Abstraerse del problema de identificación de eventos a partir de documentos Web, llevando a cabo una metodología de obtención de datos simple.
- Implementar un modelo de *clustering* para separar los documentos en *subtópicos* de cada evento, sin considerar el tipo de contenido de estos documentos.
- Analizar la efectividad del sistema implementado, evaluando casos de estudio.

1.3. Descripción general de la solución

Este trabajo puede considerarse como un punto de partida para el desarrollo de un modelo de recuperación de contenido multimedia, similar a lo que corresponde a la generación de resúmenes automáticos para múltiples documentos. En particular, se implementó un sistema que permite considerar distintas estrategias para continuar desarrollando en el futuro. Además:

- Se llevó a cabo una metodología para la obtención de documentos y enriquecerlos con datos obtenidos de fuentes sociales;
- Se implementó un procedimiento que separar estos documentos en *clusters*, *sin considerar su contenido*. Sólo se utilizó la información social asociada; y
- Se implementó además una forma de *rankear* u ordenar los resultados de acuerdo a *relevancia*, siendo ésta medida de acuerdo a la información social asociada a los documentos generados.

El sistema implementado puede dividirse en tres componentes principales:

1. La que obtiene descripciones de eventos a partir de fuentes de éstos en la Web, enriqueciéndolos con información social;
2. Otra componente que procesa y separa los documentos a partir de la información social; genera *objetos Web* y los separa en subtópicos de cada evento, respectivamente; y
3. La componente que entrega los k documentos más relevantes por cada evento obtenido, basándose en los subtópicos identificados.

Se utilizaron las API de Google News como de Last.fm para la obtención de eventos: noticias y conciertos, respectivamente. Para el enriquecimiento de los eventos se utilizó la información social que provee Twitter y su API de búsqueda de *tweets*. De la misma forma, se consideraron los metadatos de los mismos mensajes para medir la relevancia de los documentos generados.

Un documento es identificado por la URL que lo ubica en la Web. El contenido no es más que la concatenación de los tweets que mencionan al documento. Se realizó una limpieza y preprocesamiento de los datos, quitando las *stopwords* y realizando *stemming* sobre el contenido en texto. Luego, se aplicó *tf-idf* sobre los documentos, representándolos como vectores en el *space vector model*. Para identificar los subtópicos de un evento se utilizó el algoritmo de clustering k-means sobre los vectores.

Para el ranking de los documentos se decidió usar una ponderación simple sobre una serie de indicadores que dependen de los tweets y de las URLs de cada evento.

Entre las herramientas utilizadas, se usó lenguaje de programación Python, varias librerías para el manejo de datos (tales como `nltk`, `scipy`, `scikit-learn`, por nombrar las más importantes), el sistema de almacenamiento Redis, entre otras herramientas que son mencionadas en la descripción detallada de la solución.

Capítulo 2

Antecedentes

Para poder describir correctamente tanto el problema como la solución implementada, es necesario dar los punteros y conceptos básicos que los involucran. En este capítulo se discutirán los siguientes tópicos:

- La red social Twitter, la cual es utilizada como fuente de datos y documentos para este trabajo.
- Clustering de documentos, y en general, estrategias de clustering, las cuales tienen muchas aplicaciones prácticas. En este trabajo fue utilizada una de estas estrategias para poder determinar los subtópicos de cada evento.
- La identificación automática de eventos, la cual, si bien se utilizó un enfoque más simple para este trabajo, sirve para indicar en qué aspectos es posible extender este trabajo en el futuro.
- Resúmenes automáticos: una sucinta definición, y algunos enfoques que han existido en el tiempo para este procedimiento.
- Ranking de documentos, o cómo generar órdenes de acuerdo a relevancia.

Casi todos estos tópicos, a excepción del primero, involucran técnicas de Minería de Datos, Recuperación de la Información y Aprendizaje de Máquinas, entre otras áreas.

2.1. Twitter

Twitter es una red social online que permite conectar a personas mediante la comunicación de mensajes cortos, rápidos y frecuentes[1]. Estos mensajes son publicados en el perfil del usuario que los emite, pueden ser vistos directamente por los seguidores de este usuario o ser vistos directamente en el perfil o buscándolos mediante una funcionalidad que provee el servicio. Además, un usuario puede *seguir* a otros para poder ver en su *timeline* o perfil privado los mensajes de todos a quienes sigue.

[1] https://support.twitter.com/groups/31-twitter-basics/topics/104-welcome-to-twitter-support/articles/13920-get-to-know-twitter-new-user-faq

Estos mensajes, o *tweets*, sólo son cadenas de caracteres con metadatos que el mismo servicio asigna una vez enviado a la red social. Desde sus inicios (año 2007) se han añadido algunas capacidades adicionales a estos mensajes, como la de poner URLs, imágenes, vídeos, etc. Además, existen varias convenciones que han surgido a lo largo del tiempo. A continuación se describe una lista de tipos de mensajes que existen en Twitter, originados por estas convenciones:

1. Respuestas o *replies*: son mensajes del tipo `@usuario [texto]`, que ocurren usualmente en una conversación entre dos usuarios.
2. Menciones o *mentions*: un poco más general a una respuesta, el nombre del usuario mencionado puede estar en cualquier parte del mensaje. La diferencia semántica es que no se le habla "directamente" al usuario mencionado, como en una respuesta, sino que sólo es mencionado por si el mensaje es de su interés o no.
3. *Retweets*: son mensajes del tipo `RT @usuario: [texto]`. Ocurren cuando se quiere compartir el mensaje de otro usuario, o citarlo para mencionarlo en el mismo mensaje.
4. *Hashtags*: son palabras precedidas por el caracter #, que indican un identificador a cierto evento o suceso dentro o fuera de la red. Suelen usarse para categorizar de cierta forma un tópico, pero son libres de usarse como los usuarios quieran.
5. Mensaje simple: un mensaje sin menciones ni hashtags.

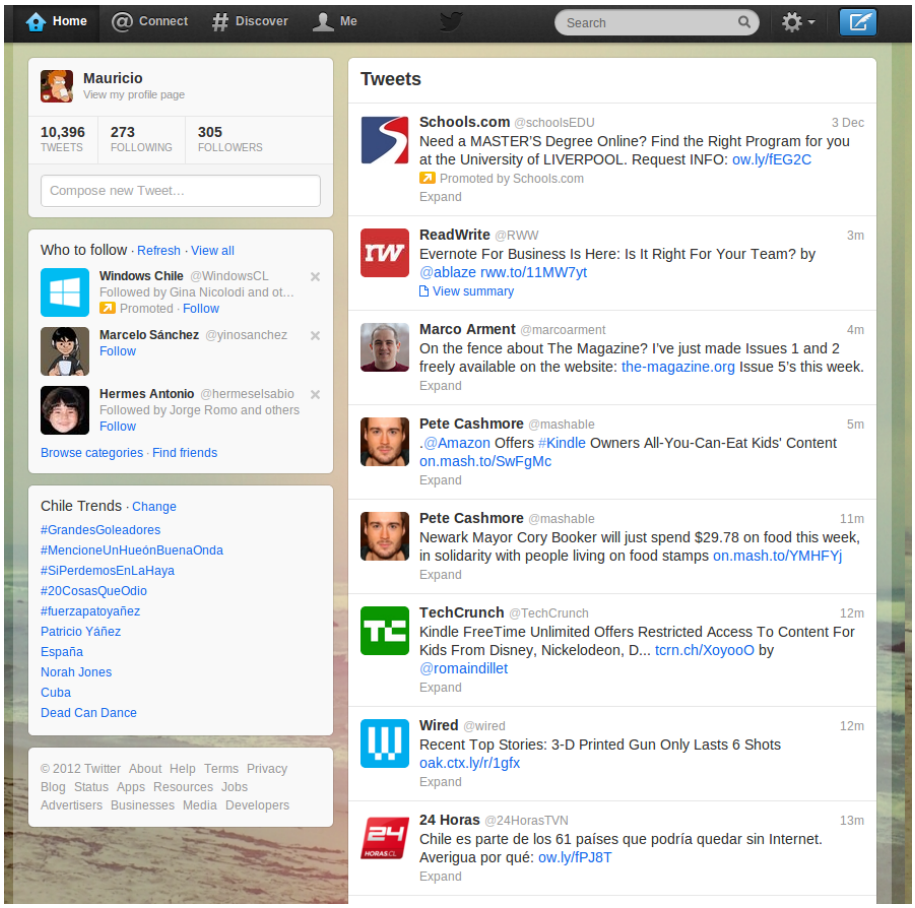

Figura 2.1: Timeline de Twitter. En éste se ve una lista en orden cronológico de los tweets generados por los usuarios que sigue el usuario actual. Además, el sitio incluye tweets promocionados por cuentas que pagan por dicho servicio, como es el caso del primer tweet en la lista.

Ejemplos:

- Mensaje simple: `Jason Funk disipa patitos`;
- Respuesta: `@jason estoy de acuerdo con lo que dices`;
- Mención: `creo que @jason es una cumbre de sabiduría`;
- Retweet: `RT @jason: Jason Funk disipa patitos`; y
- Hashtag: `Estoy escribiendo mi memoria #dcc #summarization`

Estos mensajes están limitados a 140 caracteres de extensión. Sumando esto a la integración de la red con otros servicios y dispositivos, y a la cantidad de mensajes publicados cada minuto, permite utilizar esta red como una gran fuente de datos.

Twitter además provee varios servicios adicionales, como por ejemplo, un servicio de acortamiento de URLs, para permitir incluir una URL larga sin perjudicar la cantidad de caracteres restantes para el mensaje; un servicio de alojamiento de fotos y vídeos, para hacer más sencilla la publicación de mensajes multimedia desde dispositivos móviles; un servicio de búsqueda que permite buscar una cantidad determinada de tweets sobre un término de búsqueda o un hashtag, entre otros servicios.

2.2. Clustering de documentos

El análisis de clusters o clustering es el proceso de encontrar grupos de objetos, tal que los objetos en un grupo sean similares entre sí (o relacionados) y que sean diferentes (o no relacionados) a los objetos de otros grupos. Algunas aplicaciones del análisis de clusters son, entre otras:

- Encontrar clusters naturales y describir sus propiedades (*data understanding*);
- Encontrar agrupamientos útiles (*data class identification*);
- Encontrar representantes de grupos homogéneos (*data reduction*);
- Encontrar perturbaciones aleatorias de los datos (*noise detection*);
- Encontrar objetos inusuales (*outliers detection*);
- etc.

Se denomina cluster a un grupo de objetos, mientras que clustering puede referirse al conjunto de clusters o al proceso de encontrarlos. Existen diversos tipos de procesos de clustering, una de las distinciones más importantes es entre los clusters jerárquicos y los particionales:

- Un clustering jerárquico es un conjunto de clusters anidados, organizados más bien como un árbol. Cortando el árbol en cualquier nivel da como resultado un clustering potencialmente distinto.
- El clustering particional es un conjunto de clusters de forma de partición del conjunto total, es decir, cada objeto está contenido en un sólo subconjunto o cluster.

Para describir el proceso aplicado a documentos, primero se describirán los modelos de representación más importantes para, de forma de definir la noción de documento y luego los algoritmos

de clustering aplicados a éstos.

2.2.1. Modelos de representación de documentos

Standard Boolean Model

El modelo booleano es un modelo de representación de documentos. En él, los documentos son vectores de *términos*:

$$\mathrm{d} = (w_1, w_2, \ldots, w_m)$$

Donde un término es un n-grama del texto del documento.

Definición 2.1 *Un n-grama es una secuencia contigua de n ítems a partir de un texto.*

La definición de ítem dependerá de la aplicación: en lenguaje natural el texto a su vez dependerá del idioma, por ejemplo, si el texto está en inglés o en japonés, la distinción entre palabras es distinta para cada uno.

No existe una medida de "similitud" como tal en este modelo, sino que se considera el calce exacto entre los términos de una query q y un documento d. La query puede ser una consulta hecha por un usuario al conjunto de documentos, o bien un documento del mismo conjunto.

Una consulta es una fórmula de lógica proposicional que pide los documentos que contengan o no ciertos términos.

Bag of words Model

En el modelo Bag of Words un documento d es representado como un conjunto de pares $(w_\mathrm{i}, f_\mathrm{i})$, $\mathrm{i} \in [1..m']$, donde w_i es un término del documento, f_i es la frecuencia de w_i en el mismo, y m' es la cantidad de términos distintos en el documento.

La ventaja principal por sobre el modelo anterior es que permite hacer calces parciales entre consultas y documentos. Este modelo es comúnmente utilizado para hacer clasificación de documentos, por ejemplo, para determinar si un correo electrónico es o no spam.

Vector Space Model

El *Vector Space Model* es un modelo un poco más general que el anterior. Un documento d es representado como un vector de pesos asociados a los términos:

$$\mathrm{d} = (f(w_1), f(w_2), \ldots, f(w_m))$$

Cada dimensión de este vector corresponde al peso asociado a un término del documento.

El peso puede ser directamente la frecuencia del término dentro del documento:

$$\mathrm{freq}(w, \mathrm{d}) = |\{w : w \in \mathrm{d}\}|$$

O bien, normalizar esta frecuencia para evitar que documentos más largos sean más relevantes sólo por su extensión:

$$\mathrm{tf}_0(w, \mathrm{d}) = \begin{cases} 1 & \text{si } w \in \mathrm{d} \\ 0 & \text{si no} \end{cases}$$

tf_0 o *Term Frequency* es una primera aproximación a medir la frecuencia de un término en un documento. Sin embargo, esta nueva aproximación sufre de la desventaja de que ahora un documento con una ocurrencia del término será igual de relevante que algún documento que mencione varias veces el término (por ejemplo, un diccionario que tiene el término una vez contra un artículo sobre el tema). Otra alternativa, considera no castigar demasiado a los documentos con pocas ocurrencias, pero tampoco beneficiar mucho a los que tengan muchas:

$$\mathrm{tf}_1(w, \mathrm{d}) = 1 + \log(\mathrm{freq}(w, \mathrm{d}))$$

La solución más utilizada considera la proporción con respecto al término con más ocurrencias, para esto, se normaliza por el tamaño del documento:

$$\mathrm{tf}(w, \mathrm{d}) = \frac{\mathrm{freq}(w, \mathrm{d})}{max\{\mathrm{freq}(t, \mathrm{d}) : t \in \mathrm{d}\}}$$

Otro problema que tiene utilizar esta medida como los pesos de los términos, es que un término muy repetido entre todos los documentos que hablan de un mismo tema puede significar que no es muy relevante (por ejemplo, las *stopwords* o palabras vacías, son por lo general las preposiciones, artículos, pronombres, etc.). Para esto, se considera además ponderar por el inverso de la frecuencia entre los documentos; es decir, un término frecuente entre todos los documentos ve su peso castigado a diferencia de un término que sólo es mencionado una vez en un documento. Esta medida es llamada *Inverse Document Frequency* o idf:

$$\mathrm{idf}(t, D) = \log \frac{|D|}{1 + |\{\mathrm{d} \in D : t \in \mathrm{d}\}|}$$

Finalmente, el peso de un término es la ponderación de su frecuencia dentro del documento con el inverso de la frecuencia entre los documentos, o tf-idf:

$$\text{tf-idf}(t, \text{d}, D) = \text{tf}(t, \text{d}) \times \text{idf}(t, D)$$

Para el resto de este trabajo se considerará la representación de documentos en este modelo.

2.2.2. Medidas de similitud

A lo largo de los años, dos formas han sido las usuales para determinar similitud entre documentos[3]: calculando el coseno de los dos vectores, y la distancia euclidiana.

El coseno de dos documentos d_i y d_j se define como

$$\cos(\text{d}_\text{i}, \text{d}_j) = \frac{\text{d}_\text{i}^t \text{d}_j}{\|\text{d}_\text{i}\| \|\text{d}_j\|}$$

Si ambos d_i y d_j son iguales, entonces la fórmula anterior se evalúa a 1, y 0 si no tienen nada en común, es decir, ambos vectores son ortogonales.

La distancia euclidiana entre d_i y d_j se define como

$$\text{d}(\text{d}_\text{i}, \text{d}_j) = \sqrt{(\text{d}_\text{i} - \text{d}_j)^t (\text{d}_\text{i} - \text{d}_j)} = \|\text{d}_\text{i} - \text{d}_j\|$$

En este caso, si la distancia es 0, entonces ambos documentos son idénticos. Y si ambos son ortogonales, la distancia será $\sqrt{2}$.

Si ambos documentos están normalizados (su norma es 1), entonces ambas medidas serán muy parecidas, aun así siendo una de similitud y la otra de distancia.

2.2.3. Algoritmos de clustering

Se considerará la siguiente definición para el problema de clustering:

Dado un entero k y un conjunto de n puntos en $\mathbb{R}^v$, el objetivo es determinar k puntos tal que se minimice ϕ, la suma de las distancias al cuadrado de cada punto y su centro más cercano.

Este problema es NP-hard, sin embargo, existe un algoritmo que permite realizar una búsqueda local y determinar k centros en un tiempo razonable[2].

K-means es un algoritmo de clustering particional, en el cual cada cluster tiene un *centroide* asociado, esto es, típicamente, un punto el cual es el promedio de todos los puntos del cluster, y no necesariamente corresponde a un dato real.

[2]Lloyd, S.; , "Least squares quantization in PCM," Information Theory, IEEE Transactions on , vol.28, no.2, pp. 129-137, Mar 1982

El algoritmo genera k clusters, donde k es un parámetro del algoritmo, tal que cada punto pertence al cluster cuyo centroide es el más cercano a éste.

Algoritmo 1: K-means

Data: $k > 0$, conjunto D de puntos
Result: Asignación de cada punto a un cluster i $\in [1..k]$
1 Seleccionar k puntos como centroides iniciales;
2 **while** *Los centroides cambien a cada iteración* **do**
3 Formar k clusters asignando todos los puntos al centroide más cercano;
4 Recalcular los centroides de cada cluster;

Los clusters generados dependerán de la elección inicial de los centroides, y usualmente basta con pocas iteraciones para la convergencia. La complejidad de este algoritmo es $O(nkIv)$, donde n es el número de puntos, k el parámetro de la cantidad de clusters, I es la cantidad de iteraciones que hará el algoritmo y v es la cantidad de dimensiones de los vectores.

K-means también puede verse desde el enfoque de optimizar una función criterio. La medida más común es la *suma del error cuadrado* o Sum of Squared Error (SSE). Para cada punto, su *error* es la distancia al cluster más cercano:

$$SSE = \sum_{\mathrm{i}=1}^{k} \sum_{x \in C_{\mathrm{i}}} \mathrm{dist}^2(m_{\mathrm{i}}, x)$$

Donde x es un punto en el cluster C_{i}, y m_{i} es su centroide. Dados dos clusters, en cada paso se escoge el que tenga menor error. Un problema importante de K-means es escoger los centroides iniciales. Dado que la probabilidad de acertar a los centroides "reales" (precisión) es muy baja, es posible que un clustering tenga un mal error cuadrático, de hecho, el radio competitivo con la solución óptima es no acotado incluso para k y n fijos. Existen varias formas de evitar este problema, por ejemplo, ejecutando varias veces K-means, usar clustering jerárquico para determinar los centroides iniciales, determinar más de k centroides y luego elegir los k más separados, etc.

Un enfoque utilizado es el de usar el algoritmo K-means++[1], el cual tiene como objetivo determinar los centroides iniciales para K-Means. Este algoritmo garantiza un radio competitivo de $O(\log k)$ con respecto al error cuadrático esperado del óptimo. El algoritmo K-means++ escoge los centroides con cierta probabilidad que depende de la distancia al centroide más cercano, lo cual garantiza una buena elección inicial, y luego continúa con el algoritmo usual. Sea $D(x)$ la distancia de x al centroide más cercano. El algoritmo es como sigue:

Algoritmo 2: K-means++

Data: $k > 0$, conjunto D de puntos
Result: Asignación de cada punto a un cluster i $\in [1..k]$
1 Escoger un centroide c_1 al azar uniformemente de D;
2 Escoger un nuevo centroide c_{i}, escogiendo $x \in D$ con probabilidad $\frac{D(x)^2}{\sum_{x' \in D} D(x')^2}$;
3 Repetir paso 1 hasta haber escogido k centros;
4 Proceder con K-means estándar;

2.2.4. Evaluación de clusters

Existen dos enfoques para evaluar clusterings: validación interna y externa[3] La validación interna considera evaluar el clustering con respecto a los datos que han sido agrupados (es decir, sin información adicional), mientras que la evaluación externa considera datos adicionales, como datos pre-clasificados (conjunto de prueba, de entrenamiento, etc), o los clusters "reales".

Por otra parte, dos de las métricas más usadas para evaluar clusterings, segun [3] son:

- **Entropía**: mide cómo son distribuidas las clases de documentos entre cada cluster. Se define formalmente como

$$E(C_r) = -\frac{1}{\log q} \sum_{\mathrm{i}=1}^{q} \frac{n_r^{\mathrm{i}}}{n_r} \log \frac{n_r^{\mathrm{i}}}{n_r}$$

  donde q es el número de clases en el dataset, y n_r^{i} es el número de documentos de la i-ésima clase que fue asignado al r-ésimo cluster C_r. La entropía del clustering se define como la suma ponderada de la entropía de cada cluster:

$$E(\mathfrak{C}) = \sum_{r=1}^{k} \frac{n_r}{n} E(C_r)$$

  Un clustering perfecto tendrá clusters tal que cada cluster contenga documentos de una sola clase, en ese caso la entropía será 0. En general, conviene tener bajos valores de entropía.

- **Pureza**: mide de qué forma cada cluster contiene documentos de una clase. La pureza de un cluster C_r se define como

$$P(C_r) = \frac{1}{n_r} max_{\mathrm{i}}\{n_r^{\mathrm{i}}\}$$

  la cual es la fracción del tamaño del cluster.....

2.3. Identificación automática de eventos

La identificación automática de eventos consiste en, dado un conjunto de documentos, donde cada documento está asociado a un evento (desconocido), es poder particionar el conjunto de documentos en clusters, de forma que cada cluster corresponda a todos los documentos asociados a un evento.

Se considerará la definición de "evento" dada por [2].

Definición 2.2 *Un* evento *es un suceso que ocurre en un período de tiempo determinado y en un lugar específico.*

[3] http://en.wikipedia.org/wiki/Cluster_analysis#Evaluation_of_clustering_results

El poder identificar eventos a partir de documentos publicados en los medios sociales permite mejorar la navegación de estos eventos, al mejorar la búsqueda tanto local como de motores de búsqueda.

2.4. Resúmenes automáticos

2.4.1. Evaluación de resúmenes

2.5. Ranking de documentos

Capítulo 3

Especificación del Problema

El problema a resolver consiste en poder *identificar* y *entregar contenido multimedia relevante* en la Web a *consumidores* de contenido.

Esta descripción por ser muy general necesita ser especificada un poco más:

1. El **identificar** se refiere a poder distinguir entre varios documentos asociados a un mismo evento cuáles de estos comparten o tienen cierta propiedad especial, que debe ser definida.
2. El poder **entregar** implica que no sólo debe poder identificarse este contenido, sino también ser entregado a un usuario o a una consulta realizada.
3. El **contenido multimedia** abarca documentos de todo tipo en la Web: texto, imágenes, vídeos, sonidos, binarios, y multimedios (una combinación de los anteriores). Es decir, se abstrae la noción de documento en este sentido.
4. La **relevancia** comprende la propiedad que debe ser identificada. Cómo medir la relevancia tiene muchas respuestas posibles, sin embargo, el enfoque del problema consiste en medir la relevancia hacia el *usuario*, o bien, *consumidores* de contenido..

El énfasis está en los puntos 3 y 4: una solución a este problema no debe basarse en el contenido de los documentos. Por lo mismo, tampoco es posible medir la relevancia de los documentos basándose en su contenido.

Capítulo 4

Descripción de la Solución

La solución implementada consistió en generar un resumen utilizando documentos representados apropiadamente con la información de medios sociales que los mencionan. La relevancia de estos documentos también se obtuvo con la información social más otros indicadores de la solución parcial.

Para esto, se asume que existe una fuente de eventos, de esta forma, el enfoque de la solución radica principalmente en la generación de los resúmenes y no en la identificación de eventos en medios sociales. Sin embargo, para la implementación sí fue necesario enfrentar el problema de identificar eventos, usando un enfoque práctico para obtenerlos.

La figura FIGURA muestra de manera general el modelo propuesto.

FIGURA MODELO

DESCRIPCION DEL MODELO

En la sección SECCION se describe el modelo utilizado para la implementación, con una especificación detallada de la solución. En la sección SECCION se describe la metodología de desarrollo y la implementación práctica realizada para representar el modelo formal, la cual considera la obtención de datos dentro del proceso. Luego se comentan los problemas técnicos que fueron enfrentados para terminar en la sección SECCION, donde se discuten un par de casos de estudio sobre los resultados obtenidos.

4.1. Descripción detallada

Para enfrentar el problema descrito se decidió utilizar una representación apropiada de los documentos que permita abstraerse de su contenido, utilizando la información social asociada a éstos. A partir de un evento determinado, se identifican los subtópicos del evento utilizando los documentos, y luego, para cada subtópico se determinan los documentos más relevantes utilizando esta información social.

Para esto, fue necesario contar con dos *fuentes de datos*: una fuente de eventos y otra de *contenido social*, en la forma de mensajes y actualizaciones de estado.

Se asumió que estas fuentes satisfacen los siguientes requerimientos:

- La fuente de eventos debe entregar una lista de eventos rápidamente, la cual debe contener los siguientes datos para cada entrada:
  - Un título del evento y *términos asociados*. Los términos asociados son breves frases o palabras que describan al evento, como por ejemplo, tags o etiquetas.
  - Como datos opcionales: breve descripción del evento, fecha de inicio y término, ubicación y direcciones Web.
- La o las fuentes de contenido social deben entregar una lista de mensajes, con algunos metadatos tales como la fecha de creación, si el mensaje fue compartido, etc. Además, algunos datos sobre el autor del mensaje, como la cantidad de conexiones en la red, y en general, datos que permitan comparar dos autores.

Utilizando estas dos fuentes, el siguiente paso luego de obtener una lista de eventos fue enriquecerlos utilizando las fuentes sociales, generando documentos del tipo d $= (s_1, s_2, \ldots, s_m)$, donde s_{i}, i $\in [1..m]$ es un mensaje de alguna fuente social, con los metadatos asociados. El documento es identificado por la URI de algún documento en la Web, de forma que todos los mensajes que contengan una URI en particular, corresponderán al mismo documento d.

A continuación, utilizando alguna representación adecuada (*vector space model*, *bag of words*, etc.), se generaron clusters de documentos de un mismo evento, identificando los subtópicos. Con ellos fue posible generar un resumen que abarcara todos los aspectos del evento, en contraste con seleccionar directamente los documentos más relevantes del evento en su conjunto, lo cual puede dejar puntos de vista sin ser considerados por su extensión.

Finalmente se seleccionaron los $k > 0$ documentos más representativos de cada cluster, utilizando como criterio los metadatos de los mensajes de la fuente social. De esta forma, se ordenan los documentos dejando como más "relevantes" los que más interés atrae de los usuarios.

4.2. Metodología de desarrollo e implementación

La implementación consistió en las siguientes etapas:

- Obtención del dataset de eventos y documentos.

  Para esto, se utilizaron dos fuentes de eventos: Google News y Last.fm para recolectar noticias y conciertos musicales (incluidos festivales), respectivamente. Como fuente de datos sociales se utilizó la red social Twitter, que dispone de una API para realizar búsquedas por *keywords*. Esta etapa comprendió la recolección de eventos y de documentos con información social asociada a éstos.
- Generación de clusters para la identificación de subtópicos para cada evento.

  Una vez identificados los eventos y los documentos asociados, se generaron clusters usando el algoritmo K-means con K-means++ para la inicialización. Se impuso un valor de $k =$

5 clusters por evento. En la sección de casos de estudio se discuten las alternativas y la evaluación de algunos clusters del dataset.

- Extracción de documentos relevantes para cada evento.

  Una vez identificados los subtópicos de cada cluster, se extrajeron de éstos los documentos más relevantes, utilizando la información social de cada uno de ellos, en conjunto con otros indicadores globales del clustering (como que incluyan una URL dentro de las más mencionadas dentro del cluster, entre otras). Estos documentos corresponden al output o salida del sistema.

4.2.1. Obtención de datos

Se describe a continuación el proceso diseñado para la obtención de datos, tanto de eventos como de documentos con sus indicadores sociales respectivos.

Las etapas de generación del dataset son las siguientes:

- Recolección de eventos (noticias y conciertos);
- Enriquecimiento de los eventos existentes mediante tweets; e
- Identificación de documentos a partir de los tweets por cada evento.

  Se recolectaron datos (eventos y tweets) desde el 19 de noviembre de 2012 hasta el 30 de noviembre del mismo año, todos los días desde la medianoche hasta que el proceso termina exitosamente.

Recolección de eventos

Se consideraron dos tipos de eventos para el sistema: noticias y conciertos musicales. Los conciertos incluyen festivales de varios artistas.

- Noticias Para obtener las noticias, se utilizó el servicio de Google News[1]. Existe una API (en proceso de obsolescencia[2], pero funcional a la fecha de este trabajo) que permite obtener no sólo los titulares y breve descripción de cada noticia, sino también un conjunto de entre 4-10 noticias relacionadas de otras fuentes. Esto sirvió para alimentar los términos de búsqueda para la etapa siguiente. Se guardaron los siguientes datos de una noticia:
  - Título,
  - Descripción,
  - URL de la fuente, y
  - Titulares de las noticias relacionadas.
- Conciertos Utilizando el servicio de Last.fm para obtener los conciertos y festivales de una ubicación en particular[3], se obtuvieron los conciertos y festivales de las siguientes ubicaciones:

[1] http://news.google.com
[2] http://googlecode.blogspot.com/2011/05/spring-cleaning-for-some-of-our-apis.html
[3] http://www.lastfm.es/api/show/geo.getEvents

  - Santiago, Chile;
  - Londres, Inglaterra;
  - Glastonbury, Inglaterra;
  - Las Vegas, Nevada, EE.UU.; y
  - Estocolmo, Suecia.
- Título del evento (concierto o festival);
- Artistas que participan; y
- Fechas de inicio y término (esta última no siempre está como dato).

  Además de otros datos descriptivos, como la ubicación, descripción breve, sitio web de la banda o festival, etc.

Cada vez que se obtienen los eventos se vuelven a obtener los conciertos, pero sólo agregando los nuevos. Las noticias siempre son nuevas, aun así por implementación no se consideraron los repetidos.

Enriquecimiento de eventos

Se obtuvieron tweets utilizando el servicio de búsqueda que provee Twitter en su API[4]. El objetivo es enriquecer los eventos con la información social que hay en la Web sobre éstos.

Para cada uno de los eventos obtenidos en la fase anterior, se utilizaron los términos de búsqueda asociados a ellos: los titulares de las noticias relacionadas y los nombres de los artistas para los eventos noticiosos y musicales, respectivamente.

- Para las noticias, se hace una búsqueda en Twitter de los titulares al mismo tiempo en que se obtienen de Google News, y nuevamente al día siguiente, es decir, 2 búsquedas por cada titular de un evento. Se quitan las tildes y caracteres no `ASCII` y las stopwords, para evitar problemas con la implementación y no hacer calce de stopwords en la búsqueda de Twitter.
- Para los conciertos y festivales, se utilizaron los nombres de los artistas y del evento como términos de búsqueda. De acuerdo a la información asociada al evento, se busca por una mayor cantidad de días:
  - Se busca desde un día antes de inicio del evento;
  - Si está presente la fecha de término del evento, se busca cada día dentro del intervalo "fecha de inicio" a "fecha de término" hasta tres días terminado el evento.
  - Si no está presente la fecha de término (por ejemplo, un concierto o un festival de un día), se busca hasta tres días pasada la fecha de inicio.

Identificación de documentos a partir de tweets

Luego de obtener los tweets asociados a cada evento, el siguiente paso fue generar los documentos que fueron usados para la generación de los resúmenes. Nuevamente, el modelo consistió en

[4]https://dev.twitter.com/docs/api/1.1/get/search/tweets

que cada documento se modeló como un vector de palabras, donde el identificador del documento es una URL, y sus componentes corresponden al contenido de los tweets que tienen esa URL en el texto del mensaje.

El caso en el que un tweet no tenía ninguna URL en su contenido fue abordado de la siguiente forma: la URL asociada es una tal que representa al mismo tweet (utilizando el servicio de Twitter), y el contenido de ese documento es el mismo tweet, de forma de no dejar el tweet sin ser representado.

Este proceso fue abordado recorriendo todos los eventos del dataset, observando todos los tweets asociados a cada evento, extrayendo la URL si es que hay alguna y guardando el documento con el nuevo tweet. Se marcan los tweets observados para no tener que repetir el proceso, ya que es intensivo en conexión a la red.

Dada la condición breve de los mensajes publicados en la red social, muchos de los usuarios y/o servicios que publican mensajes con una URL en su interior suelen utilizar *acortadores* (o *url shorteners*) para los enlaces, y así no utilizar mucho espacio dentro de un mensaje. Otra ventaja que ofrecen es que algunos servicios como Bit.ly[5] dan estadísticas sobre los visitantes a estos enlaces (y así saber quiénes vienen de cierta red social u otra, por ejemplo). Twitter, a su vez, actualmente también ofrece acortamiento de URLs por defecto. Esto suele producir que un enlace acortado se resuelva a otro enlace también acortado, por lo que es necesario resolver la URL completa para evitar duplicados o *pseudo-duplicados* (en el caso en que dos URLs sintácticamente distintas apunten al mismo recurso). EN LA FIGURA……

FIGURA DE LINKS CORTOS

Por lo anterior, una vez identificada la URL del texto de un tweet, se resuelve su URL completa (que puede ya serlo de antemano), lo que consume recursos de ancho de banda y tiempo.

4.2.2. Identificación de subtópicos

Una vez recolectados tanto los eventos como generados los documentos asociados, se procedió a identificar los subtópicos de cada evento. Para esto, se utilizó el algoritmo K-means para construir clusters a partir de todos los documentos de un solo evento.

Como se mencionó anteriormente, los documentos consisten en vectores del tipo $d_{URL} = (t_1, t_2, \ldots, t_m)$, donde t_i es el tweet i-ésimo que contiene a URL dentro del texto del mensaje. Para poder aplicar un algoritmo de clustering, fue necesario procesar nuevamente estos documentos para representarlos como vectores usando el vector space model. El procedimiento consta de dos partes, "normalizar" los documentos, limpiando los términos que puedan afectar al clustering, y

[5] http://bit.ly

luego aplicar tf-idf sobre los documentos normalizados:

Algoritmo 3: Preprocesamiento de documentos

Data: Conjunto de documentos D_e
Result: Conjunto de strings D'_e,

```
D'_e ← ∅;
for documento d ∈ D_e do
    d' ← ε;
    for tweet t ∈ d do
        t' ← clean(t);
        d' ← concat(d', t');
    D'_e ← D'_e ∪ {d'};
```

Donde ε es el string vacío, $\texttt{concat}(a, b)$ retorna la concatenación de a y b, y `clean` realiza las siguientes operaciones sobre el tweet:

- Remueve las URLs que contenga el texto;
- Remueve todas las menciones;
- Quita los caracteres "#", dejando los *hashtags* intactos;
- Quita las tildes, acentos, stopwords; y
- Realiza stemming en español o inglés dependiendo del idioma del evento dado por los metadatos de éste. Se utilizó el Snowball Stemmer[6] para esto.

Una vez normalizados los documentos, se convierten a la representación como vectores con pesos por cada término:

Algoritmo 4: Transformación de documentos a vector space model

Data: Conjunto de strings D'_e, vocabulario V de palabras de D'_e
Result: Conjunto de vectores D''_e, representados en vector space model

```
D''_e ← ∅ ;
for documento d ∈ D'_e do
    d' ← map(tf-idf(·, d, D'_e), words(d));
D''_e ← D''_e ∪ {d'};
```

Donde $\texttt{map}(f, l)$ mapea la función o procedimiento f a cada elemento de la lista l, retornando una nueva lista l' en la cual a cada elemento se le aplicó f y `words` retorna una lista con las palabras de d'. Este procedimiento retorna un conjunto de vectores en tf-idf, lo que permite entonces aplicar un algoritmo de clustering para identificar subtópicos.

Se utilizó el algoritmo Mini Batch K-means, el CUAL… COMPLETAR ACA Y MODIFICAR ANTES DONDE DIGA KMEANS‼!!!. Al aplicarlo sobre el conjunto de vectores, éste retorna una *lista de etiquetas* L, de largo n, la cantidad de documentos utilizados. Cada $L_i \in [0..k-1], i \in [1..n]$ indica a cuál cluster corresponde el i-ésimo documento, lo que permite fácilmente filtrar y recuperar los clusters por separado, para pasar a la siguiente etapa.

[6] http://snowball.tartarus.org/

4.2.3. Ranking de documentos

4.3. Desafíos técnicos

4.3.1. Restricciones de la API de Twitter

La API de búsqueda de Twitter permite obtener tweets de acuerdo a un término de búsqueda. Se utilizó este servicio para enriquecer los eventos con información social utilizando como términos de búsqueda tanto los títulos de las noticias como los nombres de los artistas para las noticias y los conciertos, respectivamente.

Funciona de la siguiente forma: cada vez que se hace un request a la URL dada por el servicio, éste retorna a lo más 100 tweets por página, con un máximo de 15 páginas (indicando en el request qué página queremos consultar), dando como total hasta 1500 tweets por búsqueda. Existirán términos de búsqueda que no presenten ningún resultado (ya sea por estar mal escritos o simplemente que no sean un tópico de discusión), o por el contrario, que se generen más tweets que los retornados por la búsqueda por cada ventana de tiempo que demore ésta (por ejemplo, un *trending topic* o tópico que sea muy mencionado en la red social).

Existe una limitación de uso de este servicio: sólo es posible hacer hasta 180 requests por cada 15 minutos, o 1 request cada 5 segundos. Además, sólo retorna tweets de hasta 7 días de antigüedad, y sus resultados no son necesariamente en tiempo real y su estabilidad varía de acuerdo a factores externos.

Los tweets retornados vienen en formato `JSON` (*Javascript Simple Object Notation*), e incluyen varios metadatos sobre el tweet aparte de los principales, como autor, fecha, contenido. Algunos de estos metadatos son:

- Cantidad de *retweets* hechos hasta la fecha;
- Si posee alguna URL o *hashtag* en el texto;
- Si es una *mención* a otro usuario;
- La ubicación de donde se envió el tweet;
- etc.

Además, incluye datos sobre el autor, como por ejemplo:

- Si la cuenta está *verificada*;
- La cantidad de seguidores del usuario;
- Cantidad de amigos (seguidores que también lo siguen);
- Cantidad de tweets;
- Su descripción, y si incluye alguna URL, etc;
- Ubicación (dada por el mismo usuario);
- Fecha de creación de la cuenta;
- etc.

4.4. Casos de estudio

Capítulo 5

Conclusiones

5.1. Resumen del trabajo realizado

5.2. Objetivos alcanzados

5.3. Relevancia del trabajo realizado

5.4. Trabajo futuro

Capítulo 6

Anexos

Bibliografía

[1] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[2] Yiming Yang, Jaime G. Carbonell, Ralf D. Brown, Thomas Pierce, Brian T. Archibald, and Xin Liu. Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems*, 14(4):32–43, July 1999.

[3] Ying Zhao and George Karypis. Criterion functions for document clustering: Experiments and analysis. Technical report, 2002.