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

METODOLOGÍA PARA EXTRAER INTERESES DE USUARIOS DE TWITTER PARA GENERACIÓN DE RECOMENDACIONES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

MARIO CASTRO SQUELLA

PROFESOR GUÍA:

JOSÉ ALBERTO PINO URTUBIA

MIEMBROS DE LA COMISIÓN:

JOHAN FABRY

JORGE AUGUSTO OCTAVIO OLIVOS ARAVENA

SANTIAGO DE CHILE
NOVIEMBRE 2010

Resumen

Twitter es un servicio que últimamente ha adquirido bastante notoriedad en el cual sus usuarios envían y leen entradas de texto de un largo máximo de 140 caracteres. En estos mensajes los usuarios hablan de sus actividades diarias y comparten opiniones e información. Se puede acceder a Twitter desde diversas plataformas como aplicaciones de escritorio, aplicaciones móviles y en la web. Sumado a esto, Twitter cuenta con una API abierta con la cual desarrolladores pueden acceder a su contenido y desarrollar aplicaciones. Esto presenta una oportunidad interesante para efectuar tareas de recuperación de la información y de procesamiento de lenguaje natural.

Esta memoria tuvo como objetivo aplicar una metodología para identificar y clasificar términos indicativos de intereses de los usuarios. El conocer estos intereses resulta útil para tareas de filtrado de información como lo es la generación de recomendaciones. Para identificar y clasificar los términos se hizo uso de técnicas estadísticas y resultados de búsquedas en el motor de búsqueda Google.

Los resultados obtenidos muestran que es posible identificar términos claves indicativos de intereses, y que estas técnicas se pueden refinar para obtener resultados más precisos.

"Hay que tener la mente abierta. Pero no tanto como para que se te caiga el cerebro."

Richard Feynman

Agradecimientos

Antes que nadie quisiera agradecer a mi familia, en especial a mis padres Mario y Magdalena, por todo su apoyo durante mi etapa universitaria. También quisiera agradecer al Profesor Pino, mi profesor guía, por apoyarme en mi tema de memoria poco convencional y a Silvio García y Alan García de Nuwit, por darme la idea que terminó desarrollada como tema en este trabajo. El apoyo de Gastón L'Huillier para este trabajo fue fundamental por lo cual también estoy muy agradecido.

Quisiera ocupar este espacio también para agradecer a un montón de amigos que me han acompañado en mi época universitaria. A mis amigos hechos en el Programa de Bachillerato, en la Escuela de Injenieria y en el DCC. A mis compañeros en todos mis diversos proyectos musicales (Hentai Warriors, Amperes, Zamunda, Chup de Pot, Ramonaldos, Anónimo y varios más) con los cuales mantenemos viva la música. A mis amigos del grupo Nuestra U, orgullosos Hijos de Bello y amantes del Bulla.

Agradezco también a mis héroes del Rock & Roll como Bon Scott y del fútbol y la Universidad de Chile como Marcelo Salas por la inspiración. Especial agradecimiento va para Brian May, quien demostró que se puede ser un *rockstar* y científico a la vez.

Por último, agradezco a todos mis amigos de la vida (vivos o muertos, en Chile o en el extranjero) que de una u otra forma estuvieron presentes durante mi etapa universitaria. Ellos saben quienes son.

Índice General

Índice de tablas

Índice de figuras

Parte I

Introducción y objetivos

Capítulo 1

Introducción

En su libro *The Long Tail: Why the Future of Business Is Selling Less of More* [1], Chris Anderson, editor en jefe de la revista Wired[1], argumenta que estamos migrando de un mundo físico a un mundo digital. Anteriormente, un producto físico como un libro o un CD de música, debía cumplir una demanda suficiente para justificar su lugar en el estante de una librería o en una tienda de discos, debido a los costos que involucra su almacenamiento y distribución. Sin embargo hoy en día, el almacenamiento y distribución digital disminuye drásticamente estos costos. En el mundo físico, la problemática generada por estas restricciones se abordó promoviendo los *best-sellers* o "éxitos de venta", aquellos productos que garantizan al distribuidor una masa crítica de clientes, o sea, vender mucho de unos pocos. Una parte importante de las suposiciones hechas acerca del gusto popular son consecuencia de una deficiente cruza entre oferta y demanda, respuesta del mercado a distribución ineficiente. Con Internet y la distribución en línea, se da la oportunidad de tener una mayor variedad y demostrar que lo que más vende y genera ganancias es lo que Anderson denominó la Larga Cola (*Long Tail*) [1], una gran cantidad de productos distintos que venden poco individualmente, y que antes no eran negocio, pero que sumados entregan el mayor beneficio al distribuidor.

Para que funcione lo que propone Anderson se debe producir una conexión entre la demanda y la ahora enorme oferta que se tiene. Un ejemplo de esto son los Sistemas de Recomendaciones, siendo quizás el más conocido el de la tienda Amazon[2]. Amazon recomienda a sus usuarios productos en su catálogo que les podrían ser de interés en base a sus compras anteriores, las

[1] `http://www.wired.com`
[2] `http://www.amazon.com`

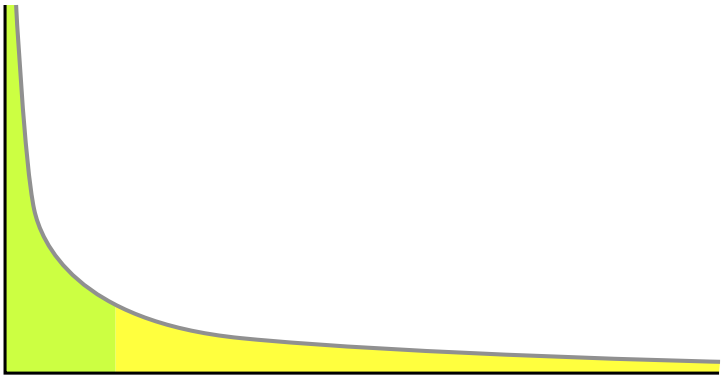

Figura 1.1: La larga cola, en amarillo, puede comprender un área incluso mayor que la de la primera parte (los *best-sellers*).

calificaciones que le han otorgado a los productos otros usuarios con "gustos" similares como también su historial de navegación en la tienda. Los Sistemas de Recomendaciones resuelven un problema importante para el negocio y su área de marketing, determinar que venderle a un cliente dado un perfil de este. Estos sistemas hacen uso de la llamada "Inteligencia Colectiva" que para este ámbito se puede definir de la siguiente forma [2]:

- La inteligencia que es extraída del conjunto de interacciones y contribuciones hechas por los usuarios.
- El uso de esta inteligencia para filtrar lo que es valioso en la aplicación para el usuario.

Originalmente el título de este trabajo iba a ser "Desarrollo de un Sistema adaptable de Recomendación de Productos para plataformas de e-commerce y m-commerce", donde el objetivo sería implementar las técnicas tradicionales que usan los Sistemas de Recomendaciones a una de las aplicaciones de e-commerce de la empresa Nuwit[3]. En particular, el objetivo era implementar el método de filtrado colaborativo (*collaborative filtering*) para un proyecto de e-commerce de Nuwit que se encuentra en producción. Sin embargo, el filtrado colaborativo requiere de, como bien dice su nombre, de la *colaboración* entre los distintos usuarios del e-commerce, en donde calificaran productos para así ir armando un perfil con sus gustos, y esto no se dió en la práctica. Esto generó un problema de "comienzo en frío" o *cold start*, en donde no se tienen suficientes datos para poder comenzar a aplicar el método (o al menos

[3] `http://www.nuwit.com`

aplicarlo de manera efectiva). Por lo tanto el enfoque se modificó del filtrado colaborativo a obtener información de los usuarios desde otras fuentes en la Web, más específicamente, los servicios de redes sociales.

Si bien Twitter[4] por definición es un servicio de *microblogging*, tiene características propias de los servicios de redes sociales como el popular Facebook[5]. Al igual que Facebook, posee una API abierta a cualquier desarrollador que quiera conectar su aplicación o sitio web con el servicio. En Twitter, los usuarios escriben mensajes cortos de 140 caracteres, y el servicio ha adquirido una enorme popularidad en Chile y en buena parte del planeta (en el capítulo 4 se explicarán en detalle todos estos puntos). El hecho de poder acceder a su conjunto de datos mediante la API brinda la posibilidad de aplicar técnicas de recuperación de la información (IR) y procesamiento de lenguaje natural (NLP) y hacer minería de texto. En este trabajo se busca usar Twitter para encontrar intereses de los usuarios. Con esto, un portal de e-commerce, mediante una integración con Twitter, podría conocer los intereses de sus potenciales clientes antes de que hagan una compra y orientar de mejor manera sus recomendaciones de productos, y esta no es la única de sus posibles aplicaciones.

[4] `http://twitter.com`
[5] `http://www.facebook.com`

Capítulo 2

Objetivos

2.1. Objetivo general

El objetivo de este trabajo es aplicar una metodología para identificar dentro de los mensajes enviados por un usuario del servicio Twitter, términos o palabras que pudiesen ser clasificadas y que pudiesen indicar temáticas que son de interés para dicho usuario.

2.2. Objetivos específicos

Los objetivos específicos de este trabajo son los siguientes:

1. Utilizar la API de Twitter para armar un conjunto de datos con mensajes publicados en este servicio por un conjunto de usuarios que han hecho públicos sus mensajes.

2. Procesar este conjunto de datos y filtrar aquellas palabras, términos o unidades léxicas que carecen de contenido semántico usando de técnicas de IR.

3. Identificar los términos con mayor contenido semántico en el conjunto de datos procesado usando técnicas estadísticas.

4. Medir la semejanza semántica entre los términos encontrados en el punto anterior y un listado de categorías representativas de intereses usando un motor de búsqueda, en este caso Google.

5. En base a lo anterior, categorizar estos términos en estas distintas categorías y de tal forma, identificar cuales categorías son más representativas de los intereses de un usuario en particular.

2.3. Estructura del informe

La estructura de este informe se puede resumir de la siguiente forma:

- En el capítulo 1 se introdujo el problema y la motivación de este trabajo y las aplicaciones prácticas que tendría su integración en sistemas reales.
- El el capítulo 3 se revisa la bibliografía que sirvió como referencia para este trabajo.
- En el capítulo 4 se describe una breve introducción a Twitter, explicando su dinámica y definiendo términos propios a él y que serán usados a lo largo de este informe.
- En los capítulos 5 a 8 se describe el desarrollo realizado para cumplir los objetivos enumerados en la sección 2.2.
- En los capítulos 9 y 10 se produce el análisis y discusión de los resultados experimentales para concluir en la sección 10.2.

Capítulo 3

Revisión bibliográfica

En este capítulo se revisará la investigación realizada por otros y que provee el marco teórico para este trabajo.

3.1. Intención y uso en el microblogging

El llamado *microblogging* es una práctica de comunicación relativamente nueva en donde las personas comparten información mediante mensajes cortos y en tiempo real [3]. El hecho de que el acto de microblogging se pueda efectuar mediante plataformas móviles (como smartphones) ha permitido de que la presencia en línea de una persona se vuelva efímera y de naturaleza dinámica como lo son sus pensamientos e intereses [4]. El contenido de estos mensajes va desde compartir actividades diarias con amigos, familiares o colegas, compartir información y noticias o compartir opiniones con observadores interesados [3]. También se ha observado de que se usa (en especial Twitter) para buscar en los mensajes públicos conocimiento y experiencia en distintos temas [5] [6].

En el caso específico de Twitter, Java et al. [6] identificaron las principales intenciones de los usuarios de Twitter (que se expondrán en el siguiente capítulo) identificando además distintas categorías de usuarios en base a una estructura de enlaces que forman segun la relación de "seguimiento" (*following*) propia de este servicio. En esta misma estructura, Kwak et al. [7] encontraron que las características de red social de Twitter difieren de las características conocidas de las redes sociales humanas [8]. En un estudio similar, Weng et al. [9] estudiaron el comportamiento de los usuarios mediante la relación de *following* y como se identifican a usuarios con mayor influencia que otros. Mendoza et al. [10] y Vieweg et al. [11] estudiaron

la diseminación de información a través de Twitter, específicamente en casos de emergencia, como por ejemplo el terremoto del 27 de febrero del 2010 en Chile [10]. Estos estudios mostraron que rumores falsos tienden a ser cuestionados y que en situaciones de crisis el vocabulario usado por los usuarios muestra poca varianza.

La investigación de Banerjee et al. [4] fue la que resultó ser de mayor influencia para este trabajo. En ella se investigó Twitter como una potencial fuente de información contextual acerca de sus usuarios, y se encontró que existe una cantidad suficiente de palabras clave indicativas que manifiestan intereses de los usuarios y que en estas se pueden encontrar asociaciones y aplicarles técnicas de clasificación.

3.2. Medición de semejanza entre palabras usando motores de búsqueda

La relación semántica entre palabras es importante para muchas tareas de NLP, y una buena parte del NLP es encontrar la propensión a que las palabras aparezcan juntas en textos, conocida como su semejanza de distribución. En el idioma inglés, comúnmente se usa la base de datos léxica WordNet[1] para medir semejanza entre palabras [12] [13] [14]. Sin embargo, el problema que presenta este método es que está confinado a medir la semejanza entre palabras según las conexiones léxicas que han sido manualmente asignadas a ellas [15] además que para el caso particular de este trabajo, en donde se trabajará con texto en idioma castellano, no existe un equivalente a WordNet en este idioma. En vez de usar un cuerpo lingüistico "clásico" como WordNet o el British National Corpus (BNC)[2] (ambos para el idioma inglés), se propone usar la Web donde el problema de cobertura es menos probable. Nuevas palabras se crean constantemente y nuevos significados se le están dando a palabras ya existentes [16]. La búsqueda en la Web se considera una parte importante de medir semejanza, ya que proporciona información actualizada con respecto a frecuencias de co-ocurrencia de palabras en la colección más grande de documentos en varios idiomas [15].

[1] http://wordnet.princeton.edu/
[2] http://www.natcorp.ox.ac.uk/

Diversos métodos se han propuesto para medir la semejanza semántica. Sahami et al. [17] proponen una función kernel usando los resultados de un motor de búsqueda para medir la semejanza semántica entre pequeños fragmentos de texto. El uso de grafos para hacer clustering de palabras en base a la cantidad de *page counts* fue el método empleado por Matsuo et al. [18], en donde las palabras conforman nodos y la arista entre dos nodos está dada por la cantidad de resultados de buscar en un motor de búsqueda las palabras que conforman esos nodos. Bollegala et al. [16] integran a los resultados de búsqueda la extracción de patrones léxico-sintácticos de fragmentos de texto y Máquinas de Vectores de Soporte (SVMs) para mejorar la precisión de las medidas de semejanza entre palabras.

Capítulo 4

Breve introducción a Twitter

En este capítulo se detallará la forma de usar Twitter y la dinámica que se produce al usar este servicio. Además, se revisarán los distintos usos que la gente le ha dado a este servicio.

4.1. Definición del servicio

Antes de describir su uso, se debe primero establecer que Twitter es un servicio de *micro-blogging*. Para entender el concepto de microblogging se deben definir los conceptos de *blog*, y de *blogging*.

- Blog: contracción de *"web log"* (diario o bitácora web), es un sitio web periódicamente actualizado que recopila cronológicamente textos o artículos de uno o varios autores, apareciendo primero el más reciente, donde el autor conserva siempre la libertad de dejar publicado lo que crea pertinente. [19]
- Blogging: El verbo blogging hace referencia a mantener o agregar contenido a un blog. [20]

Los blogs generalmente son mantenidos por una persona o un grupo de personas con entradas regulares que contienen comentarios, descripción de eventos u otros contenidos multimedia como imágenes y video. [20] Muchos blogs comentan o comparten noticias con respecto a temas específicos (por ejemplo FayerWayer[1] es un conocido blog de noticias de tecnología en español), mientras que otros abarcan una variedad de temas y otros actúan como diarios personales en línea.

[1] `http://www.fayerwayer.com/`

Wikipedia [21] define el microblogging (también llamado *nanoblogging*) como un medio de difusión pasivo en forma de blogging, pero que se diferencia de los blogs tradicionales en que su contenido es mucho más reducido. En el caso de Twitter, las entradas se limitan a 140 caracteres, límite impuesto para ser compatible con servicios de mensajerá de texto o SMS. [22] [23] El medio del microblogging se ubica entre los mensajes de texto, los mensajes de estado de los servicios de mensajería instantánea (IM) como el de Windows Live Messenger[2], los blogs y los sitios de redes sociales (como Facebook). [3]

4.2. Particularidades de Twitter

La interfaz web de Twitter le presenta al usuario la pregunta *"?'Qué pasa?"* (*"What's happening?"* en su idioma inglés original) junto a un campo de texto donde puede escribir un mensaje de actualización de estado (*status update*) de no más de 140 caracteres (mensajes más largos que el límite se verán truncados. Twitter maneja una serie de convenciones y vocabulario particular [24] (buena parte creado por sus propios usuarios[3]) para referirse a distintos elementos de su interacción:

- Tweet: nombre que se le da a un mensaje publicado via Twitter.
- Follower: un *follower* (que se traduce como "seguidor") es un usuario de Twitter que se ha suscrito a los Tweets y actualizaciones de otro usuario en particular. Un follower no necesita el permiso del usuario al cual quiere seguir (*follow*) para poder seguirlo.
- Retweet: un tweet de otro usuario que ha sido reenviado por alguien al cual se sigue (*follow*). Se abrevia RT.
- Hashtag: El símbolo # es usado para marcar palabras claves o temas en un Tweet. Estas palabras marcadas son denominadas *hashtags*.
- Trending Topic: Un tema que se determina algorítmicamente a ser de los más populares en Twitter en un determinado momento. Se abrevia TT.

[2] `http://windowslive.com/desktop/messenger`
[3] http://www.fayerwayer.com/2010/07/jack-en-chile-en-entel-summit-en-vivo/

- @: La arroba es usada para hacer referencia a nombres de usuario dentro de los Tweets. Cuando el nombre de usuario es precedido por @, se convierte en un enlace al perfil del usuario.

- Timeline: se traduce como "línea del tiempo", y es una lista en tiempo real de los Tweets en Twitter. Para un usuario cualquiera, en su *timeline* apareceran los Tweets de los usuarios a los cual sigue.

La figura 4.1 muestra un Tweet del usuario @twitter de en donde se aprecian varios de los elementos antes definidos:

Figura 4.1: Retweet hecho por el usuario @twitter de un tweet originalmente publicado por @USATODAY en donde se hace referencia al hashtag #tech y también al usuario @Twitter (en Twitter, los nombres de usuario no distinguen entre mayúsculas y minúsculas por lo tanto @twitter y @Twitter corresponden a la misma cuenta).

En la figuras 4.2 y 4.3 se puede apreciar el Timeline del autor de este trabajo (@MarcAstr0) tanto en su interfaz web como en su interfaz de la aplicación para el iPhone de Apple. Una de las fortalezas de Twitter es que permite la publicación de tweets desde diversos clientes aparte de su interfaz web, como correo electrónico, mensajes de texto SMS, aplicaciones para diversos teléfonos celulares de última generación (smartphones), aplicaciones de escritorio y también diversos servicios basados en la web. Esto se debe a que Twitter cuenta con una API [25] completamente documentada [26] para que desarrolladores puedan desarrollar aplicaciones que se integren con Twitter (en capítulos posteriores se repasarán las partes de la API que son relevantes para este trabajo).

4.3. Uso del servicio

El hecho de limitar el tamaño de las entradas a mensajes cortos (en el caso de Twitter a 140 caracteres) ha convertido al microblogging en una forma de comunicación sencilla y liviana que

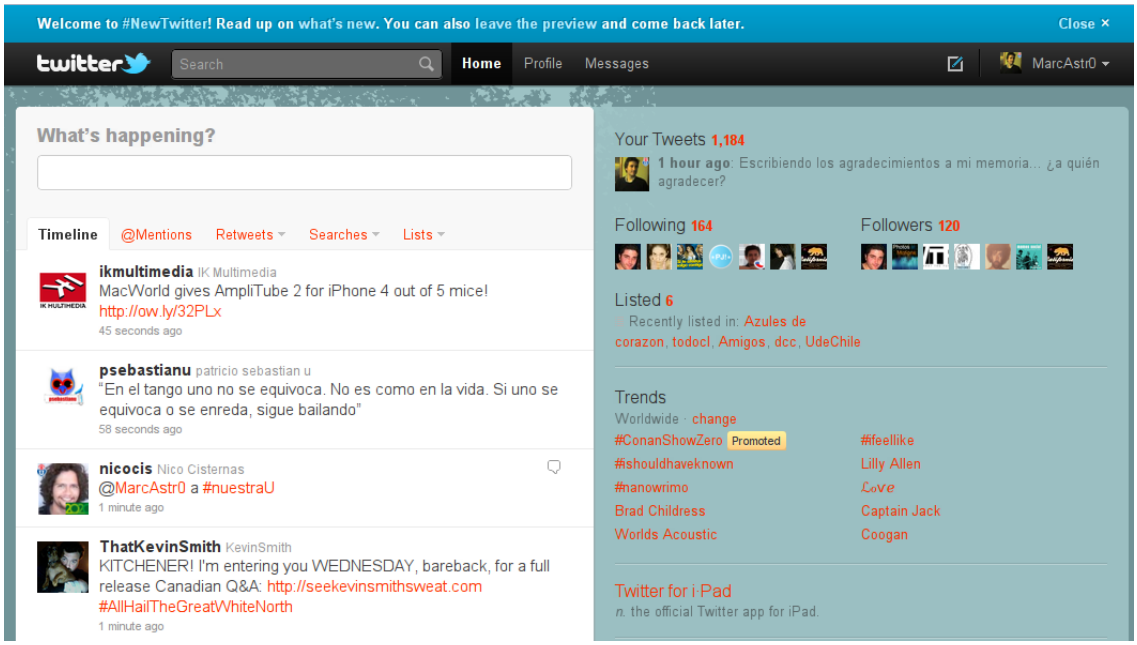

Figura 4.2: Timeline del usuario @MarcAstr0 visto desde la interfaz web.

Figura 4.3: Timeline del usuario @MarcAstr0 visto desde la aplicación para iPhone.

permite a los usuarios transmitir y compartir información acerca de sus actividades, opiniones y estado actual. [6] Comparado con el blogging, el microblogging satisface la necesidad de una forma de comunicación rápida y en la cual el requisito de tiempo y pensamiento para generar

el contenido se ve disminuido. Esto permite también una mayor frecuencia de actualización. En promedio, los blogueros actualizan sus blogs después de pocos días, mientras que un microbloguero puede publicar varias actualizaciones en un solo día. [6] Esto ha llevado a que Twitter sea principalmente usado con las siguientes intenciones [6]:

- Charla diaria: tweets en los cuales los usuarios describen que están haciendo en ese instante.
- Conversaciones: tweets que referencian a otros usuarios (anteponiendo @ al nombre del usuario referenciado) para responder a los tweets de esos usuarios.
- Compartir información/URLs
- Reportar noticias

Twitter además provee funcionalidades características de los servicios de redes sociales (como Facebook o MySpace[4]) con el modelo de seguidores o *followers* definido anteriormente. La cantidad de followers es un buen indicativo de la popularidad de un usuario de Twitter.

[4] `http://www.myspace.com`

Parte II

Investigación y desarrollo

Capítulo 5

Recolección de tweets y conjunto de datos

En este captúlo se describe la metodología utilizada para obtener el conjunto de datos usados en esta investigación que sumó un total de 105737 tweets obtenidos a partir de 44 de usuarios de Twitter. Para recolectar los tweets se hizo uso de la API que Twitter provee, además de una biblioteca especializada para analizar sintácticamente (*parse*) XML.

5.1. La API de Twitter

Twitter provee de una API para poder desarrollar aplicaciones que interactúan con Twitter. Para acceder a la API de Twitter se hacen peticiones (*requests*) por HTTP y la API devuelve datos en forma estructurada para facilitar su análisis sintáctico. Esta se encuentra documentada en el portal para desarrolladores y su wiki. La API consiste en tres partes:

- REST API: los métodos de esta API permiten acceder a la esencia de los datos.
- Search API: como su nombre lo indica, se usa para la búsqueda.
- Streaming API: para acceder a los datos casi en tiempo real.

Para este trabajo solo interesan los métodos de la primera parte (REST API), debido a que solo se quiere recolectar los tweets para un conjunto de usuarios en particular, indicando el nombre de usuario de estos. Como se mencionó anteriormente, los métodos de la API se llaman mediante peticiones HTTP, que pueden ser de tipo GET, POST o DELETE. Mediante GET, se pueden acceder a métodos que no efectúan cambios en los servidores de Twitter

(como obtener el *timeline* de un usuario), para métodos que sí efectúan cambios, se usa POST o DELETE en algunos casos. Los datos que devuelven los métodos pueden venir estructurados con los siguientes formatos:

- XML
- JSON
- RSS
- Atom

El formato XML fue el escogido para recolectar los datos debido a que existe la biblioteca Beautiful Soup para el análisis sintáctico de documentos en XML.

5.1.1. Obtención tweets para un único usuario

Para obtener tweets de un solo usuario se puede llamar al siguiente método de la API:

```
https://twitter.com/statuses/user_timeline/id.xml
```

en donde `id` es reemplazado por el nombre de usuario o el número identificador de este. Por ejemplo, para el usuario @MarcAstr0 sería:

```
https://twitter.com/statuses/user_timeline/MarcAstr0.xml
```

Este método devuelve los últimos 20 tweets en el *timeline* del usuario en formato XML, si se quisiera obtenerlos en formato JSON, solo basta con cambiar la extensión e invocar la siguiente URL:

```
https://twitter.com/statuses/user_timeline/id.json
```

El timeline se representa como un arreglo de objetos en JSON. En XML, cada tweet se encuentra delimitado dentro de una etiqueta `<status>`, y el timeline se encuentra dentro de la etiqueta `<statuses>`. A continuación se pueden apreciar las diferencias entre los formatos para un mismo tweet:

JSON


```
{
  "geo":null,
  "in_reply_to_screen_name":null,
  "in_reply_to_user_id":null,
  "in_reply_to_status_id":null,
  "retweeted":false,
  "truncated":false,
  "entities":{
         "hashtags":[

         ],
         "urls":[

         ],
         "user_mentions":[

         ]
  },
  "place":null,
  "source":"<a href=\"http://twitter.com/\" rel=\"nofollow\">Twitter for iPhone</a>",
  "retweet_count":0,
  "favorited":false,
  "user":{
         "listed_count":7,
         "favourites_count":0,
         "profile_link_color":"FF3300",
         "description":"",
         "contributors_enabled":false,
         "following":true,
         "geo_enabled":true,
         "notifications":false,
         "profile_sidebar_fill_color":"A0C5C7",
         "time_zone":"Santiago",
         "profile_image_url":"http://a1.twimg.com/profile_images/579069573/
           twitterProfilePhoto_normal.jpg",
         "verified":false,
         "profile_sidebar_border_color":"86A4A6",
         "follow_request_sent":false,
         "screen_name":"MarcAstr0",
         "profile_use_background_image":true,
         "profile_background_color":"709397",
         "followers_count":107,
         "protected":false,
         "profile_background_image_url":"http://s.twimg.com/a/1284676327/
           images/themes/theme6/bg.gif",
         "url":null,
         "name":"Mario Castro",
         "show_all_inline_media":false,
         "friends_count":142,
         "profile_text_color":"333333",
         "id":79473145,
         "lang":"en",
         "statuses_count":888,
```

```
        "profile_background_tile":false,
        "utc_offset":-14400,
        "created_at":"Sat Oct 03 14:28:43 +0000 2009",
        "location":"Santiago, Chile"
  },
  "contributors":null,
  "id":25315850908,
  "coordinates":null,
  "text":"Recien me jugue un raspe y gane $100. Debe ser mi dia
    de suerte.",
  "created_at":"Thu Sep 23 15:15:01 +0000 2010"
}
```

XML

```
<status>
  <created_at>Thu Sep 23 15:15:01 +0000 2010</created_at>
  <id>25315850908</id>
  <text>Recien me jugue un raspe y gane $100. Debe ser mi dia de suerte.</text>
  <source>&lt;a href="http://twitter.com/" rel="nofollow"&gt;Twitter for
    iPhone&lt;/a&gt;</source>
  <truncated>false</truncated>
  <in_reply_to_status_id/>
  <in_reply_to_user_id/>
  <favorited>false</favorited>
  <in_reply_to_screen_name/>
  <retweet_count/>
  <retweeted>false</retweeted>
  <user>
    <id>79473145</id>
    <name>Mario Castro</name>
    <screen_name>MarcAstr0</screen_name>
    <location>Santiago, Chile</location>
    <description/>
    <profile_image_url>http://a1.twimg.com/profile_images/579069573/
          twitterProfilePhoto_normal.jpg</profile_image_url>
    <url/>
    <protected>false</protected>
    <followers_count>107</followers_count>
    <profile_background_color>709397</profile_background_color>
    <profile_text_color>333333</profile_text_color>
    <profile_link_color>FF3300</profile_link_color>
    <profile_sidebar_fill_color>A0C5C7</profile_sidebar_fill_color>
    <profile_sidebar_border_color>86A4A6</profile_sidebar_border_color>
    <friends_count>142</friends_count>
    <created_at>Sat Oct 03 14:28:43 +0000 2009</created_at>
    <favourites_count>0</favourites_count>
    <utc_offset>-14400</utc_offset>
    <time_zone>Santiago</time_zone>
    <profile_background_image_url>http://s.twimg.com/a/1284676327/
          images/themes/theme6/bg.gif</profile_background_image_url>
    <profile_background_tile>false</profile_background_tile>
    <profile_use_background_image>true</profile_use_background_image>
    <notifications>false</notifications>
    <geo_enabled>true</geo_enabled>
```

```
    <verified>false</verified>
    <following>false</following>
    <statuses_count>888</statuses_count>
    <lang>en</lang>
    <contributors_enabled>false</contributors_enabled>
    <follow_request_sent>false</follow_request_sent>
    <listed_count>7</listed_count>
    <show_all_inline_media>false</show_all_inline_media>
  </user>
  <geo/>
  <coordinates/>
  <place/>
  <contributors/>
</status>
```

De toda la información que se devuelve para cada tweet, para este trabajo solo será relevante lo que aparece entre los tags `<text>`, que corresponde al texto del tweet:

```
<text>Recien me jugue un raspe y gane $100. Debe ser mi dia de suerte.</text>
```

5.1.2. Obtención de más tweets y rate limiting

El método recién expuesto muestra los últimos 20 tweets en el *timeline* del usuario, cantidad insuficiente para el análisis que se desea hacer en este estudio. A traves de la glsapi es posible acceder a los últimos 3200 tweets (en caso de tener igual o mayor cantidad de tweets). Para ello se usa el mismo método anterior, pero se le entrega una variable adicional en la URL de la siguiente forma:

```
https://twitter.com/statuses/user_timeline/id.xml?page=x
```

Donde `x` es reemplazado por el número de página. Si el usuario tiene más de 3200 tweets, llegará solamente hasta la página 160. Invocando al mismo método, se puede saber la cantidad total de tweets del usuario que está dada por el tag `<statuses_count>` y dividiendo por 20 se puede obtener la cantidad de páginas para poder iterar.

Cuando Twitter adquirió suficiente popularidad la sobrecarga de llamadas a métodos de la API producidos en sus servidores obligó a instaurar una política que limita la cantidad de llamados a la API, conocida como *rate limiting*. Actualmente, para la REST y Search API, se permite invocar un máximo de 150 métodos por hora. Si se desea obtener los tweets para un

usuario con 3200 tweets, donde se debe invocar al método anterior 160 veces (una vez cada página) se producirá un error HTTP de tipo 400 (*Bad Request*) antes de obtener todos los tweets. Esta limitante está considerada en la implementación que se describirá más adelante.

5.2. Análisis sintáctico usando Beautiful Soup

Para procesar los XML obtenidos a través de la invocación del método descrito anteriormente, se usó la biblioteca Beautiful Soup [27]. Beautiful Soup, es un *parser* o analizador sintáctico de HTML/XML para el lenguaje de programación Python con el cual, en conjunto de la biblioteca `urllib` de manera fácil se pueden obtener los elementos y navegar el árbol de análisis sintáctico o *parse tree*. Un ejemplo sencillo de uso a través de la consola de Python se puede ver a continuación [28]:

```
>>> from BeautifulSoup import BeautifulSoup
>>> from urllib import urlopen
>>> soup=BeautifulSoup(urlopen('http://google.com'))
>>> soup.head.title
<title>Google</title>
>>> links=soup('a')
>>> len(links)
21
>>> links[0]
<a href="http://www.google.com/ig?hl=en">iGoogle</a>
>>> links[0].contents[0]
u'iGoogle'
```

5.3. Implementación del recolector de tweets

Para obtener el conjunto de datos, se desarrolló la aplicación `fetchtweets.py` que a partir de un archivo de texto con una lista de usuarios de Twitter, escribiera un archivo de texto por cada usuario en esa lista con la mayor cantidad de tweets que se pudieran obtener. El código fuente de `fetchtweets.py` se encuentra en el anexo A

Capítulo 6

Preprocesamiento del texto

En el capítulo anterior se describió la forma en que se obtuvo un conjunto de datos comprendido por una cantidad considerable de tweets de distintos usuarios. En este capítulo se describen las transformaciones que se le hacen a los tweets, para solo dejar aquellas palabras que tengan más significado, o para decirlo de otra forma, que describan mejor el contenido del tweet.

En general, las palabras sustantivas (o grupos de palabras sustantivas) son las más representativas del contenido de un documento, más que adjetivos, adverbios y verbos [29]. Las palabras que son muy frecuentes dentro de un documento como artículos, preposiciones y conjunciones que por si solas no llevan ningún significado son conocidas como palabras vacías o *stopwords*. Por eso, para analizar los tweets antes es necesario filtrar este conjunto de palabras.

Además de las palabras vacías, Twitter cuenta con palabras especiales y específicas a este servicio, que son las etiquetas o hashtags, y los nombres de usuario, como también abreviaciones propias a Twitter como RT (*retweet*), TL (*timeline*), TT (*trending topic*). Si bien algunas de estas palabras (como los hashtags) pueden tener un contenido semántico, por razones que se mencionarán más adelante serán filtradas antes del análisis de los tweets.

Para encontrar y filtrar todas estas palabras se hará uso extensivo de expresiones regulares.

6.1. Filtrado de URLs

Una cantidad significativa de tweets contienen URLs. En el conjunto de datos de este estudio se encontró que aproximadamente un 21 % de tweets contiene URLs. Con ellas los usuarios comparten información contenida en enlaces externos. Debido al límite de 140 caracteres por tweet, se ha vuelto frecuente el uso de servicios que acortan URLs como TinyURL[1] o bit.ly[2]. Las URLs de por si no llevan ningún contenido semántico, ya que el contenido se encuentra en el documento al cual enlazan (de hecho, pueden enlazar a otros tweets). Para ver el aporte semántico de las URL sería necesario analizar el contenido del enlace (a traves de un *webbot* o *crawler*), que puede ser páginas web con distintas estructuras, contenido multimedia como imágenes o video lo cual conlleva un análisis distinto al objetivo de este estudio. Es por esta razón que se decidió excluir las URLs. La expresión regular usada para encontrar las URLs es la descrita a continuación:

```
http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+
```

6.2. Filtrado de hashtags

Las etiquetas o hashtags fueron un invento de la propia comunidad de Twitter para poder agrupar tweets. Estas etiquetas pueden ser letras o números pero siempre precedidos del caracter '#'. Por ejemplo, muchos usuarios de Twitter dan a conocer la música que están escuchando mediante el hashtag #nowplaying. Otro ejemplo digno de destacar del uso de hashtags es su uso durante emergencias, como sucedió después del terremoto del 27 de febrero del 2010 en Chile, donde el hashtag #terremotochile era usado para etiquetar tweets con noticias relativas al terremoto [10]. La interfaz web de Twitter así como también las distintas aplicaciones para dispositivos permiten hacer búsquedas por hashtags.

Si bien en los hashtags se puede encontrar contenido semántico, se ha decidido filtrarlos debido a las limitaciones de su construcción. Los hashtags como se mencionó anteriormente se componen por letras o números precedidos de '#'y no pueden contener caracteres como signos

[1] http://tinyurl.com
[2] http://bit.ly

de puntuación o un espacio, y generalmente los hashtags son palabras concatenadas como por ejemplo #terremotochile, concatenación de "terremoto" y "chile". Este ejemplo corresponde al bigrama "terremoto chile". El análisis semántico del hashtag implicaría descomponerlo en el bigrama (o *n*-grama si corresponde) correspondiente, pero la ausencia de espacios o signos de puntuación complica la automatización de este proceso (a pesar de que se acepta el formato CamelCase para separar las palabras, la mayoría de las veces los hashtags están todos en letras minúsculas).

Para encontrar los hashtags se usó la siguiente expresión regular:

```
(\A|\s|\w|\W)#(\w+)
```

6.3. Filtrado de nombres de usuario

Los nombres de usuario en Twitter se identifican con el caracter '@' y se usan para dirigir los tweets a otros usuarios, con los cuales estos pueden llevar a cabo verdaderas "conversaciones". Por si solos no entregan significado por lo tanto se filtran. La siguiente expresión regular encuentra una cadena correspondiente a un nombre de usuario:

```
(\A|\s|\w|\W)@(\w+)
```

6.4. Filtrado de palabras vacías o stopwords

Como se mencionó anteriormente, las palabras vacías o stopwords son palabras que por si solas no llevan ningún significado como lo son los artículos, pronombres, preposiciones, conjunciones, como también algunos verbos, adverbios y adjetivos. La lista completa de palabras vacías filtradas se encuentra en el anexo F y el código de la aplicación `removestopwords.py` encargada de filtrar el texto se encuentra en el anexo B .

Capítulo 7

Procesamiento del texto

Una vez preprocesado el texto eliminando palabras vacías (palabras que por si solas no tienen significado) y otras palabras específicas a Twitter que no aportan contenido semántico (hashtags, nombres de usuario, abreviaciones) se procesa el texto para encontrar las palabras y frases que se usarán como palabras clave o keywords al momento de medir su similitud usando Google. Para esta parte se hizo uso del Ngram Statistics Package (NSP)[1], herramienta desarrollada en Perl que identifica *n*-gramas, secuencias de *n* unidades léxicas (*tokens*).

7.1. Identificando las palabras más usadas

Usando NSP es sencillo conseguir un listado de las palabras en orden descendente por frecuencia de aparición en el texto. Solo basta considerar que se buscan *n*-gramas con $n = 1$. Esto se efectúa usando la instrucción `count.pl` de la siguiente forma:

```
> count.pl --ngram 1 --newLine --token tokens.txt output.cnt input.txt
```

La opción `--ngram 1` indica que se buscan *n*-gramas con $n = 1$ (i.e. palabras sueltas). La opción `--newLine` le dice a NSP que se desea que los *n*-gramas no abarquen más de 1 línea (en este caso esta opción es innecesaria). Con la opción `--token tokens.txt` se le dice a NSP que identifique *n*-gramas que cumplan con la expresión regular indicada en el archivo `tokens.txt`. Por defecto, NSP viene pensado para las siguientes dos expresiones regulares:

```
\w+
[\.,;:\?!]
```

[1] `http://www.d.umn.edu/~tpederse/nsp.html`

Estas dos expresiones regulares coinciden con secuencia de caracteres alfanuméricos ASCII y signos de puntuación, sin embargo, los tweets a analizar son de usuarios chilenos cuya mayor cantidad de tweets están en idioma castellano, que contiene caracteres que no se ven incluidos por aquellas expresiones regulares mencionadas anteriormente, como lo son la letra 'ñ'y las vocales con tilde. Por eso, el archivo `tokens.txt` contiene una expresión regular que incluye estos caracteres y otros más que después de una inspección manual de los tweets se vieron incluídos, como las apóstrofes, *slash* y *backslash*.

Los archivos `output.cnt` y `input.txt` corresponden a los archivos de salida y entrada respectivamente. En la primera línea de `output.cnt` aparece la cantidad total (tamaño de la muestra) de unigramas (n-gramas con $n = 1$) y en las líneas siguientes se listan los unigramas con sus frecuencias. A modo de ejemplo, la tabla 7.1 lista los 10 unigramas más frecuentes del usuario @PrensaFutbol:

unigrama	**Frecuencia**
Colo	359
Chile	306
U	290
partido	199
Mundial	155
final	152
Copa	148
España	130
San	120
Católica	116

Tabla 7.1: Los 10 unigramas más frecuentes para el usuario @PrensaFutbol

En este ejemplo, sabiendo de antemano que @PrensaFutbol es la cuenta de Twitter del conocido blog de fútbol Prensafútbol[2] es fácil notar que 'Mundial'hace referencia al Campeonato Mundial de Fútbol de la FIFA y 'Católica'se refiere al equipo de fútbol de la Universidad Católica. Sin embargo, estas palabras por si solas ofrecen un significado ambiguo si no se sabe que @PrensaFutbol se dedica exclusivamente a discutir fútbol. ¿"Católica" por "Iglesia Católi-

[2] `http://www.prensafutbol.cl`

ca"? ¿por "Isabel La Católica"? etc. Esta ambigüedad hace necesario un análisis de n-gramas con $n > 1$.

7.2. Identificando bigramas

Se entenderá que un bigrama es un n-grama con $n = 2$. Como en el caso de los unigramas, usando la instrucción `count.pl` se consigue la lista de bigramas ordenados por frecuencia:

```
> count.pl --ngram 2 --newLine --token tokens.txt output.cnt input.txt
```

Por defecto `count.pl` busca bigramas por lo tanto la opción `--ngram 2` puede omitirse.

Usando el mismo ejemplo anterior para el usuario @PrensaFutbol, la tabla 7.2 muestra la lista de sus 10 bigramas más frecuentes:

Bigrama	**Frecuencia**	f_i	f_d
Colo Colo	179	330	338
Real Madrid	66	79	103
Humberto Suazo	43	44	96
Copa Libertadores	36	133	56
Marcelo Bielsa	36	48	83
Copa Chile	33	133	294
San Felipe	31	120	35
Universidad Católica	31	64	109
Jorge Valdivia	30	41	59
Jaime Valdés	29	38	56

Tabla 7.2: Los 10 bigramas más frecuentes para el usuario @PrensaFutbol (tamaño de la muestra: 27182)

En esta tabla se incluyen dos frecuencias adicionales. f_i indica la cantidad de veces que aparece la unidad léxica izquierda en el lado izquierdo de un bigrama y f_d indica la cantidad de veces que aparece la unidad léxica derecha en el lado derecho de un bigrama. Por ejemplo, el bigrama "Real Madrid" aparece 66 veces, la unidad léxica "Real" aparece 79 veces al lado izquierdo, no solo aparece en "Real Madrid" sino que también podría aparecer por ejemplo en "Real Sociedad", "Real Valladolid", etc. Lo mismo con "Madrid" que aparece 103 veces en

un lado derecho, además de "Real Madrid" también podría aparecer en por ejemplo "Atlético Madrid".

Como se puede apreciar en el listado de bigramas, estos son mucho menos ambiguos en su significado que los unigramas, y serán la base para la medición de similitud. De la misma forma se pueden ir encontrando trigramas ($n = 3$), tetragramas ($n = 4$) pero en este caso solo se llegará a $n = 3$. El listado de trigramas para @PrensaFutbol se puede apreciar en la tabla 7.3:

Trigrama	**Frecuencia**	f_1	f_2	f_3	f_{12}	f_{13}	f_{23}
San Pedro Atacama	15	102	30	15	16	15	15
Buenos días amigos	10	22	31	18	21	10	10
Arrancó segunda mitad	10	29	33	22	12	10	15
San Carlos Apoquindo	10	102	57	11	20	10	11
Everton Viña Mar	9	48	18	17	9	9	17
Inter Porto Alegre	8	67	9	8	8	8	8
San Luis Quillota	8	102	46	9	25	8	8
Colo Colo Olimpia	8	294	309	23	150	8	8
final Copa Libertadores	7	95	127	54	15	7	35
partido Colo Colo	7	148	309	302	7	8	158

Tabla 7.3: Los 10 trigramas más frecuentes para el usuario @PrensaFutbol

En este caso las frecuencias f_{ij} corresponden a la cantidad de veces que aparecen las unidades léxicas i y j en las posiciones i y j de otros trigramas.

7.3. Medidas de asociación para *n*-gramas

Una de las características importantes que ofrece NSP es aplicar una serie de medidas de asociación a los datos obtenidos por `count.pl` [30]. Estas medidas permiten juzgar si las unidades léxicas que componen los n-gramas aparecen juntas más de lo que se esperaría a que si hubieran sido puestas al azar (i.e. no son mera coincidencia) [30] [31]. NSP soporta el test exacto de Fisher, el coeficiente de verosimilitud (*log-likelihood ratio*) como también el coeficiente de Sørensen-Dice. Este último no permite asociarle un nivel de significancia a su valor.

7.3.1. Coeficiente de verosimilitud

Considerar un bigrama compuesto por las palabras palabra1 y palabra2. El coeficiente de verosimilitud [32] mide la desviación entre los datos observados y lo que se esperaría si palabra1 y palabra2 fuesen independientes. Mientras más alto este puntaje, menor es la evidencia a favor de que las palabras son independientes. Los datos obtenidos se pueden resumir en la siguiente tabla de contingencia de 2×2:

	palabra2	¬palabra2	totales
palabra1	n_{11}	n_{12}	n_{1p}
¬palabra1	n_{21}	n_{22}	n_{2p}
totales	n_{p1}	n_{p2}	n_{pp}

De la tabla 7.2 se sabe que n_{pp} corresponde al tamaño de la muestra, n_{11} corresponde a la frecuencia del bigrama, n_{p1} corresponde a f_d y n_{1p} a f_i. Los otros valores de la tabla de contingencia se pueden obtener sabiendo que:

$$n_{12} = n_{1p} - n_{11}$$

$$n_{21} = n_{p1} - n_{11}$$

$$n_{p2} = n_{pp} - n_{p1}$$

$$n_{2p} = n_{pp} - n_{1p}$$

$$n_{22} = n_{2p} - n_{21}$$

Los valores esperados para las celdas interiores se calculan como el producto entre sus marginales asociados dividido por el tamaño de la muestra, por ejemplo:

$$m_{11} = \frac{n_{p1} n_{1p}}{n_{pp}}$$

Con esto, para esta tabla de contingencia general se puede calcular el coeficiente de verosimilitud como la siguiente estadística:

$$-2 \log \Lambda = 2 \sum_{i,j} n_{ij} \log \left(\frac{n_{ij}}{m_{ij}} \right)$$

NSP efectúa el cálculo de esta estadística con la instrucción `statistic.pl` de la siguiente forma:

```
> statistic.pl -score 6.00 -frequency 5 ll.pm output.ll input.cnt
```

El archivo de entrada `input.cnt` corresponde al archivo de salida entregado por `count.pl`. Esta instrucción le indica a NSP que entregue la lista de bigramas cuyo coeficiente de verosimilitud sea mayor o igual a 6 y que aparezcan 5 o más veces. La tabla 7.4 lista los 10 primeros bigramas para el usuario @PrensaFutbol rankeados por el coeficiente de verosimilitud. Notar que la mayoría de los bigramas se repiten de la tabla 7.2

Bigrama	**Coef de verosimilitud**
Colo Colo	1234.3322
Real Madrid	721.4402
Humberto Suazo	498.9731
Costa Marfil	388.9583
Marcelo Bielsa	381.5384
Santa Laura	350.4681
Jorge Valdivia	339.0754
Gerardo Pelusso	336.9837
Jaime Valdés	335.7174
Alexis Sánchez	328.9956

Tabla 7.4: Primeros 10 bigramas para el usuario @PrensaFutbol rankeados por el coeficiente de verosimilitud (*log-likelihood ratio*)

7.3.2. Test exacto de Fisher

Considerar la tabla de contingencia presentada anteriormente. R. A. Fisher[3] mostró que la probabilidad de obtener ese conjunto de valores estaba dada por la distribución hipergeométrica:

$$p = \frac{\binom{n_{1p}}{n_{11}}\binom{n_{2p}}{n_{21}}}{\binom{n_{pp}}{n_{p1}}} = \frac{n_{1p}!n_{2p}!n_{p1}!n_{p2}!}{n_{11}!n_{12}!n_{21}!n_{22}!n_{pp}!}$$

[3]Ronald Aylmer Fisher, (1890 – 1962) científico, matemático, estadístico, biólogo evolutivo y genetista inglés.

Los Tests exactos de Fisher [33] se calculan dejando fijos los totales marginales (i.e. n_{1p}, n_{2p}, n_{p1} y n_{p2}) y computando las probabilidades hipergeométricas para las tablas de contingencia posibles. En el caso de un test de cola izquierda, se calculan las probabilidades para las tablas de contingencia con valores menores a n_{11} (con los totales marginales fijos y usando las ecuaciones descritas anteriormente se pueden obtener los valores para n_{12}, n_{21} y n_{22}) y se suman para obtener el valor de significancia. Para un test de cola derecha se usan los valores mayores o iguales a n_{11}. Para determinar dependencia en bigramas, se interpreta el Test exacto de Fisher como un test de cola izquierda, ya que este muestra cuán probable es ver el bigrama observado una menor cantidad de veces en otra muestra aleatoria en una población donde la hipótesis de independencia es cierta. Si esta probabilidad calculada es alta entonces las palabras que forman el bigrama son dependientes [34].

Con NSP el cómputo del Test de Fisher se hace con `statistic.pl`:

```
> statistic.pl -score 0.90 leftFisher.pm output.fish input.cnt
```

Aquí nuevamente `input.cnt` corresponde al archivo entregado por `count.pl`. La opción `-score 0.90` hace que NSP solo escriba en el archivo de salida los bigramas con probabilidades mayores o igual a 0.90. Siguiendo el mismo ejemplo prevalente, en la tabla 7.5 se aprecian los 10 primeros bigramas rankeados por probabilidad dada por el Test exacto de Fisher de cola izquierda para el usuario @PrensaFutbol. En esta tabla todos presentan probabilidad igual a 1 (i.e. son palabras muy dependientes entre ellas). El segundo criterio para su ranking es por frecuencia de aparición del bigrama. En este ejemplo la lista de bigramas de la tabla 7.5 resultó ser idéntica a la de la tabla 7.2.

7.3.3. Coeficiente de Sørensen-Dice

El índice de Sørensen[4], también conocido como el coeficiente de similitud de Sørensen o el coeficiente de Dice[5] (de ahora en adelante referido a él como el coeficiente de Sørensen-Dice) es una estadística usada para medir la similitud entre dos muestras. Para un bigrama, usando la tabla de contingencia definida anteriormente, este coeficiente se define como:

[4]Atribuido a Thorvald Julius Sørensen (1902 – 1973) [35], botánico y biólogo evolutivo danés.
[5]Atribuido a Lee Raymond Dice (1887 – 1977), zoólogo, ecólogo y geneticista estadounidense.

Bigrama	Probabilidad
Colo Colo	1.0000
Real Madrid	1.0000
Humberto Suazo	1.0000
Copa Libertadores	1.0000
Marcelo Bielsa	1.0000
Copa Chile	1.0000
San Felipe	1.0000
Universidad Católica	1.0000
Jorge Valdivia	1.0000
Jaime Valdés	1.0000

Tabla 7.5: Primeros 10 bigramas para el usuario @PrensaFutbol rankeados por el Test exacto de Fisher de cola izquierda.

$$s = \frac{2n_{11}}{n_{p1} + n_{1p}}$$

Un coeficiente cercano a 1 indicará que las palabras son muy dependientes entre si. Con NSP se obtiene una lista de bigramas rankeados por el coeficiente de Sørensen-Dice con la siguiente instrucción:

```
> statistic.pl -rank 10 dice.pm output.dice input.cnt
```

Siguiendo con el usuario @PrensaFutbol, la tabla 7.6 muestra los primeros 10 bigramas rankeados por el coeficiente de Sørensen-Dice. En caso de ser igual el coeficiente de Sørensen-Dice los términos son rankeados por frecuencia.

El hecho de que para este coeficiente no se le permita asignarle una significancia a su valor se ve reflejado en los valores de la tabla 7.6. Este coeficiente prioriza los bigramas en donde las palabras salen exactamente siempre juntas, por ejemplo, fuera del contexto del fútbol (chileno) es difícil encontrar la palabra "Mayne" asociada a otra palabra[6].

[6]Harold Mayne-Nicholls es el presidente de la Asociación Nacional de Fútbol Profesional (ANFP) de Chile.

Bigrama	**Coef de Sørensen-Dice**	**Frecuencia**	f_i	f_d
Sao Paulo	1.0000	19	19	19
Mayne Nicholls	1.0000	16	16	16
EE UU	1.0000	15	15	15
Ciudad Cabo	1.0000	8	8	8
SANTA LAURA	1.0000	8	8	8
Trinidad Tobago	1.0000	7	7	7
Cesc Fábregas	1.0000	4	4	4
Ruiz Tagle	1.0000	4	4	4
Mercado Fichajes	1.0000	3	3	3
Carles Puyol	1.0000	3	3	3

Tabla 7.6: Primeros 10 bigramas para el usuario @PrensaFutbol rankeados por el coeficiente de Sørensen-Dice.

Capítulo 8

Categorización de palabras y bigramas usando un motor de búsqueda

En los capítulos previos se expuso como filtrar los tweets de palabras con poco o sin contenido semántico y además como encontrar, mediante métodos estadísticos, pares de palabras (conocidos como bigramas) que juntas corresponden a frases con significado propio, por ejemplo, la palabra "Santiago" puede hacer referencia a la ciudad capital de Chile, a la ciudad gallega, o al apostol, pero sin embargo con el bigrama "Santiago Chile" es claro que se refiere a la ciudad chilena.

El siguiente paso es poder categorizar dentro de un conjunto de temas, definidos a priori, las palabras y bigramas (y si es necesario, trigramas). Para aquello se usará un motor de búsqueda, en este caso se hará usando Google haciendo uso de la Google AJAX Search API[1] y se medirá la semejanza entre los *n*-gramas y las categorías (en base a la cantidad de páginas de resultados de la búsqueda en Google [36]) para poder acercarse a una categorización apropiada para el *n*-grama.

8.1. Google AJAX Search API

Esta API es una biblioteca Javascript que permite integrar el motor de búsqueda Google en aplicaciones o sitios web externos. La API provee una interfaz RESTful (mediante el método GET) para ambientes sin JavaScript que retorna sus resultados codificados en JSON. Un ejemplo sencillo de acceso a la API es usando `curl` de la siguiente forma:

[1] `http://code.google.com/apis/ajaxsearch/`

```
curl -e http://www.my-ajax-site.com \
'http://ajax.googleapis.com/ajax/services/search/web?v=1.0
 &q=Paris%20Hilton&key=INSERT-YOUR-KEY'
```

Este llamado a la API consultará en Google la palabra clave (más bien frase) "Paris Hilton"[2]. El resultado codificado en JSON que devuelve este llamado a la API es el que se ve a continuación:


```
{"responseData": {
 "results": [
  {
   "GsearchResultClass": "GwebSearch",
   "unescapedUrl": "http://en.wikipedia.org/wiki/Paris_Hilton",
   "url": "http://en.wikipedia.org/wiki/Paris_Hilton",
   "visibleUrl": "en.wikipedia.org",
   "cacheUrl": "http://www.google.com/search?q\u003dcache:TwrPfhd22hYJ:en.wikipedia.org",
   "title": "\u003cb\u003eParis Hilton\u003c/b\u003e - Wikipedia, the free encyclopedia",
   "titleNoFormatting": "Paris Hilton - Wikipedia, the free encyclopedia",
   "content": "\[1\] In 2006, she released her debut album..."
  },
  {
   "GsearchResultClass": "GwebSearch",
   "unescapedUrl": "http://www.imdb.com/name/nm0385296/",
   "url": "http://www.imdb.com/name/nm0385296/",
   "visibleUrl": "www.imdb.com",
   "cacheUrl": "http://www.google.com/search?q\u003dcache:1i34KkqnsooJ:www.imdb.com",
   "title": "\u003cb\u003eParis Hilton\u003c/b\u003e",
   "titleNoFormatting": "Paris Hilton",
   "content": "Self: Zoolander. Socialite \u003cb\u003eParis Hilton\u003c/b\u003e..."
  },
  ...
 ],
 "cursor": {
  "pages": [
   { "start": "0", "label": 1 },
   { "start": "4", "label": 2 },
   { "start": "8", "label": 3 },
   { "start": "12","label": 4 }
  ],
  "estimatedResultCount": "59600000",
  "currentPageIndex": 0,
  "moreResultsUrl": "http://www.google.com/search?oe\u003dutf8\u0026ie\u003dutf8..."
 }
}
, "responseDetails": null, "responseStatus": 200}
```


[2]Paris Hilton es una celebridad, autora, modelo, actriz, diseñadora y cantante estadounidense.

En este objeto retornado el valor de interés será el de `"estimatedResultCount"`. En el ejemplo anterior, este valor nos indica que la búsqueda en Google de "Paris Hilton" devuelve aproximadamente 59600000 páginas. Este valor puede que no necesariamente equivalga al valor de hacer la misma búsqueda en Google.com[3].

Para este trabajo se desarrolló la clase `GoogleAJAXAPI` para hacer llamados a la API. El constructor de la clase recibe tres parámetros, la API Key[4] (que si bien no es necesaria Google recomienda su uso), la dirección IP de donde se ejecutará el código y la URL necesaria para enviar como el parámetro `Referer` en el encabezado de la petición HTTP.

```
1  def __init__(self, apikey, userip, host):
2    """ Constructor de la clase """
3    self.apikey = apikey
4    self.userip = userip
5    self.host   = host
```

La única función de esta clase es la función `search` que es una función genérica de búsqueda. `search` recibe un único parámetro de entrada que es una lista con los términos de búsqueda y devuelve el objeto JSON que se recibió de respuesta por la API.

```
1  def search(self, searchterms):
2    """ Funcion que a partir de una lista de terminos devuelve los
       resultados de su busqueda en Google """
3    termlist = []
4    for searchterm in searchterms:
5      q = string.split(searchterm, ' ')
6      termlist.append('+'.join(q))
7    querystring = '"+"'.join(termlist)
8    querystring = unicode(querystring).encode('utf-8')
9    url = urllib.quote('http://ajax.googleapis.com/ajax/services/search/
       web?v=1.0&q="' + querystring + '"&key=' + self.apikey + '&userip='
       + self.userip, '/:?=&')
10   request = urllib2.Request(url, None, {'Referer': self.host})
11   response = urllib2.urlopen(request)
12   results = simplejson.load(response)
13   return results
```

[3] http://code.google.com/apis/ajaxsearch/documentation/reference.html
[4] http://code.google.com/apis/ajaxsearch/key.html

Teniendo esto es fácil conseguir los valores estimados de los *page counts* o cantidad de resultados de búsqueda para uno o varios términos de búsqueda. Usando estos valores se estimará la semejanza o asociación entre distintos términos como ya se ha visto en [36] y en [18]. El código completo de `GoogleAJAXAPI.py` se encuentra en el anexo C

8.2. Medidas de co-ocurrencia

Sean P y Q dos términos distintos, que pueden ser tanto palabras o frases (n-gramas). La cantidad de resultados que arroja la búsqueda $P \wedge Q$ (i.e. P y Q) se puede considerar como una aproximación a la co-ocurrencia de estos términos en la web [36]. Sin embargo, la búsqueda de $P \wedge Q$ por si sola no expresa con precisión la semejanza semántica [36]. No solo hay que considerar los resultados de buscar $P \wedge Q$, sino que también se deben considerar los resultados de las búsquedas para los términos individuales P y Q. Para computar la semejanza semántica se consideraron cuatro medidas de co-ocurrencia: el coeficiente de Jaccard, el coeficiente de superposición (*overlap*) o de Simpson, el coeficiente de Sørensen-Dice (introducido en el capítulo anterior) y PMI (*point-wise mutual information*). De ahora en adelante se adoptará la notación $H(P)$ para denotar la cantidad de resultados de buscar P. $H(P \cap Q)$ denota la cantidad de resultados de buscar $P \wedge Q$.

8.2.1. Coeficiente de Jaccard

El índice o coeficiente de semejanza de Jaccard[5] [37] usado para comparar la semejanza y diversidad en conjuntos de muestras, se define formalmente como el tamaño de la intersección dividido por el tamaño de la unión de los conjuntos:

$$J(P,Q) = \frac{|P \cap Q|}{|P \cup Q|}$$

Se modificará este índice de la siguiente forma para trabajar con los resultados de búsqueda:

$$WebJaccard(P,Q) = \begin{cases} 0 & \text{si } H(P \cap Q) \leq c \\ \frac{H(P \cap Q)}{H(P)+H(Q)-H(P \cap Q)} & \end{cases}$$

[5] Atribuido a Paul Jaccard (1868 – 1944), botánico suizo.

Se usará el valor umbral c para discriminar de inmediato términos que aparecen muy pocas veces juntos (y por lo tanto tendrían poca semejanza).

8.2.2. Coeficiente de superposición

Esta medida de semejanza se relaciona al índice de Jaccard [38]. Se define como:

$$Overlap(P,Q) = \frac{|P \cap Q|}{\text{mín}(|P|,|Q|)}$$

Si $P \subset Q$ o $Q \subset P$ este coeficiente es igual a 1. Se le hace una modificación similar al coeficiente de Jaccard, introduciendo el valor umbral c:

$$WebOverlap(P,Q) = \begin{cases} 0 & \text{si } H(P \cap Q) \leq c \\ \frac{H(P \cap Q)}{\text{mín}(H(P),H(Q))} & \end{cases}$$

8.2.3. Coeficiente de Sørensen-Dice

Este coeficiente, definido en el capítulo anterior, se modifica de la misma forma que a los coeficientes anteriores:

$$WebOverlap(P,Q) = \begin{cases} 0 & \text{si } H(P \cap Q) \leq c \\ \frac{2H(P \cap Q)}{H(P)+H(Q)} & \end{cases}$$

PMI

La información mutua punto a punto o *point-wise mutual information* (PMI) [39] de dos variables aleatorias discretas x e y cuantifica la discrepancia entre la probabilidad de coincidencia dada su distribución conjunta versus la probabilidad de coincidencia dadas solo sus distribuciones individuales y asumiendo independencia. Matemáticamente esto se traduce de la siguiente forma:

$$PMI(x,y) = \log\left(\frac{p(x,y)}{p(x)p(y)}\right)$$

Para el caso de los resultados de las búsquedas, estas probabilidades se pueden entender como:

$$p(P,Q) = \frac{H(P \cap Q)}{N}$$

Donde, N viene a ser el número de documentos indexados por el motor de búsqueda (en nuestro caso, Google). El valor de N no es conocido exactamente, pero se estima en un valor cercano a $N = 10^{10}$. Con esto en consideración se redefine PMI:

$$WebPMI(P,Q) = \begin{cases} 0 & \text{si } H(P \cap Q) \leq c \\ \log_2 \left(\frac{\frac{H(P \cap Q)}{N}}{\frac{H(P)}{N} \frac{H(Q)}{N}} \right) & \end{cases}$$

8.2.4. Implementación

En base a lo anterior se crea el módulo para Python PageCountSS (su nombre se debe a *page count similarity scores*). Con él se pueden acceder a métodos para calcular las distintas medidas de co-ocurrencia. Por ejemplo, la función `webJaccard` calcula el coeficiente de Jaccard de la siguiente forma:

```
1 def webJaccard(p, q, r, c):
2   if r <= c:
3     return 0.0
4   else:
5     return r/(p+q-r)
```

El parámetro de entrada `p` corresponde a $H(P)$, `q` a $H(Q)$, `r` a $H(P \cap Q)$ y `c` es el valor umbral c. El código completo de este módulo se encuentra en el anexo D .

8.3. Experimentación

Para probar la efectividad de estas medidas de asociación entre palabras, se computó para un conjunto de 5 términos de prueba, obtenidos de los tweets del usuario @MarcAstr0 y que están entre los 10 primeros según el coeficiente de Sørensen-Dice, los puntajes de `webJaccard`, `webOverlap`, `webDice` y `webPMI` entre estos términos y las siguientes categorías definidas arbitrariamente:

- Television
- Tecnología
- Política
- Viajes
- Arte
- Música
- Negocios
- Actualidad
- Sociedad
- Juegos
- Deportes
- Cine

Los términos de prueba son los siguientes:

- "ZZ Top": grupo estadounidense de *southern rock* y *hard rock*.
- "Monkey Island": saga de videojuegos de aventuras producida y publicada por LucasArts.
- "Jeff Beck": guitarrista inglés de *rock/blues*
- "Google Wave": herramienta en línea que permite a sus usuarios comunicarse y colaborar en tiempo real desarrollada por Google.
- "Coco Legrand": humorista chileno.

Las tablas 8.1 a 8.5 muestran los resultados de computar los puntajes de co-ocurrencia entre estos términos y las distintas categorías. Estas tablas se encuentran ordenadas ascendentemente según el coeficiente de Sørensen-Dice, basándose en que esta medida mostró tener mejor F Measure en [36].

Como se puede apreciar, según el coeficiente de Sørensen-Dice los términos "ZZ Top" (tabla 8.1) y "Monkey Island" (tabla 8.2) se categorizaron correctamente (en las categorías "Música" y "Juegos" respectivamente. Para "Jeff Beck" (tabla 8.3), que se esperaría que se asociaría más fuertemente con "Música", el puntaje de Sørensen-Dice más alto es con la categoría "Cine", sin embargo, el coeficiente de superposición clasifica correctamente este término bajo "Música". Algo similar sucede con "Google Wave" (tabla 8.4) a quien el coeficiente de superposición clasifica correctamente bajo "Tecnología". En el caso de "Coco Legrand" (tabla

Categoría	Jaccard	Overlap	Dice	PMI
		ZZ Top		
Television	0.000269	0.091858	0.000538	2.494534
Politica	0.000390	0.008351	0.000780	3.093996
Viajes	0.000584	0.008372	0.001168	3.711617
Actualidad	0.000629	0.007912	0.001257	3.833541
Negocios	0.000813	0.011858	0.001625	4.187064
Sociedad	0.000837	0.015094	0.001672	4.207490
Tecnologia	0.001249	0.027557	0.002496	4.770013
Deportes	0.001261	0.028601	0.002519	4.781423
Arte	0.001476	0.061169	0.002948	4.978675
Juegos	0.002464	0.036117	0.004915	5.782835
Cine	0.003767	0.068476	0.007507	6.373313
Musica	0.003875	0.127140	0.007719	6.376741

Tabla 8.1: Puntajes obtenidos para las medidas de co-ocurrencia entre el término "ZZ Top" y un conjunto de categorías.

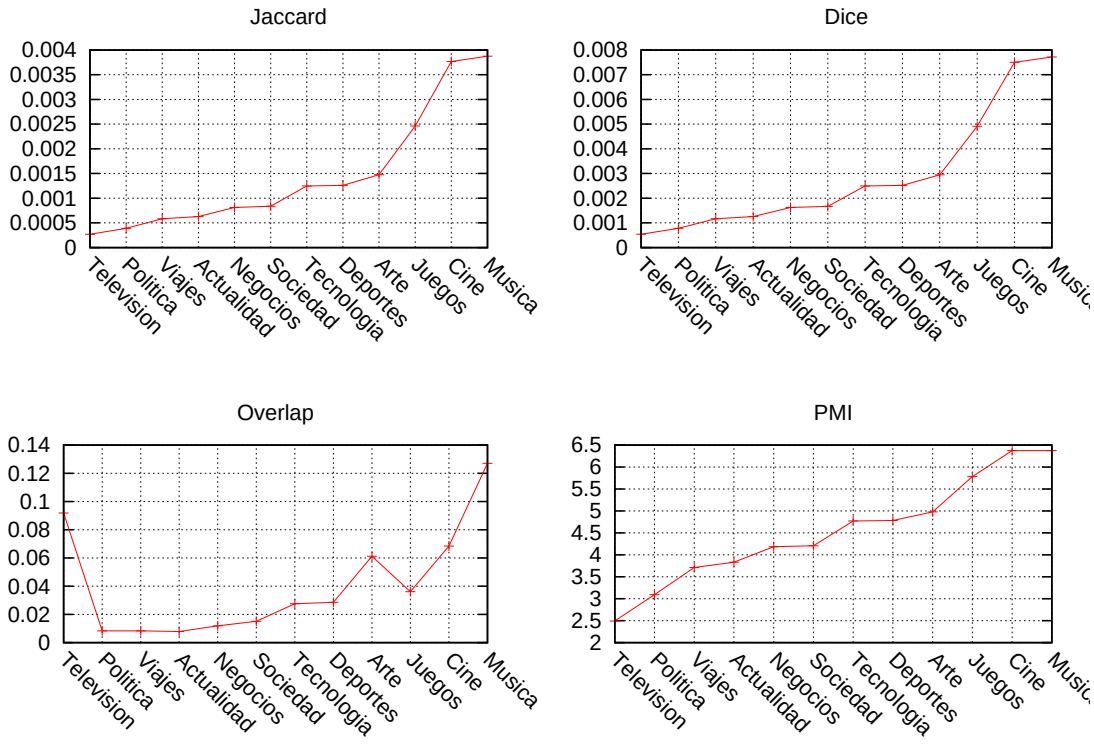

Figura 8.1: Gráfico de los puntajes obtenidos para las medidas de co-ocurrencia entre el término "ZZ Top" y un conjunto de categorías.

Categoría	Jaccard	Overlap	Dice	PMI
Monkey Island				
Television	0.000193	0.145833	0.000386	3.161377
Sociedad	0.000425	0.016481	0.000849	4.334366
Politica	0.000510	0.023611	0.001020	4.593488
Negocios	0.001088	0.033843	0.002174	5.700039
Viajes	0.001088	0.033241	0.002174	5.700993
Arte	0.001179	0.106944	0.002355	5.784661
Actualidad	0.001709	0.045556	0.003413	6.358995
Tecnologia	0.001835	0.087500	0.003664	6.436856
Musica	0.002067	0.148148	0.004125	6.597365
Deportes	0.002616	0.128241	0.005219	6.946127
Cine	0.003457	0.135185	0.006890	7.354580
Juegos	0.009159	0.284722	0.018152	8.761644

Tabla 8.2: Puntajes obtenidos para las medidas de co-ocurrencia entre el término "Monkey Island" y un conjunto de categorías.

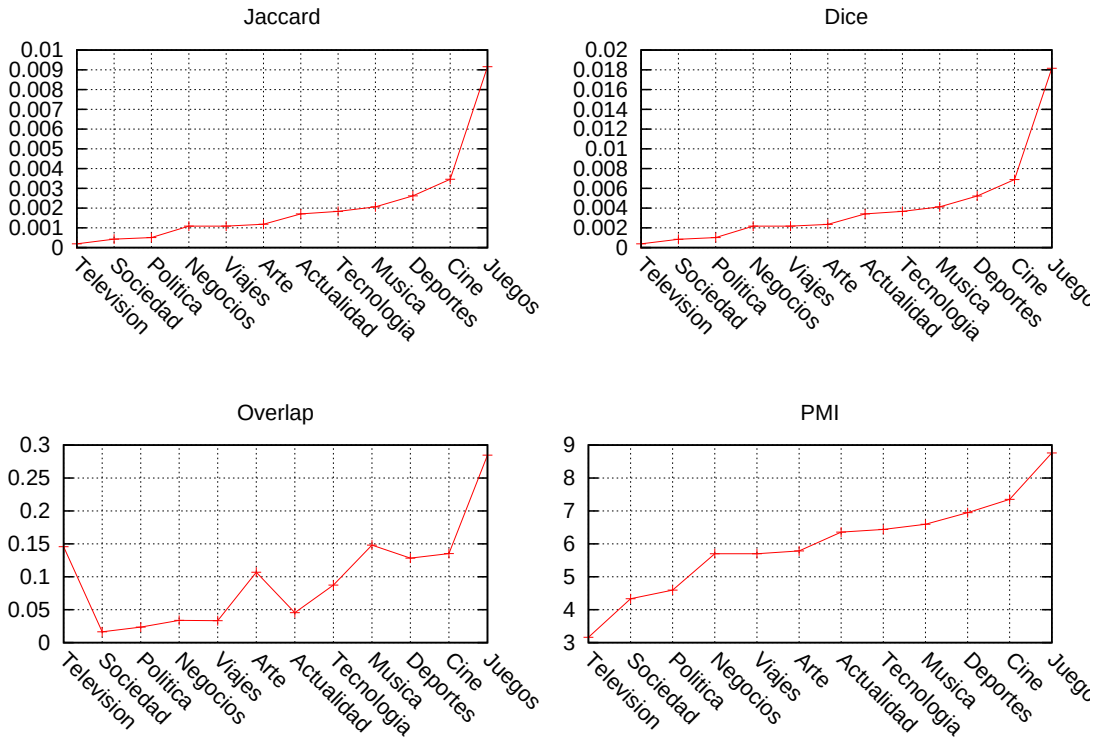

Figura 8.2: Gráfico de los puntajes obtenidos para las medidas de co-ocurrencia entre el término "Monkey Island" y un conjunto de categorías.

Categoría	Jaccard	Overlap	Dice	PMI
Jeff Beck				
Television	0.000173	0.083186	0.000345	2.351466
Politica	0.000438	0.013068	0.000876	3.740043
Sociedad	0.000669	0.016785	0.001337	4.360663
Tecnologia	0.000753	0.023156	0.001504	4.518980
Viajes	0.000808	0.016018	0.001614	4.647707
Actualidad	0.000954	0.016549	0.001905	4.898084
Arte	0.001573	0.091445	0.003141	5.558782
Deportes	0.001586	0.050147	0.003166	5.591522
Negocios	0.001814	0.036578	0.003621	5.812182
Juegos	0.002412	0.048968	0.004812	6.221986
Musica	0.002603	0.119764	0.005192	6.290519
Cine	0.002635	0.066667	0.005256	6.334680

Tabla 8.3: Puntajes obtenidos para las medidas de co-ocurrencia entre el término "Jeff Beck" y un conjunto de categorías.

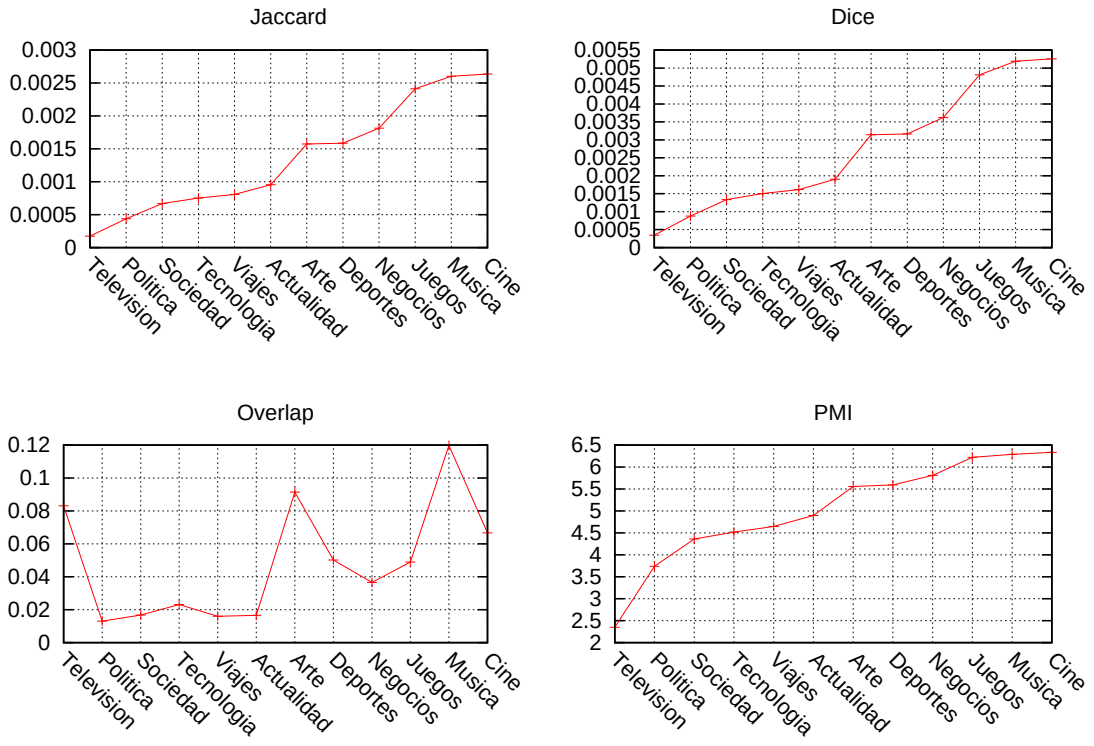

Figura 8.3: Gráfico de los puntajes obtenidos para las medidas de co-ocurrencia entre el término "Jeff Beck" y un conjunto de categorías.

Categoría	Jaccard	Overlap	Dice	PMI
Google Wave				
Television	0.000199	0.038398	0.000398	1.236164
Viajes	0.001317	0.011213	0.002630	4.133237
Politica	0.001375	0.017197	0.002747	4.136154
Deportes	0.001424	0.018846	0.002845	4.179580
Sociedad	0.001532	0.016254	0.003060	4.314352
Actualidad	0.001612	0.012132	0.003219	4.450176
Arte	0.001796	0.042756	0.003585	4.462005
Negocios	0.001933	0.016726	0.003859	4.683253
Musica	0.002608	0.049470	0.005202	5.014949
Cine	0.003724	0.039812	0.007421	5.590901
Tecnologia	0.004283	0.055006	0.008530	5.767159
Juegos	0.005483	0.047585	0.010906	6.180679

Tabla 8.4: Puntajes obtenidos para las medidas de co-ocurrencia entre el término "Google Wave" y un conjunto de categorías.

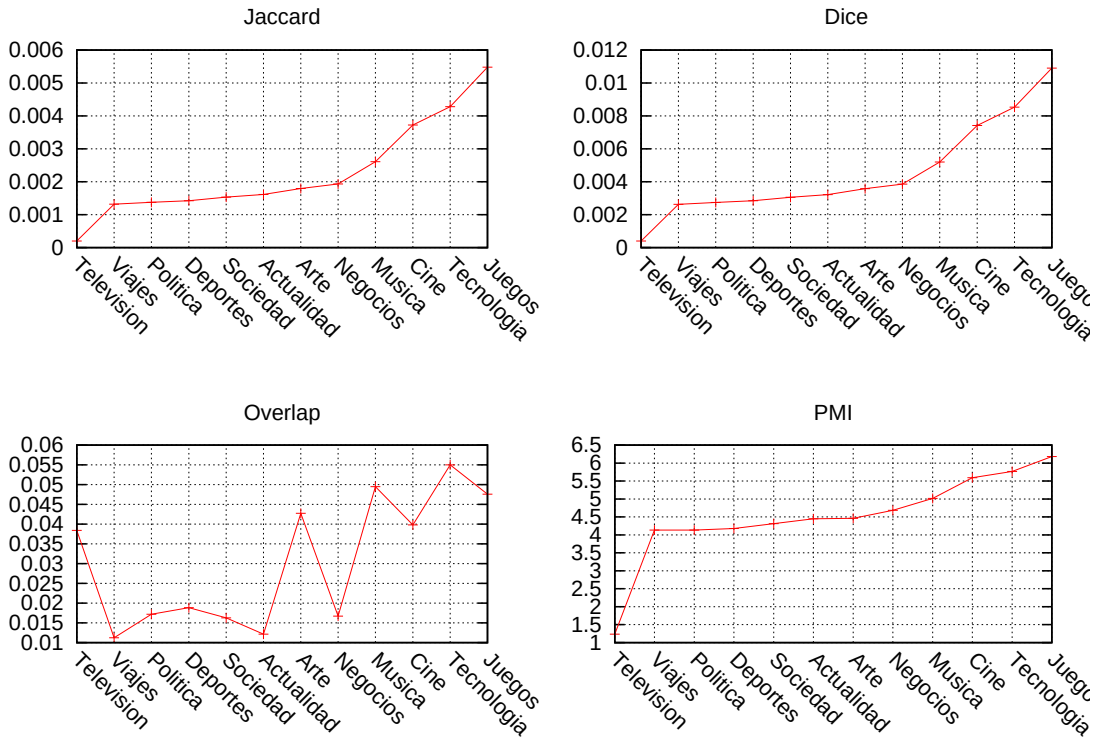

Figura 8.4: Gráfico de los puntajes obtenidos para las medidas de co-ocurrencia entre el término "Google Wave" y un conjunto de categorías.

Categoría	Jaccard	Overlap	Dice	PMI
Coco Legrand				
Television	0.000004	0.059316	0.000009	1.863555
Tecnologia	0.000037	0.032308	0.000075	4.999451
Politica	0.000051	0.042906	0.000103	5.455201
Viajes	0.000077	0.041966	0.000153	6.037255
Arte	0.000086	0.141880	0.000171	6.192474
Musica	0.000144	0.188034	0.000287	6.941320
Negocios	0.000167	0.093162	0.000334	7.160946
Actualidad	0.000185	0.088034	0.000370	7.309432
Sociedad	0.000227	0.158974	0.000455	7.604242
Juegos	0.000266	0.149573	0.000533	7.832935
Deportes	0.000293	0.260684	0.000586	7.969573
Cine	0.000307	0.217094	0.000614	8.037962

Tabla 8.5: Puntajes obtenidos para las medidas de co-ocurrencia entre el término "Coco Legrand" y un conjunto de categorías.

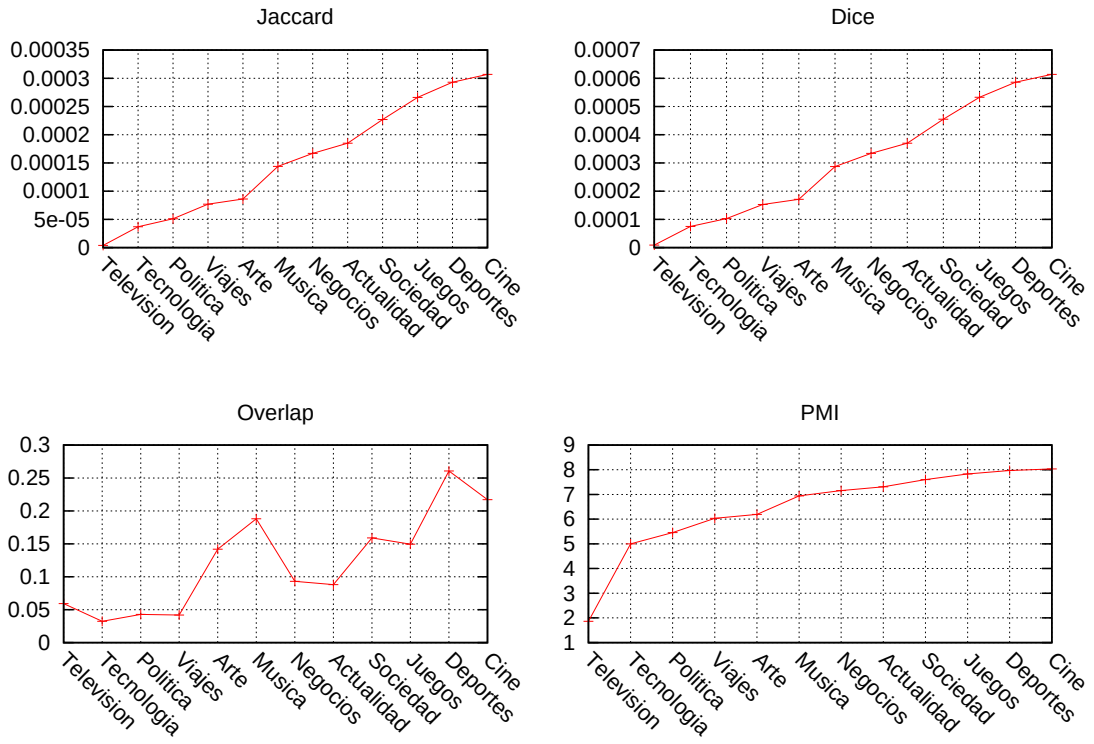

Figura 8.5: Gráfico de los puntajes obtenidos para las medidas de co-ocurrencia entre el término "Coco Legrand" y un conjunto de categorías.

8.5), que se trata de un humorista, el coeficiente de superposición lo ubica más semejante a "Deportes" mientras que los otros tres coeficientes adquieren mayor puntaje en "Cine". Dentro de estas categorías este término es difícil de clasificar. Se hubiera esperado un mayor puntaje en la categoría "Televisión", debido a que buena parte de la carrera de este humorista se ha desarrollado en la pantalla chica, o también en "Sociedad", debido al comentario social que el humorista le imprime a sus monólogos. El problema para clasificar este término podría radicar en las categorías escogidas, el significado de algunas podría ser algo ambigüo o en momentos compartir significado con otras de las categorías. Otro punto clave es que los resultados de Google AJAX Search API difieren de los que uno obtendría usando Google.com. Esto porque con Google.com se obtienen resultados regionalizados a la localización de quién efectúa la consulta, y esto afecta la cantidad de resultados obtenidos para distintos términos.

Parte III

Análisis y conclusiones

Capítulo 9

Análisis experimental

En base a lo señalado en la sección 8.3, en este capítulo se analizará la efectividad de categorizar usando alguna de las medidas de semejanza. Para ello se seleccionó al azar un subconjunto de usuarios para los cuales se les capturó sus tweets y se categorizó para ellos un subconjunto de 20 términos. Se analizarán dos aspectos, efectividad, si el clasificador pudo o no relacionar el término con alguna de las categorías definidas, y precisión, si clasificó "bien" el término.

9.1. Preparación

Para este experimento se uso el programa `classify.py`, que hace uso de las clases `GoogleAJAXAPI` y `PageCountSS` descritas en capítulos anteriores. Este programa recibe de entrada un archivo con un listado de bigramas entregado por NSP y entrega un archivo CSV con los resultados. Además, se le debe definir los siguientes parámetros de entrada:

- Valor del umbral c.
- Cantidad de bigramas a analizar por archivo.
- Medida de co-ocurrencia a usar (Jaccard, superposición, Sørensen-Dice o PMI).

Para c se usó el valor usado por Bollegala et al. [16] en su experimentación. Se seleccionó al azar 13 de los 44 usuarios del conjunto de tweets y para cada uno de ellos se les analizaron sus 20 primeros términos rankeados usando el coeficiente de superposición. Para cada uno de los términos se busca la categoría con mayor puntaje de co-ocurrencia, de no ser posible el

programa indica en el archivo de salida que dicho término no se pudo clasificar. Las categorías que se usaron fueron las mismas de la sección 8.3.

9.2. Medición de efectividad

Para este análisis se mide la efectividad como la relación entre total de términos y total de términos que se pudieron clasificar (estén bien o mal clasificados). La tabla 9.1 resume estos resultados:

Total de términos	260	100 %
Términos clasificados con éxito	219	84.23 %
Términos sin clasificar	41	15.77 %

Tabla 9.1: Cantidades de términos clasificados y no clasificados

Los resultados de la tabla 9.1 indican que el coeficiente de superposición es efectivo a la hora de encontrar la co-ocurrencia o semejanza entre los términos y las categorías. A continuación de listan algunos de los términos que no se lograron clasificar:

- "Remazterizado Exelente"
- "wuajajajjajajaja wuajajajjajajaja"
- "moralina tuitera"
- "conoce papers"
- "echele vistazo"

Las razones por las cuales no se pudieron clasificar estos términos pueden ser diversas, sin embargo se pueden intuir al ver estos términos. El primero está mal escrito (quizás deliberadamente), el segundo término es una deformación del lenguaje. El tercer término podría corresponder a una jerga propia al autor o a la red social del autor. Los últimos dos parecen describir acciones que así solas no tienen ningún significado.

9.3. Medición de precisión

Si bien muchos términos se lograron clasificar (es decir, se encontró una mayor semejanza con alguna de las categorías) esto no significa necesariamente que se haya clasificado bien, es decir, que en realidad corresponda a esa categoría. Se entenderá la precisión como cuan bien logra clasificar los términos. Para esto, se evalúan manualmente los términos clasificados y se comparan como aparecen en los tweets originales. Los resultados de esta evaluación se resumen en la tabla 9.2:

Total de términos	260	100 %
Términos clasificados	219	84.23 %
Términos bien clasificados	96	36.92 %
Términos mal clasificados	123	47.31 %
Términos sin clasificar	41	15.77 %

Tabla 9.2: Cantidades de términos clasificados (bien y mal) y no clasificados

Del total, un poco más de un tercio de los términos resultó bien clasificado, y limitándose solo a los terminos que fueron clasificados con éxito, corresponde a un 43.8 %. Las posibles causas de esta imprecisión se discutirán en el capítulo siguiente.

Capítulo 10

Conclusiones e investigación futura

10.1. Discusión

Los análisis hechos en el capítulo 9 y la sección 8.3 mostraron que para una cantidad significativa de términos es posible categorizarlos correctamente en base a medidas de co-ocurrencia usando los resultados de la búsqueda en Google. Sin embargo, en la mayoría de los casos se produce una categorización errónea o no se logra encontrar una asociación. Las razones por la cual sucede esto se exponen a continuación.

10.1.1. Presencia de ruido

Si bien el procedimiento descrito en el capítulo 6 se encargó de filtrar una cantidad importante de texto que solo generaría ruido a la hora de medir semejanza (palabras vacías, hashtags, URLs), una inspección manual del texto procesado reveló una cantidad significativa de términos que entrarían en esta clasificación y que no fueron filtrados, que serían:

- Abreviaciones de palabras vacías existentes, como "tb" para decir "también", "q" o "k" para decir "que", "x" para decir "por".
- Expresiones para emociones como la risa, con "jajaja" sus diversas variaciones.
- Secuencias de caracteres para expresar emociones, también conocidos como *emoticones*, por ejemplo `:D` (cara felíz) o `:(` (cara triste),

A estos hay que sumarle también que muchos usuarios escriben de vez en cuando tweets en otros idiomas, particularmente el inglés, y el preprocesamiento del capítulo 6 solo toma en

cuenta palabras vacías para el idioma castellano, haciendo que muchas palabras en inglés que generan ruido no se filtren.

10.1.2. Selección de categorías

Para mejorar la precisión de la categorización, las categorías deben escogerse de tal forma que entre dos o más no compartan mucho significado (sean lo más disimil posible). Por ejemplo, en el conjunto de categorías de los experimentos, estaban "Política" y "Sociedad", términos con definiciones distintas, pero que en la práctica comparten muchos términos asociados. Por ejemplo, el presidente Sebastián Piñera, parte del ámbito *político*, ejerce un cargo que afecta a la *sociedad* chilena. Por lo tanto, se vuelve confuso categorizar un término como "Sebastián Piñera" en una de estas dos categorías, lo que lleva al siguiente punto...

10.1.3. Contexto de las palabras

El gran inconveniente que tiene la metodología expuesta en este trabajo es que no toma en cuenta el contexto en el cual se están usando las palabras. Por ejemplo, la palabra "Jaguar" puede hacer alusión al fabricante inglés de automóviles como también al felino. La desambigüación de la palabra ocurre en el contexto en el cual se le está dando uso dentro de una oración (o en el caso específico de este trabajo, dentro de un tweet). La limitante entregada por las medidas de co-ocurrencia usadas en este trabajo ya ha sido expuesta por Bollegala et al. [16] y Sahami et al. [17], en donde ambos sugieren la extracción de trozos de texto para encontrar patrones léxico-sintácticos para complementar las medidas de co-ocurrencia.

10.2. Conclusión

En base a lo expuesto en este trabajo se pueden llegar a las siguientes conclusiones:

Twitter es una fuente muy rica en datos para hacer minería de texto. Y por ello puede entregar mucha información relativa a las personas que usan el servicio. Esta información puede ser aprovechada para filtrar contenido y entregarlo de forma más eficiente y así, como se mencionó en el capítulo 1, conectar la demanda con la enorme oferta. Por ejemplo, una aplicación web de e-commerce, que organiza sus productos en un conjunto de categorías,

usando la información de sus usuarios que usan Twitter, puede relacionar cuales categorías serían más afines a ellos, y de esa forma recomendarles productos y ofertas en forma eficiente. Hoy en día, la integración de Twitter con aplicaciones web es sencilla gracias a OAuth[1].

Usar un motor de búsqueda se puede considerar como una forma válida para medir semejanza semántica entre términos. Los motores de búsqueda, en particular Google, permiten acceder a millones de documentos en la web, y de esta forma, la web podría considerarse como el cuerpo lingüístico (conjunto, normalmente muy amplio, de ejemplos reales de uso de una lengua) más grande que hay para varios idiomas. La ausencia de una base de datos léxica para el idioma castellano (como lo es WordNet para el inglés) hace que el uso de Google u otros motores de búsqueda tome fuerza para medir semejanza entre palabras.

Los resultados de búsqueda o page-counts por si solos no son suficientes para medir con precisión la semejanza entre términos. Como se argumentó en las secciones 10.1.2 y 10.1.3 se deben tomar en cuenta otros factores como el contexto en el que están los términos para encontrar una mejor semejanza semántica.

El enfoque utilizado en este trabajo se limita a relacionar los términos claves encontrados en Twitter con un conjunto de términos o categorías definidas a priori. Una futura implementación podría encontrar las categorías de interés por si sola, como lo que hacen Hong et al. [40] al implementar el algoritmo LDA (*Latent Dirichlet Allocation*) [41] para el modelamiento de temas o tópicos dentro del mismo Twitter.

Si se refinan estos métodos para extraer información desde Twitter, se puede llegar a una forma eficiente y novedosa para encontrar los intereses de los usuarios de este servicio, y de esta manera hacerles llegar y recomendarles contenido que realmente les sea de interés y conectando la demanda con la Larga Cola.

[1] `http://oauth.net/`

Parte IV

Referencias y anexos

Referencias

[1] C. Anderson, *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion, 2006.

[2] S. Alag, *Collective Intelligence in Action*. Greenwich, CT, USA: Manning Publications Co., 2008.

[3] J. H. Grace, D. Zhao, and d. boyd, "Microblogging: what and how can we learn from it?," in *CHI EA '10: Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems*, (New York, NY, USA), pp. 4517–4520, ACM, 2010.

[4] N. Banerjee, D. Chakraborty, K. Dasgupta, S. Mittal, A. Joshi, S.Ñagar, A. Rai, and S. Madan, "User interests in social media sites: an exploration with micro-blogs," in *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, (New York, NY, USA), pp. 1823–1826, ACM, 2009.

[5] D. Zhao and M. B. Rosson, "How and why people twitter: the role that micro-blogging plays in informal communication at work," in *GROUP '09: Proceedings of the ACM 2009 international conference on Supporting group work*, (New York, NY, USA), pp. 243–252, ACM, 2009.

[6] A. Java, X. Song, T. Finin, and B. Tseng, "Why we twitter: understanding microblogging usage and communities," in *WebKDD/SNA-KDD '07: Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, (New York, NY, USA), pp. 56–65, ACM, 2007.

[7] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?," in *WWW '10: Proceedings of the 19th international conference on World wide web*, (New York, NY, USA), pp. 591–600, ACM, 2010.

[8] M. E. J. Newman and J. Park, "Why social networks are different from other types of networks," 2003.

[9] J. Weng, E.-P. Lim, J. Jiang, and Q. He, "Twitterrank: finding topic-sensitive influential twitterers," in *WSDM '10: Proceedings of the third ACM international conference on Web search and data mining*, (New York, NY, USA), pp. 261–270, ACM, 2010.

[10] M. Mendoza, B. Poblete, and C. Castillo, "Twitter under crisis: Can we trust what we rt?," in *1st Workshop on Social Media Analytics (SOMA '10)*, ACM Press, 2010.

[11] S. Vieweg, A. L. Hughes, K. Starbird, and L. Palen, "Microblogging during two natural hazards events: what twitter may contribute to situational awareness," in *CHI '10: Proceedings of the 28th international conference on Human factors in computing systems*, (New York, NY, USA), pp. 1079–1088, ACM, 2010.

[12] A. Budanitsky and G. Hirst, "Evaluating wordnet-based measures of lexical semantic relatedness," *Comput. Linguist.*, vol. 32, no. 1, pp. 13–47, 2006.

[13] T. Hughes and D. Ramage, "Lexical semantic relatedness with random graph walks," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, (Prague, Czech Republic), pp. 581–589, Association for Computational Linguistics, June 2007.

[14] T. P. University, T. Pedersen, and S. Patwardhan, "Wordnet::similarity - measuring the relatedness of concepts," pp. 1024–1025, 2004.

[15] A. Gledson and J. Keane, "Using web-search results to measure word-group similarity," in *COLING '08: Proceedings of the 22nd International Conference on Computational Linguistics*, (Morristown, NJ, USA), pp. 281–288, Association for Computational Linguistics, 2008.

[16] D. Bollegala, Y. Matsuo, and M. Ishizuka, "Measuring the similarity between implicit semantic relations using web search engines," in *WSDM '09: Proceedings of the Second ACM International Conference on Web Search and Data Mining*, (New York, NY, USA), pp. 104–113, ACM, 2009.

[17] M. Sahami and T. D. Heilman, "A web-based kernel function for measuring the similarity of short text snippets," in *WWW '06: Proceedings of the 15th international conference on World Wide Web*, (New York, NY, USA), pp. 377–386, ACM, 2006.

[18] Y. Matsuo, T. Sakaki, K. Uchiyama, and M. Ishizuka, "Graph-based word clustering using a web search engine," in *EMNLP '06: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, (Morristown, NJ, USA), pp. 542–550, Association for Computational Linguistics, 2006.

[19] Wikipedia, "Blog — wikipedia, la enciclopedia libre." `http://es.wikipedia.org/w/index.php?title=Blog&oldid=39263759`, 2010.

[20] Wikipedia, "Blog — wikipedia, the free encyclopedia." `http://en.wikipedia.org/w/index.php?title=Blog&oldid=376463067`, 2010.

[21] Wikipedia, "Microblogging — wikipedia, the free encyclopedia." `http://en.wikipedia.org/w/index.php?title=Microblogging&oldid=376982799`, 2010.

[22] Wikipedia, "Twitter — wikipedia, the free encyclopedia." `http://en.wikipedia.org/w/index.php?title=Twitter&oldid=377157885`, 2010.

[23] K. Makice, *Twitter API: Up and Running Learn How to Build Applications with the Twitter API*. O'Reilly Media, Inc., 2009.

[24] Twitter, "The twitter glossary." `http://support.twitter.com/groups/31-twitter-basics/topics/104-welcome-to-twitter-support/articles/166337-the-twitter-glossary`, 2010.

[25] Twitter, "Twitter developers." `http://dev.twitter.com/`, 2010.

[26] Twitter, "Api documentation." `http://dev.twitter.com/doc/`, 2010.

[27] L. Richardson, "Beautiful soup documentation." http://www.crummy.com/software/BeautifulSoup/documentation.html, 2008.

[28] T. Segaran, *Programming collective intelligence*. O'Reilly, 2007.

[29] R. A. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[30] S. Banerjee and T. Pedersen, "The design, implementation, and use of the ngram statistics package," in *CICLing'03: Proceedings of the 4th international conference on Computational linguistics and intelligent text processing*, (Berlin, Heidelberg), pp. 370–381, Springer-Verlag, 2003.

[31] T. Pedersen, M. Kayaalp, and R. Bruce, "Significant lexical relationships," in *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96*, pp. 455–460, 1996.

[32] Wikipedia, "Likelihood-ratio test — wikipedia, the free encyclopedia." http://en.wikipedia.org/w/index.php?title=Likelihood-ratio_test&oldid=383950698, 2010.

[33] Wikipedia, "Fisher's exact test — wikipedia, the free encyclopedia." http://en.wikipedia.org/w/index.php?title=Fisher%27s_exact_test&oldid=386662300, 2010.

[34] T. Pedersen, "Fishing for exactness," in *In Proceedings of the South-Central SAS Users Group Conference*, pp. 188–200, 1996.

[35] Wikipedia, "Dice's coefficient — wikipedia, the free encyclopedia." http://en.wikipedia.org/w/index.php?title=Dice%27s_coefficient&oldid=357695079, 2010.

[36] "Measuring semantic similarity between words using web search engines," in *WWW '07: Proceedings of the 16th international conference on World Wide Web*, (New York, NY, USA), pp. 757–766, ACM, 2007.

[37] Wikipedia, "Jaccard index — wikipedia, the free encyclopedia." http://en.wikipedia.org/w/index.php?title=Jaccard_index&oldid=385884659, 2010.

[38] Wikipedia, "Overlap coefficient — wikipedia, the free encyclopedia." http://en.wikipedia.org/w/index.php?title=Overlap_coefficient&oldid=303769361, 2009.

[39] Wikipedia, "Pointwise mutual information — wikipedia, the free encyclopedia." http://en.wikipedia.org/w/index.php?title=Pointwise_mutual_information&oldid=376133468, 2010.

[40] L. Hong and B. D. Davidson, "Empirical study of topic modeling in twitter," in *Proceedings of the SIGKDD Workshop on Social Media Analytics (SOMA) at KDD 2010*, (Washington, DC, USA), July 2010.

[41] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.

Anexos

A . fetchtweets.py

../src/fetchtweets.py

```
1  from BeautifulSoup.BeautifulSoup import BeautifulStoneSoup
2  import urllib2
3  import math
4  import sys
5  import time
6  from time import gmtime, strftime
7
8  usersfile = open('users.txt', 'r')
9
10 requests = 0
11
12 for line in usersfile:
13   user_url = 'http://api.twitter.com/1/users/show/' + line.strip() + '.xml
       '
14   timeline_url = 'https://twitter.com/statuses/user_timeline/' + line.
       strip() + '.xml'
15
16   # primero buscamos la cantidad total de tweets del usuario
17   if requests == 150:
18     time.sleep(3600)
19     requests = 0
20
21   requests += 1
22   print '[' + strftime("%Y-%m-%d %H:%M:%S", gmtime()) + ']', 'Request #',
       requests, ' ', user_url
23   request = urllib2.Request(user_url)
24   response = urllib2.urlopen(request)
25   xml = response.read()
26
27   soup = BeautifulStoneSoup(xml)
28   tweets = int(soup.statuses_count.contents[0])
29   # teniendo la cantidad de tweets se consigue la cantidad de paginas
30   pages = int(math.ceil(tweets/20.0))
31
32   if pages > 160:
33     page = 160
34   else:
35     page = pages
36   tweets = []
```

```
37
38    # extraer tweets para cada pagina
39    while page > 0:
40      if requests == 150:
41        time.sleep(3600)
42        requests = 0
43
44      requests += 1
45      print '[' + strftime("%Y-%m-%d %H:%M:%S", gmtime()) + ']', 'Request #'
            , requests, ' ', timeline_url + '?page=' + str(page)
46      request = urllib2.Request(timeline_url + '?page=' + str(page) )
47      response = urllib2.urlopen(request)
48      xml = response.read()
49      soup = BeautifulStoneSoup(xml, convertEntities=BeautifulStoneSoup.
            HTML_ENTITIES)
50      texts = soup.findAll('text')
51      tweets.extend(texts)
52      page -= 1
53
54    f = open('tweets/' + line.strip() + '.txt', 'w')
55
56    for tweet in tweets:
57      f.write(tweet.string.encode('UTF-8') + '\n')
58
59    f.close()
60
61  usersfile.close()
```

B . removestopwords.py

../src/removestopwords.py

```
# -*- coding: utf-8 -*-

import re
import codecs
import os

archivos = []
path = "/home/mario/Documentos/CC69F/src/tweets/"
dirList = os.listdir(path)
for fname in dirList:
  if fname.endswith(".txt"):
    archivos.append(fname)

f = codecs.open('stopwords.txt', 'r', 'utf-8')
stopwords = f.read().split('\r\n')
f.close()

i = 1

for archivo in archivos:
  print 'Procesando ' + str(i) + '/' + str(len(archivos)) + ': ' + archivo
  f = codecs.open('tweets/' + archivo, 'r', 'utf-8')
  tweets = f.read().split('\n')
  f.close()

  tweets2 = []
  tweets3 = []

  for tweet in tweets:
    twt = tweet
    # URLs
    twt = re.sub('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\)
        ,]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', ' ', twt)
    # Hashtags
    twt = re.sub(r'(\A|\s|\w|\W)#(\w+)', ' ', twt)
    # Usernames
    twt = re.sub(r'(\A|\s|\w|\W)@(\w+)', ' ', twt)
    # Retweets
    #twt = re.sub(r'(\A|\s)RT(\s|:)', ' ', twt)
    tweets2.append(twt)

  for tweet in tweets2:
    twt = tweet
    for stopword in stopwords:
      #print "stopword: " + stopword
      twt = re.sub(r'(?iu)(\A|\s|\W)'+stopword+'(\s|\W|\b)', ' ', twt)
    tweets3.append(twt)

  f = codecs.open('tweets/processed/_' + archivo, 'w', 'utf-8')

```

```
50    for tweet in tweets3:
51      #print tweet
52      f.write(tweet + '\n')
53
54    f.close()
55    i += 1
```

C . GoogleAJAXAPI.py

../src/GoogleAJAXAPI.py

```
1  # -*- coding: utf-8 -*-
2
3  import urllib2
4  import urllib
5  import simplejson
6  import string
7  import codecs
8
9  class GoogleAJAXAPI:
10
11   def __init__(self, apikey, userip, host):
12     """ Constructor de la clase """
13     self.apikey = apikey
14     self.userip = userip
15     self.host   = host
16
17   def search(self, searchterms):
18     """ Funcion que a partir de una lista de terminos devuelve los
           resultados de su busqueda en Google """
19     termlist = []
20     for searchterm in searchterms:
21       q = string.split(searchterm, ' ')
22       termlist.append('+'.join(q))
23     querystring = '"+"'.join(termlist)
24     querystring = unicode(querystring).encode('utf-8')
25     url = urllib.quote('http://ajax.googleapis.com/ajax/services/search/
          web?v=1.0&gl=cl&q="' + querystring + '"&key=' + self.apikey + '&
          userip=' + self.userip, '/:?=&')
26     request = urllib2.Request(url, None, {'Referer': self.host})
27     response = urllib2.urlopen(request)
28     results = simplejson.load(response)
29     return results
```

D . PageCountSS.py

../src/PageCountSS.py

```
# -*- coding: utf-8 -*-

import math

def webJaccard(p, q, r, c):
  if r <= c:
    return 0.0
  else:
    return r/(p+q-r)

def webOverlap(p, q, r, c):
  if r <= c:
    return 0.0
  else:
    return r/min(p, q)

def webDice(p, q, r, c):
  if r <= c:
    return 0.0
  else:
    return (2*r)/(p+q)

def webPMI(p, q, r, N, c):
  if r <= c:
    return 0.0
  else:
    return math.log((r/N)/((p/N)*(q/N)), 2)
```

E . classify.py

../src/classify.py

```
# -*- coding: utf-8 -*-

from GoogleAJAXAPI import GoogleAJAXAPI as google
import PageCountSS as ss
import urllib2
import simplejson
import math
import codecs
import csv
import os
import sys

# Constantes
APIKEY  = 'ABQIAAAAmJ8tw8qdjArRiJCW2oLM4xT-
    sUc5y07Fl4uOhL7Z2E9gV2iTKRRE8Cm5xUV-e86-hp_tTg5oBI6Shg'
USERIP  = '200.104.16.14'
HOST    = 'kv7.servebeer.com'

N = math.pow(10, 10)

if __name__ == "__main__":
  g = google(APIKEY, USERIP, HOST)

  f = codecs.open('categories.txt', 'r', 'utf-8')
  categories = f.read().split('\n')
  f.close()

  q                     = int(sys.argv[1])              # cantidad de
      terminos a obtener
  c             = float(sys.argv[2])    # valor umbral
  coef          = sys.argv[3]                     # medida de co-ocurrencia
      a usar
  ruta          = sys.argv[4]                     # archivo de entrada
  salida        = sys.argv[5]                     # archivo de salida
  archivo       = os.path.basename(ruta)

  f = open(ruta, 'r')
  lines = f.read().split('\n')
  f.close()

  keywords = []

  # obtenemos los top 20 keywords
  for i in range(q):
    keyword = lines[i+1].split('<>')[0] + ' ' + lines[i+1].split('<>')[1]
    keywords.append(keyword.decode('utf-8'))

  writer = csv.writer(open(salida, 'wb'), delimiter=',', quotechar='"',
      quoting=csv.QUOTE_MINIMAL)
  writer.writerow(['Keyword', 'Category', 'Score'])
```

```
47
48  for keyword in keywords:
49
50    scores = {}
51    maxvalue = 'No se pudo clasificar'
52    maxscore = 0.0
53
54    for category in categories:
55
56      p = g.search([keyword])
57      q = g.search([category])
58      r = g.search([keyword, category])
59
60      #se debe manejar la excepcion en caso de que la bÃºsqueda no
            devuelva resultados
61      try:
62        p_ = float(simplejson.JSONDecoder().decode(p['responseData']['
              cursor']['estimatedResultCount']))
63      except KeyError:
64        p_ = 0.0
65      try:
66        q_ = float(simplejson.JSONDecoder().decode(q['responseData']['
              cursor']['estimatedResultCount']))
67      except KeyError:
68        q_ = 0.0
69      try:
70        r_ = float(simplejson.JSONDecoder().decode(r['responseData']['
              cursor']['estimatedResultCount']))
71      except KeyError:
72        r_ = 0.0
73      #print simplejson.dumps(results, sort_keys=True, indent=4)
74
75      if coef == 'jaccard':
76        scores[category] = ss.webJaccard(p_, q_, r_, c)
77      elif coef == 'overlap':
78        scores[category] = ss.webOverlap(p_, q_, r_, c)
79      elif coef == 'dice':
80        scores[category] = ss.webDice(p_, q_, r_, c)
81      elif coef == 'pmi':
82        scores[category] = ss.webPMI(p_, q_, r_, N, c)
83
84    for k, v in scores.iteritems():
85      if v > maxscore:
86        maxscore = v
87        maxvalue = k
88
89    writer.writerow([unicode(keyword).encode('utf-8'), unicode(maxvalue).
        encode('utf-8'), '%.6f' % maxscore])
```

F . Listado de palabras vacías utilizadas

- a
- acá
- ahí
- ajena
- ajenas
- ajeno
- ajenos
- al
- algo
- alguna
- algunas
- alguno
- algunos
- algún
- alli
- allá
- allí
- ambos
- ampleamos
- ante
- antes
- aquel
- aquella
- aquellas
- aquello
- aquellos
- aqui
- aquí
- arriba
- asi
- atras
- aun
- aunque
- bajo
- bastante
- bien
- cabe
- cada
- casi
- cierta
- ciertas
- cierto
- ciertos
- como
- con
- conmigo
- conseguimos
- conseguir
- consigo
- consigue
- consiguen
- consigues
- contigo
- contra
- cual
- cuales
- cualquier
- cualquiera
- cualquieras
- cuan
- cuando
- cuanta
- cuantas
- cuanto
- cuantos
- cuán
- cuánta
- cuántas
- cuánto
- cuántos
- cómo
- de
- dejar
- del
- demas
- demasiada
- demasiadas
- demasiado
- demasiados
- demás

- dentro
- desde
- donde
- dos
- durante
- e
- el
- ella
- ellas
- ello
- ellos
- empleais
- emplean
- emplear
- empleas
- empleo
- en
- encima
- entonces
- entre
- era
- erais
- eramos
- eran
- eras
- eres
- es
- esa
- esas
- ese
- eso
- esos
- esta
- estaba
- estabais
- estaban
- estabas
- estad
- estada
- estadas
- estado
- estados
- estais
- estamos
- estan
- estando
- estar
- estaremos
- estará
- estarán
- estarás
- estaré
- estaréis
- estaría
- estaríais
- estaríamos
- estarían
- estarías
- estas
- este
- estemos
- esto
- estos
- estoy
- estuve
- estuviera
- estuvierais
- estuvieran
- estuvieras
- estuvieron
- estuviese
- estuvieseis
- estuviesen
- estuvieses
- estuvimos
- estuviste
- estuvisteis
- estuviéramos
- estuviésemos
- estuvo
- está
- estábamos
- estáis
- están

- estás
- esté
- estéis
- estén
- estés
- etc
- fin
- fue
- fuera
- fuerais
- fueran
- fueras
- fueron
- fuese
- fueseis
- fuesen
- fueses
- fui
- fuimos
- fuiste
- fuisteis
- fuéramos
- fuésemos
- gueno
- ha
- habida
- habidas
- habido
- habidos
- habiendo
- habremos
- habrá
- habrán
- habrás
- habré
- habréis
- habría
- habríais
- habríamos
- habrían
- habrías
- habéis
- había
- habíais
- habíamos
- habían
- habías
- hace
- haceis
- hacemos
- hacen
- hacer
- haces
- hacia
- hago
- han
- has
- hasta
- hay
- haya
- hayamos
- hayan
- hayas
- hayáis
- he
- hemos
- hube
- hubiera
- hubierais
- hubieran
- hubieras
- hubieron
- hubiese
- hubieseis
- hubiesen
- hubieses
- hubimos
- hubiste
- hubisteis
- hubiéramos
- hubiésemos
- hubo
- incluso
- intenta

- intentais
- intentamos
- intentan
- intentar
- intentas
- intento
- ir
- jamás
- junto
- juntos
- la
- largo
- las
- le
- les
- lo
- los
- mas
- me
- menos
- mi
- mia
- mias
- mientras
- mio
- mios
- mis
- misma
- mismas
- mismo
- mismos
- modo
- mucha
- muchas
- mucho
- muchos
- muchísima
- muchísimas
- muchísimo
- muchísimos
- muy
- más
- mí
- mía
- mías
- mío
- míos
- nada
- ni
- ningun
- ninguna
- ningunas
- ninguno
- ningunos
- no
- nos
- nosotras
- nosotros
- nuestra
- nuestras
- nuestro
- nuestros
- nunca
- o
- os
- otra
- otras
- otro
- otros
- para
- parecer
- pero
- poca
- pocas
- poco
- pocos
- podeis
- podemos
- poder
- podria
- podriais
- podriamos
- podrian
- podrias

- por
- por qué
- porque
- primero
- primero desde
- puede
- pueden
- puedo
- pues
- que
- querer
- quien
- quienes
- quienesquiera
- quienquiera
- quiza
- quizas
- quién
- qué
- sabe
- sabeis
- sabemos
- saben
- saber
- sabes
- se
- sea
- seamos
- sean
- seas
- segun
- sentid
- sentida
- sentidas
- sentido
- sentidos
- ser
- seremos
- será
- serán
- serás
- seré
- seréis
- sería
- seríais
- seríamos
- serían
- serías
- seáis
- si
- siempre
- siendo
- siente
- sin
- sino
- sintiendo
- so
- sobre
- sois
- solamente
- solo
- somos
- son
- soy
- sr
- sra
- sres
- sta
- su
- sus
- suya
- suyas
- suyo
- suyos
- sí
- sín
- tal
- tales
- tambien
- también
- tampoco
- tan
- tanta
- tantas

- tanto
- tantos
- te
- tendremos
- tendrá
- tendrán
- tendrás
- tendré
- tendréis
- tendría
- tendríais
- tendríamos
- tendrían
- tendrías
- tened
- teneis
- tenemos
- tener
- tenga
- tengamos
- tengan
- tengas
- tengo
- tengáis
- tenida
- tenidas
- tenido
- tenidos
- teniendo
- tenéis
- tenía
- teníais
- teníamos
- tenían
- tenías
- ti
- tiempo
- tiene
- tienen
- tienes
- toda
- todas
- todo
- todos
- tomar
- trabaja
- trabajais
- trabajamos
- trabajan
- trabajar
- trabajas
- trabajo
- tras
- tu
- tus
- tuve
- tuviera
- tuvierais
- tuvieran
- tuvieras
- tuvieron
- tuviese
- tuvieseis
- tuviesen
- tuvieses
- tuvimos
- tuviste
- tuvisteis
- tuviéramos
- tuviésemos
- tuvo
- tuya
- tuyas
- tuyo
- tuyos
- tú
- ultimo
- un
- una
- unas
- uno
- unos
- usa
- usais

- usamos
- usan
- usar
- usas
- uso
- usted
- ustedes
- va
- vais
- valor
- vamos
- van
- varias
- varios
- vaya
- verdad
- verdadera
- verdadero
- vosostras
- vosostros
- vosotras
- vosotros
- voy
- vuestra
- vuestras
- vuestro
- vuestros
- y
- ya
- yo
- él
- éramos

Glosario

API Una interfaz de programación de aplicaciones o API (del *inglés application programming interface*) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. 4, 12, 16, 17, 20, 34–36

Atom El Formato de Redifusión Atom es un archivo en formato XML usado para Redifusión web. 17

CSV Los archivos CSV (del inglés *comma-separated values*) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por coma. 48

DELETE Método de petición HTTP que borra el recurso especificado. 16, 17

F **Measure** En estadística, el *F* Measure es una medida de la precisión de un test. 40

GET Método de petición HTTP que pide una representación del recurso especificado. Por seguridad no debería ser usado por aplicaciones que causen efectos ya que transmite información a través de la URI agregando parámetros a la URL. 16

HTML HTML, siglas de *HyperText Markup Language* (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, asï¿½ como para complementar el texto con objetos tales como imágenes. 21

HTTP Hypertext Transfer Protocol o HTTP es el protocolo usado en cada transacción de la World Wide Web. 16, 21, 36

IP Una dirección IP es un número que identifica de manera lógica y jerárquica a una interfaz de un dispositivo (habitualmente una computadora) dentro de una red que utilice el protocolo IP (Internet Protocol), que corresponde al nivel de red del protocolo TCP/IP. 36

IR La recuperación de información, llamada en inglés *information retrieval* (IR), es la ciencia de la búsqueda de información en documentos, búsqueda dentro de los mismos, búsqueda de metadatos que describan documentos, o también la búsqueda en bases de datos relacionales, ya sea a través de internet, intranet, para textos, imágenes, sonido o datos de otras características, de manera pertinente y relevante. 4, 5

JSON JSON, acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML. 17, 34–36

POST Método de petición HTTP que somete los datos a que sean procesados para el recurso identificado. Los datos se incluirán en el cuerpo de la petición. Esto puede resultar en la creación de un nuevo recurso o de las actualizaciones de los recursos existentes o ambas cosas. 16, 17

REST La Transferencia de Estado Representacional (Representational State Transfer) o REST es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. 16, 20, 34

RSS RSS es una familia de formatos de fuentes web codificados en XML. Se utiliza para suministrar a suscriptores de información actualizada frecuentemente. 17

smartphone Un smartphone (teléfono inteligente en español) es un dispositivo electrónico que funciona como un teléfono móvil con características similares a las de un computador personal. 7

SVM Las máquinas de soporte vectorial o máquinas de vectores de soporte (*Support Vector Machines*, SVMs) son un conjunto de algoritmos de aprendizaje supervisado desarrollados por Vladimir Vapnik y su equipo en los laboratorios AT&T. 9

URL Uniform Resource Locator, más comúnmente denominado URL, es una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en Internet para su localización o identificación, como por ejemplo documentos textuales, imágenes, videos, presentaciones digitales, etc. 14, 17, 20, 23, 36, 51

XML XML, siglas en inglés de *eXtensible Markup Language* (lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). 16, 17, 21