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

HERRAMIENTA DE VISUALIZACIÓN Y AGRUPACIÓN DE IMÁGENES PARA TWITTER

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

JUAN IGNACIO VÁSQUEZ LAVAL

PROFESOR GUÍA:
BÁRBARA POBLETE LABRA

MIEMBROS DE LA COMISIÓN:
JOSE A. PINO URTUBIA
NELSON BALOIAN TATARYAN

SANTIAGO DE CHILE
JUNIO 2012

Resumen

Este informe detalla el proceso de investigación y desarrollo de un prototipo de aplicación *WEB* que, a partir de los mensajes de la red social *Twitter* que se muestran a un usuario, genera un *collage* con imágenes que representan visualmente la información.

Para lograr esto se aprovecha el hecho de que un gran número de los mensajes desplegados contienen un vínculo hacia alguna página WEB y a su vez, estas contienen una imagen relevante a la información transmitida, ya sea un logo corporativo o una ilustración. Utilizando la *API* de Twitter y una serie de librerías externas, se logra obtener estos mensajes junto con sus respectivas imágenes para generar su posterior procesamiento.

El contenido descargado es agrupado automáticamente haciendo uso de herramientas de minería de datos. El objetivo de este proceso es generar una agrupación por tópicos y una selección de mensajes representativos de cada grupo con la finalidad de hacer más simple la visualización de la información.

Por último, el prototipo generado es evaluado en cuanto a rendimiento y percepción por parte de los usuarios. Los resultados y sus conclusiones son discutidos en este informe.

Agradecimientos

Quisiera agradecer a mi familia, especialmente a mis padres, mis hermanos y mi abuela, quienes me han dado siempre su apoyo incondicional y han ayudado a formarme como persona.

Además, quisiera dar las gracias a mis amigos Claudia Allendes, Nancy Allendes, Fabián Bravo, Nicolás Carrasco, Pablo Hevia, Juan Silva y Carolina Vélez, quienes me han acompañado y apoyado en esta última etapa de mi vida y mis estudios.

Me gustaría también agradecer de manera especial a Raúl Amunátegui y Pablo Hevia, quienes de manera desinteresada me ayudaron en la corrección de este informe.

Por último, quisiera agradecer a Fernanda Berckhoff, Sebastián Carraha, Trinidad Cofré, José Ducci, Daniel Espinoza, Federico Evans, Felipe Fresno, Tomás Iglesias, Pablo Lavín, Maximiliano Narváez, Tomás Oyarzún, Carlos Tapia, Rafael Troncoso y Sebastián Volosky, mis amigos de la infancia y adolescencia quienes han estado siempre presentes y son un pilar fundamental en mi vida.

Índice General

Índice de figuras

Capítulo 1

Introducción

Las *redes sociales online* han pasado a ser una parte importante de la vida de muchas personas. Éstas partieron como una forma de socializar con amigos, compartir imágenes, conocer nuevas personas, etc. Hoy en día, las redes sociales van mucho más allá de esto. Éstas son utilizadas, junto con lo antes mencionado, para vender, organizar eventos, para darse a conocer y para informar, por sólo nombrar algunos de sus usos.

El gran potencial informativo que tienen *Facebook*[1] y *Twitter*[2], que son algunas de las redes más importantes, impresiona muchísimo. Los medios de comunicación escrita más importantes ofrecen la opción de dejar comentarios a sus noticias a través de Facebook, o de seguir sus canales en Twitter para mantenerse informados. Hay una gran cantidad de gente que participa en estos medios de información, periodistas renombrados y políticos, quienes lo hacen para mantenerse informados e informar de lo que está pasando en el mundo de manera casi instantánea.

Si nos centramos más específicamente en la plataforma Twitter, ésta es una red social de *microblogging* que permite a los usuarios registrados enviar un mensaje de un tamaño máximo de 140 caracteres. Dichos mensajes por defecto son públicos y pueden ser vistos por quienes sigan a dicho usuario. La capacidad de poder seguir lo que escriben los usuarios en "tiempo real" da a este formato de red social un carácter informativo sin precedentes. Pero esta plataforma sólo se intercambia texto y para hacer referencia a objetos externos, como URL e imágenes, se introduce un *enlace*. Muchas veces estos objetos externos expresan más

[1] **Facebook:** http://www.facebook.com
[2] **Twitter:** http://www.twitter.com

detalle sobre el suceso.

Una de las características que han hecho de Twitter un gran potencial de información es su integración con una gran cantidad de dispositivos móviles, más específicamente *smartphones*. Esto ha permitido que la gente pueda publicar mensajes desde cualquier lugar y en el momento en que ocurren los acontecimientos noticiosos. Lo que hace que la gente se mantenga informada de manera casi simultánea sobre eventos de importancia, lo cual permite que Twitter sea mucho más rápido que medios tradicionales de comunicación. Sin ir más lejos, para el terremoto del año 2010 en Chile, tanto las líneas de teléfono, como el suministro de electricidad permanecieron interrumpidos en muchos lugares, pero la conexión a *3G* mantuvo su funcionamiento. Por este motivo muchas personas utilizaron Twitter como medio para informarse acerca de qué había ocurrido y para comunicar el estado en el que se encontraban. En la Figura 1.1 se puede apreciar como Twitter es utilizado por medios periodísticos para informar lo que está ocurriendo.

Figura 1.1: Twitter versión WEB

Las características anteriores permiten que Twitter cuente con gran popularidad. Sin embargo debido a la gran sobrecarga de información actual es imposible para los usuarios mantenerse al día con todo lo que se publica. Es por esto que surge la necesidad de contar con un sistema que permita visualizar en forma compacta la información más relevante de esta red. Este es el punto que motiva este trabajo, en el cual se busca una forma preliminar

de organizar visualmente eventos publicados en Twitter. Debido a esto se propone la implementación de un prototipo de visualización de imágenes y organización de la información para Twitter.

Este informe que se desarrolla a continuación se divide en las siguientes partes: *Conceptos Importantes y Trabajo Relacionado*, en donde se explican los conceptos a tener en cuenta para la comprensión de este informe y donde además se expresa el trabajo investigativo previo al desarrollo de este proyecto; *Diseño*, en el cual se discuten las decisiones que se tomaron respecto a las tecnologías que se utilizaron en el desarrollo del proyecto además de una breve descripción de los módulos que forman parte de la aplicación;*Desarrollo*, capítulo que detalla como se llevó a cabo la implementación de la aplicación *Twitter Gráfico*; *Evaluación*, en donde se realizaron distintas pruebas para medir el desempeño de la aplicación en distintos aspectos y, por último, *Conclusión*, capítulo donde se comentan los resultados generales del desarrollo de este trabajo.

1.1. Propuesta de Trabajo de Título

Las redes sociales, en particular Twitter, tienen un potencial muy grande para mantener informadas a las personas. Esto se debe a la inmensa cantidad de usuarios que participan de ellas y el gran número de mensajes que son escritos a diario. Resulta imposible leer todos los mensajes publicados cada día, lo cual hace que la información sea muy volátil. Debido a esto, proponemos desarrollar una aplicación prototipo que muestre de manera resumida y gráfica los principales mensajes a los que está suscrito un usuario por día. Es importante notar que en esta memoria se propone implementar una primera aproximación al problema. Este prototipo ha sido pensado de forma modular, lo cual permitirá mejorar su funcionalidad a futuro.

En detalle, para desplegar un resumen gráfico de lo que ha ocurrido en el día, a partir de mensajes, se propone aprovechar el hecho de que muchos mensajes contienen enlaces a fuentes externas de información. Estos enlaces, además de informar con más detalle la noticia, contienen imágenes relacionadas con ella. En base a estas imágenes surge la idea de generar un collage o alguna otra manera de representación visual, que permita hacerse una idea rápida de qué eventos han marcado el desarrollo del día.

Los mensajes de Twitter son muy simples y carecen de cualquier *metainformación*, como por ejemplo categorías. Debido a esto se propone categorizar los mensajes automáticamente de acuerdo a ciertas características que poseen(por ejemplo, autor, de manera que se pueda distinguir entre fuentes periodísticas y amigos). Esto puede lograrse de varias maneras diferentes. Por ejemplo: hacer la distinción entre medios masivos de comunicación, por la cantidad de seguidores que tienen; hacer que el usuario categorice manualmente a quienes sigue en Twitter o hacer un sistema de votación donde entre varios usuarios deciden la categoría de quien escribe los mensajes. Esto facilitaría la lectura, dado que permitiría filtrar los mensajes por tipo de autor.

Nuevamente, ya que los mensajes carecen de clasificación, creemos que también sería útil implementar un sistema de etiquetas o "*tags*" relevantes que permitan categorizar los mensajes según sus contenidos. Para lograr esto, podemos analizar diferentes formas de clasificación, como por ejemplo: un *ranking* con las palabras más repetidas y éstas utilizarlas

como tags. Alternativamente se propone agrupar mensajes utilizando técnicas de similitud, como *clustering*, que agrupa automáticamente elementos.

Mostrar lo ocurrido en Twitter de manera gráfica es una solución interesante, desde el punto de vista de manejo de grandes volúmenes de información, ya que permitiría al usuario hacerse una idea rápida de lo que ha acontecido recientemente y lo que ocurre en la actualidad, permitiéndole revisar sólo los detalles de la información que le interesen realmente. El sistema que se propone permitirá mezclar información visual(imágenes) con clasificación automática. Este sistema generará un collage que muestre una imagen representativa por cada grupo y, al seleccionar el grupo que se desea ver, se desplegarán las demás ilustraciones pertenecientes a dicha clasificación.

Para tener una idea más clara de cómo funcionará el sistema, en la Figura 1.2 se muestra un diagrama con el flujo de información que se llevará a cabo para poder desplegar el resumen gráfico.

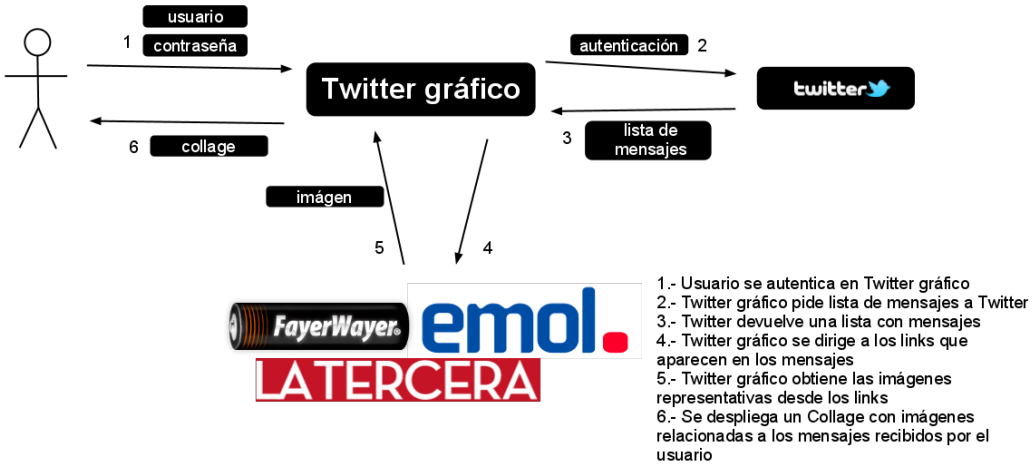

Figura 1.2: Diagrama de pasos que seguirá la aplicación

1.2. Objetivos

1.2.1. Objetivo General

Clasificar y mostrar de manera gráfica y resumida los contenidos obtenidos de una cuenta de Twitter o de una búsqueda en Twitter, dando extensibilidad hacia otras redes sociales.

1.2.2. Objetivos Específicos

- Clasificar los mensajes automáticamente según la procedencia, de manera tal que se puedan diferenciar los mensajes que vienen de fuentes periodísticas o de amigos.
- Implementar una estrategia de recolección de imágenes relacionadas al *timeline* recolectado del usuario. Esta solución debe ser eficiente y escalable.
- Agrupar mensajes con tópicos similares o que pertenezcan al mismo evento.
- Construir un resumen visual a partir de las imágenes representativas de cada conjunto de tópicos del timeline del usuario.
- Desplegar un collage visual sobre la base de las imágenes de cada tópico encontrado en el timeline del usuario. Las ilustraciones más predominantes representarán los temas más importantes.

Capítulo 2

Conceptos Importantes y Trabajo Relacionado

En este capítulo se presentan algunos conceptos importantes a tener en cuenta para el entendimiento de este informe. Además, se muestran algunas aplicaciones que sirvieron como referencia para realizar este trabajo.

2.1. Conceptos

3G Es la abreviación de tercera generación de transmisión de voz y datos a través de telefonía móvil.

Ajax Es una técnica de desarrollo web para crear aplicaciones interactivas

API(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. Una API puede incluir especificaciones para rutinas, estructuras de datos, objetos de clases y variables. [33]

Base de Datos Orientada a Documentos Es una base de datos donde cada entrada o registro puede tener un esquema de datos diferente, con atributos o "columnas" que no tienen por qué repetirse de un registro a otro.

Base de Datos Relacional Es una base de datos que cumple con el modelo relacional, el cual es el modelo más utilizado en la actualidad para implementar bases de datos ya planificadas. [10]

Clustering Es un procedimiento de agrupación de una serie de elementos de acuerdo con un criterio de cercanía. [25]

Cookie Las Cookies son pequeños archivos que se encuentran en el lado del cliente en los cuales el sitio WEB puede almacenar información. Esto se utiliza comúnmente para guardar información de la sesión. [32]

Crontab Crontab es un archivo que se encuentra en los sistemas operativos Linux, en el cual se definen procesos que deben ejecutarse en segundo plano en un cierto intervalo de tiempo. [36]

Decorador Un decorador es un patrón de diseño de software. Un decorador modifica dinámicamente la funcionalidad de una función, método o clase sin tener que usar directamente subclases o cambiar el código fuente de la funcion que es decorada. [30]

Enlace o Hiperenlace También llamado hipervínculo, es parte fundamental de la arquitectura de la World Wide Web, pero el concepto no se limita al HTML o a la Web. Casi cualquier medio electrónico puede emplear alguna forma de hiperenlace.

Expresión Regular En el área de la programación las expresiones regulares son un método por medio del cual se pueden realizar búsquedas dentro de cadenas de caracteres. [27]

Framework Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, con base a la cual otro proyecto de software puede ser más fácilmente organizado y desarrollado.

Hash Una función de hash es un algoritmo que mapea sets de datos grandes y con tamaños variables en sets de datos más pequeños y con un largo fijo. Al valor retornado por esta función se le llama hash. [31]

HTML(HyperText Markup Language) 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, así como para complementar el texto con objetos tales como imágenes. [24]

Metainformación Es, en palabras simples, la información que va más allá de lo que se puede ver. En el fondo es información de la información.

Microblogging También conocido como nanoblogging, es un servicio que permite a sus usuarios enviar y publicar mensajes breves. [23]

MVC(Modelo Vista Controlador) Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos. Este es comúnmente usado en aplicaciones WEB. [29]

OAuth OAuth es un protocolo abierto que permite la autorización segura de un API de manera estándar. [34]

Parsear Corresponde a la acción de analizar sintácticamente un elemento y descomponerlo en partes para poder procesarlo de manera más fácil. [20]

Redes Sociales Son estructuras sociales compuestas de grupos de personas, las cuales están conectadas por uno o varios tipos de relaciones, tales como amistad, parentesco, intereses comunes o que comparten conocimientos.

Smartphone Es un teléfono móvil construido sobre una plataforma informática móvil, el cual tiene más capacidad de computación avanzada y conectividad que un teléfono móvil estándar.

Stopwords Es el nombre que reciben (en inglés) las palabras sin significado como artículos, pronombres, preposiciones, etc. que son filtradas antes o después del procesamiento de datos en lenguaje natural (texto). [26]

Timeline Es la línea en la que se ven todos los tweets propios y de los usuarios a quienes se está siguiendo, en orden cronológico.

Tweet Es un mensaje escrito en la red social Twitter.

Ventana modal Una ventana modal es una ventana que se despliega en primer plano dentro de la aplicación para mostrar algún contenido de manera destacada. [35]

WSGI Define una interfaz simple y universal entre servidores WEB y aplicaciones WEB o framework para el lenguaje de programación Python. [28]

XML(Extensible Markup Language) Es un lenguaje de marcas que define una serie de reglas que le dan formato a un documento de manera que sea fácil de interpretar ya sea para una persona como para una máquina. [22]

XPATH Es un lenguaje que permite construir expresiones que recorren y procesan un documento XML. [21]

2.2. Trabajo Relacionado

En esta sección se presentan algunas aplicaciones que fueron tomadas en cuenta como alusión a lo que se quería desarrollar en este trabajo.

Se tomó como referencia para el desarrollo del despliegue de las imágenes una sección de fotografías de http://www.latercera.com (Figura 2.1), en la cual se muestra un collage de ilustraciones con las noticias del día, pero, a diferencia de éste, el collage se generó automáticamente utilizando los mensajes de Twitter de la línea de tiempo "timeline" como fuente.

Otro sitio que muestra una forma interesante de desplegar imágenes es http://www.ernestdelgado.com/public-tests/canvasphoto (Figura 2.2), el cual presenta una librería hecha en el lenguaje *javascript*, que hace posible desplegar imágenes obtenidas desde una cuenta de flikr de manera interactiva, pero dado que se buscaba un despliegue simple de las ilustraciones se descartó trabajar con esta librería. Además, dicha librería no era fácilmente adaptable para interactuar con otras librerías con las que se trabajó.

Además de estos sitios, se revisó una serie de publicaciones respecto a cómo agrupar textos y a cómo seleccionar un texto representativo dentro de un cierto grupo. De estas publicaciones, las que mejor definían lo que se pretendía hacer en el presente trabajo son: la publicación de Mor Naaman [12], en la cual se evalúa el rendimiento de distintos métodos de agrupación de texto y la publicación [13], del mismo autor, en la cual se evalúa el rendimiento de diversos métodos para seleccionar el texto más representativo dentro de un grupo de textos.

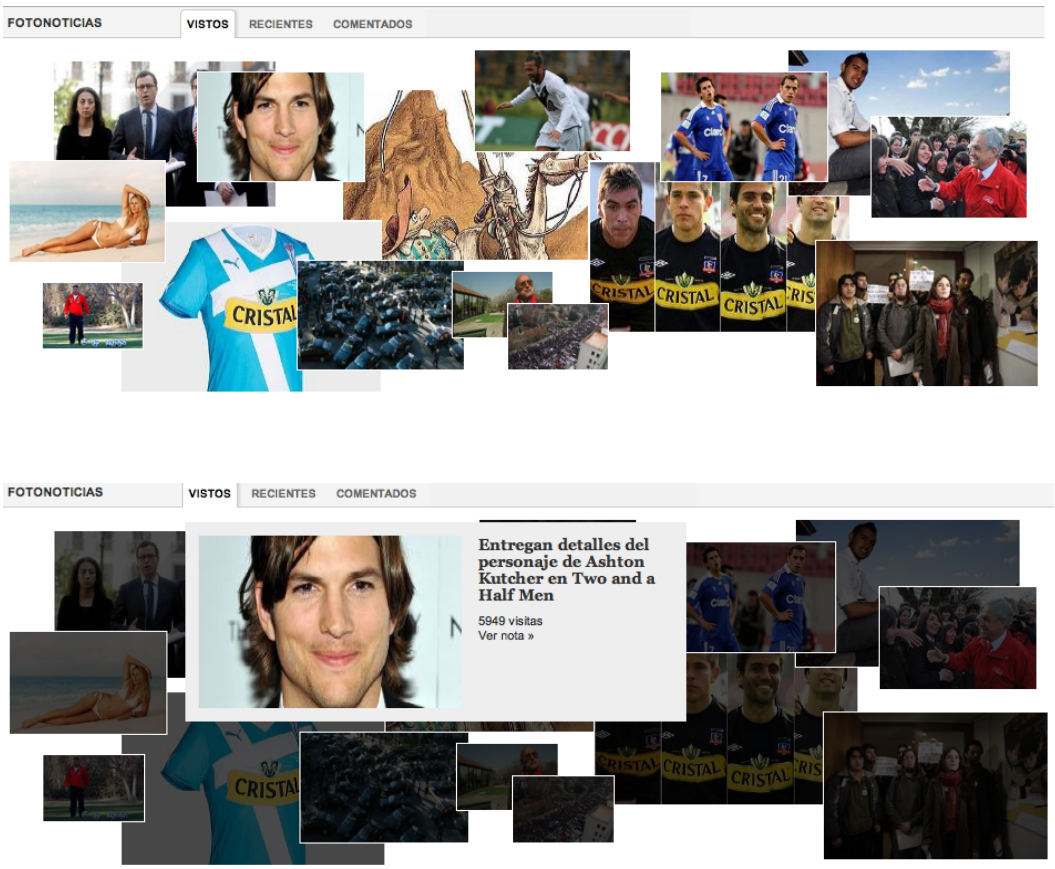

Figura 2.1: Imágenes de http://www.latercera.com

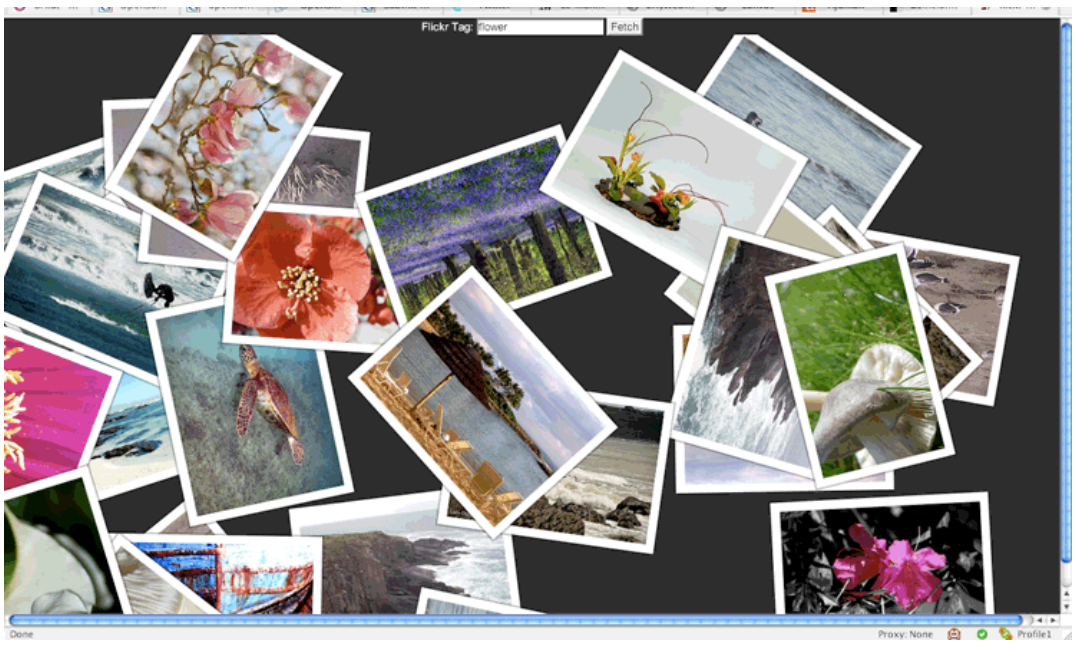

Figura 2.2: Canvas Photo Experiment

Capítulo 3

Diseño

3.1. Requerimientos Técnicos

En esta sección se discuten las necesidades técnicas para poder desarrollar la aplicación junto con las decisiones que se tomaron respecto a como abordar dichas necesidades. Se revisa en detalle los requerimientos de base de datos, cuál es el lenguaje de programación a utilizar, cómo se procesan los enlaces que aparecen en los mensajes, qué herramientas se utilizan en el despliegue de las imágenes, de qué manera se agrupan los mensajes y de qué manera se realiza la limpieza de los mensajes para poder procesarlos de mejor manera.

3.1.1. Requerimientos Base de Datos

Primero que todo, se realizó una investigación para saber si es necesario la utilización de una base de datos que guarde la información de las imágenes que se están tratando de obtener desde los mensajes de twitter o si éstas se pueden obtener en el momento en que se realiza la petición sin demasiada tardanza. Para esto se diseñó un módulo en python que obtiene los mensajes de twitter de un usuario, revisa si hay algún enlace en los mensajes, luego se conecta a los enlaces que encuentra y obtiene las imágenes relevantes desde dichos sitios. Dicho programa tarda 20 segundos aproximadamente en obtener imágenes de 10 sitios, lo que es mucho tiempo si es que se espera hacer esto a mayor escala. Es por esto que se decidió utilizar una base de datos que se encargue de ir almacenando las URL de las imágenes que va encontrando.

Para poder obtener los mensajes de un usuario se necesita que dicho usuario autorice a la aplicación para acceder a su información. Para esto Twitter utiliza el protocolo *Oauth*, que envía al usuario a un sitio de Twitter para que el usuario se autentifique y dé permisos a la

aplicación para acceder a sus datos, en este caso a sus mensajes.

A continuación se investigó sobre el tipo de base de datos por utilizar en la construcción del sistema propuesto. Las alternativas que se vieron fueron *bases de datos relacionales*, como *MySQL*[1] o *PostgreSQL*[2], o *bases de datos orientadas a documentos*, tales como *MongoDB*[3]. Dado que lo que se pretende guardar son documentos que contengan la información del mensaje junto con la información de la noticia, no es necesario utilizar una base de datos relacional. Además el alumno tiene bastante experiencia previa trabajando con MongoDB y es por esto que se optó por utilizar esta base de datos.

Los documentos en MongoDB no tienen una estructura fija, por lo que es difícil hablar de un modelo de datos. Dadas las necesidades del proyecto, cada documento en la base de datos tiene la siguiente información:

- Id_usuario (Id del usuario que escribió el mensaje)
- Usuario (usuario que escribió el mensaje)
- Fecha (fecha y hora en la que fue escrito el mensaje)
- Imagen (URL de la imagen contenida en el mensaje, si es que la hay)
- Mensaje (texto del *tweet*)
- Título (título que aparece en la descripción de la página apuntada por el tweet)
- Descripción (descripción que aparece en la URL del mensaje)
- URL (URL que aparece en el tweet).

Para encontrar el enlace dentro de un mensaje se utiliza una *expresión regular* obtenida de una fuente externa [9], la cual fue probada usando varios textos de prueba y no ha tenido problemas.

[1]**MySQL:** http://www.mysql.com/
[2]**PostgreSQL:** http://www.postgresql.org/
[3]**MongoDB:** http://www.mongodb.org/

3.1.2. Lenguaje de Programación

Se investigó qué lenguaje de programación es más adecuado para realizar este trabajo de título. Las alternativas que se vieron son: *PHP*, *Python*, *Java* y *Ruby*, por ser éstos, según experiencia del autor del trabajo, los lenguajes más utilizados para el desarrollo de aplicaciones WEB. Los aspectos que se utilizaron para comparar y elegir uno de éstos fueron: su simplicidad, que permita desarrollar de manera ágil, que tenga librerías que permitan trabajar de manera simple con la *API* de Twitter [2](las librerías disponibles se encuentran en [1]), que tengan librerías que faciliten el *parseo* de páginas WEB([3], [4], [5] y [6]) y que tenga buenas herramientas para trabajar con MongoDB.

El alumno no tiene experiencia trabajando con Ruby, por lo que se descartó trabajar con este lenguaje, dado que se trata de un proyecto que se desarrolla en corto plazo. Los demás lenguajes de programación tienen librerías bastante completas para trabajar con la API de Twitter(la librería de Python se encuentra en [11]) y además tienen librerías para trabajar con MongoDB, pero se consideró que Java era muy complejo para un proyecto de tan corto plazo. Por último se decidió adoptar Python para realizar el proyecto, dado que el alumno se encuentra trabajando actualmente en un proyecto WEB que involucra Python y MongoDB, por lo que tiene más experiencia trabajando con estas herramientas.

3.1.3. Procesamiento de URL

En cuanto al parseo del sitio WEB donde se encuentra la imagen, se tomaron en cuenta dos librerías de Python que facilitan esta tarea, las cuales son *minidom*[4] y *etree*[5]. El alumno tiene experiencia trabajando con ambas librerías y se decidió utilizar la segunda, porque ésta permite trabajar utilizando el lenguaje *XPath*, lo que facilita *parsear* documentos *HTML* o *XML* de manera muy sencilla.

[4]**Minidom:** http://wiki.python.org/moin/MiniDom
[5]**Etree:** http://lxml.de/tutorial.html

3.1.4. Despliegue de Imágenes

El despliegue de las imágenes se realiza a través un sitio WEB y no en un programa de escritorio, porque es más simple tanto para el programador como para el usuario, además porque se desarrolla de manera más ágil y porque el alumno tiene bastante experiencia trabajando en proyectos WEB. Para realizar este proyecto se investigaron distintos *frameworks* tales como *DJango*[6], *Pylons*[7] y *Bottle*[8]. Se descartó utilizar DJango porque es un framework demasiado grande para las proporciones del proyecto y además tiene muchas herramientas, tales como autenticación, manejo de roles y otras, que no se utilizan en este proyecto. Se decidió finalmente utilizar Bottle por sobre Pylons porque el alumno tiene experiencia trabajando con este framework, y Pylons no presenta ventajas por sobre Bottle en el desarrollo de este proyecto.

3.1.5. Clustering de Mensajes

Dado que la cantidad de mensajes es bastante grande, se implementó un módulo que se encarga de agrupar los mensajes por evento o tópico. Esta parte se desarrolla modularmente para que, en un trabajo futuro, esta función pueda ser reemplazada por otra para probar diferentes soluciones. Para lograr esto, se utilizó una herramienta externa llamada *Cluto*[18], la cual se encarga de agrupar textos de manera automática. Previamente al desarrollo del proyecto se estudió la posibilidad de implementar un método que se encargara de agrupar los mensajes, pero se optó por utilizar una herramienta externa para enfocar el trabajo en la obtención y despliegue de la información y no en la forma en que se agrupa un texto. Además, dicha implementación ya se encuentra desarrollada y estudiada ampliamente, por lo que no es algo novedoso. Sin embargo vale la pena explicar en qué consiste el algoritmo de agrupación que se pretendía implementar. Para lograr agrupar los mensajes, dicho algoritmo asigna funciones de similitud [17] al título del mensaje, la fecha y la descripción. En el caso de twitter se puede utilizar el mensaje en sí, la fecha y la descripción y título que aparecen en los metadatos del enlace contenido en el tweet. Además se debe tener lo que llamaremos *umbral de similitud*, lo cual corresponde al mínimo de similitud que debe tener un mensaje

[6] **DJango:** https://www.djangoproject.com
[7] **Pylons:** http://www.pylonsproject.org
[8] **Bottle:** http://bottlepy.org/docs/dev

con respecto a un grupo para poder pertenecer a éste. Para llevar a cabo la agrupación en *clusters*[9] comparamos cada uno de los documentos con los clusters existentes (se puede comparar con cada uno de los documentos o con la media del cluster). Si ninguno se encuentra dentro del umbral de similitud, entonces se crea un cluster nuevo o se agrega el documento al cluster con mayor similitud. Para obtener un buen resultado en la agrupación por eventos o tópicos, se deben hacer varias pruebas e ir variando el umbral de similitud hasta obtener una buena aproximación de los grupos deseados.

Teniendo agrupados los mensajes en clusters, se muestra un mensaje de calidad junto con su imagen, que represente dicho evento o tópico y, si el usuario lo desea, puede acceder a ver el resto de los mensajes con sus respectivas imágenes (si es que tienen). Para encontrar el mensaje más representativo del evento se desarrolló un módulo encargado de encontrar dicho mensaje. Este módulo debe filtrar los mensajes que contienen imágenes antes de elegir al más representativo y luego devolver los mensajes rankeados según su representatividad dentro del tópico. Se investigaron distintas opciones que fueron evaluadas en [13]. Pero finalmente se decidió implementar una solución que aprovecha la información adicional que entrega Cluto para asignarle un puntaje a las palabras más importantes de cada grupo y con esto asignar puntajes a cada mensaje.

Otra aproximación que se evaluó implementar para seleccionar al mensaje más representativo fue vista en [13]. Dicha solución toma al centroide del cluster como el mensaje más representativo, es decir, el mensaje que se encuentra más cerca del centro del cluster según la función de similitud nombrada previamente. Al rankear los mensajes se asignará puntaje tanto por el contenido (cercanía al centroide) como por la fuente de donde provenga (si tiene muchos seguidores se asignará más puntaje). Al ser desarrollado modularmente se abre la posibilidad de, en un trabajo futuro, probar distintas aproximaciones de este problema para resolver.

Otra solución más que se evaluó para escoger el mensaje más representativo dentro de un grupo, es tomar todas las palabras relevantes dentro de los textos de los tweets, los títulos

[9] **Cluster:** grupo.

y las descripciones que aparecen en la URL contenida en el mensaje, ignorando palabras monosílabas o muy comunes, conocidas como *stopwords*. Luego se asigna un puntaje a cada palabra según la frecuencia con que aparece dentro del grupo de mensajes. Después se le asigna un puntaje a cada mensaje, equivalente a la suma de los puntajes de las palabras contenidas en él. Por último, se normalizan los puntajes de los mensajes dividiendo por la cantidad de palabras que tienen (omitiendo las stopwords). De esta forma, el mensaje con mayor puntaje normalizado corresponde al mensaje con mayor relevancia dentro del grupo.

Al hacer pruebas de obtención de imágenes se encontró que algunas fuentes periodísticas tenían como imagen representativa su logo corporativo, lo que no aporta información visual respecto del evento ocurrido. Es por esto que se implementó una lista negra de imágenes que no aportan información. Para esto se tiene una lista con las URL de estas imágenes y al obtener un mensaje se contrasta la URL de su imagen con la lista y de encontrarse en ésta simplemente se omite dicha ilustración y se considera como un tweet sin imagen.

3.1.6. Limpieza de Imágenes

Se investigó la posibilidad de descartar imágenes que fueran muy similares entre sí utilizando librerías externas o API tales como *PixMatch* [14], que son capaces de encontrar porcentajes de similitud en imágenes. Se descartó utilizar librerías externas, porque para esto habría que almacenar la imagen en el servidor, lo que ocuparía muchos recursos de espacio y además tomaría mucho tiempo en descargar las imágenes desde cada uno de los sitios. Por otro lado se descartó utilizar la API de PixMatch en este proyecto, ya que es pagada, por lo que se propone como trabajo futuro encontrar imágenes similares para no mostrarlas en conjunto en la vista previa de los mensajes más relevantes, o, en el caso de que sean idénticas o con un porcentaje de similitud muy alto, simplemente descartarlas. Por otra parte también se tomó en cuenta comparar los *hash* MD5 de las imágenes para descartar imágenes idénticas, pero para esto habría que descargar todas las imágenes en el servidor, lo que tomaría mucho tiempo y podría afectar el rendimiento del sistema. Por lo demás, en las pruebas que se han hecho no se ha visto que se repitan imágenes de manera frecuente, más allá de los logos corporativos de algunos sitios, por lo que no vale la pena perder rendimiento para tratar de

descartar imágenes repetidas.

3.1.7. Evaluaciones

Por último, se puso en funcionamiento la aplicación en un servidor, con la intención de realizar pruebas de rendimiento y evaluaciones con usuarios. De esta manera se pudo comparar resultados para saber a qué es necesario brindarle más importancia dentro de los desarrollos futuros.

3.2. Módulos

Para entender de mejor manera cómo funcionará la aplicación, a continuación se muestra un pequeño diagrama con los módulos que formarán parte de la aplicación y una breve descripción de éstos (Figura 3.1).

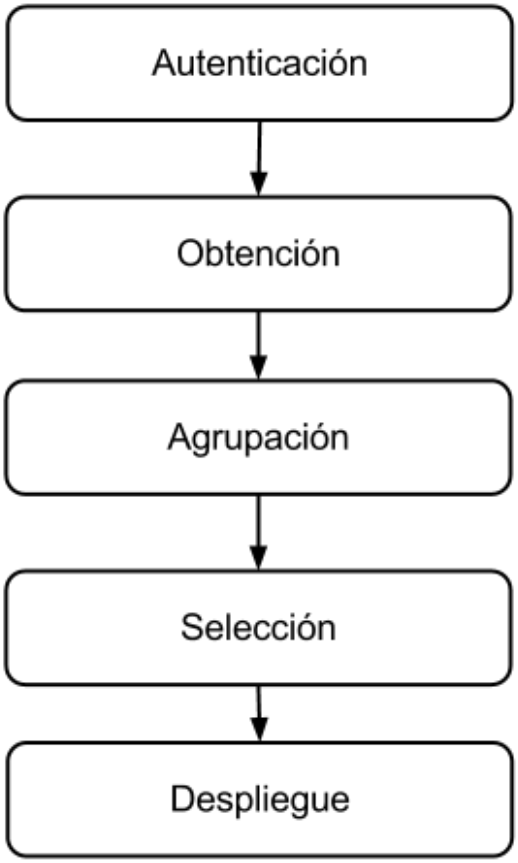

Figura 3.1: Diagrama de pasos que seguirá la Aplicación

3.2.1. Obtención

Este módulo se encarga de obtener los mensajes de Twitter y a su vez contiene submódulos encargados de obtener la URL del mensaje (de haber una) y otro encargado de obtener la imagen relevante dentro de esta URL junto con otra información relevante adjunta en ésta.

3.2.2. Agrupación

Éste está encargado de agrupar los mensajes por tópicos o eventos.

3.2.3. Selección

Este módulo es el encargado de ordenar los mensajes según su relevancia, la información que aporten al grupo y la fuente del mensaje. Además, selecciona un mensaje cuya imagen será la representativa del grupo, para ser desplegada por la aplicación.

3.2.4. Despliegue

Este módulo muestra el resumen gráfico en forma de nube de imágenes en su página principal y los mensajes correspondientes a cierto tópico, si es que el usuario desea ver los mensajes de manera más detallada.

Capítulo 4

Desarrollo

En este capítulo se explica como se llevó a cabo el desarrollo de la aplicación "*Twitter Gráfico*". Se revisa en detalle la implementación de cada uno de los módulos presentes en el programa y además se explica la función que realiza cada uno de estos dentro de la aplicación.

Para una mejor comprensión de la estructura de la presente aplicación, se presenta a continuación un diagrama general (figura 4.1) que ilustra el camino que siguen los llamados a los servicios externos a la aplicación.

4.1. Módulos

Como se dijo anteriormente, para facilitar la extensibilidad del programa, éste se desarrolló en distintos módulos de manera que sean modificados o reemplazados fácilmente. A continuación se describe el desarrollo de dichos módulos, y se realiza un análisis de éste.

4.1.1. Obtención

En esta sección se describe la función e implementación del módulo de *Obtención*. Para una mejor comprensión de éste se incluye un diagrama explicativo (figura 4.2).

Este módulo corresponde a lo que se conoce en computación como *ETL(Extract, transform and load)*[16]. ETL corresponde a un proceso en el desarrollo de un proyecto de software que se encarga de la extracción de datos, su transformación y luego la carga de éstos en la base de datos. En el caso de este módulo, éste se encarga de obtener los mensajes de Twitter,

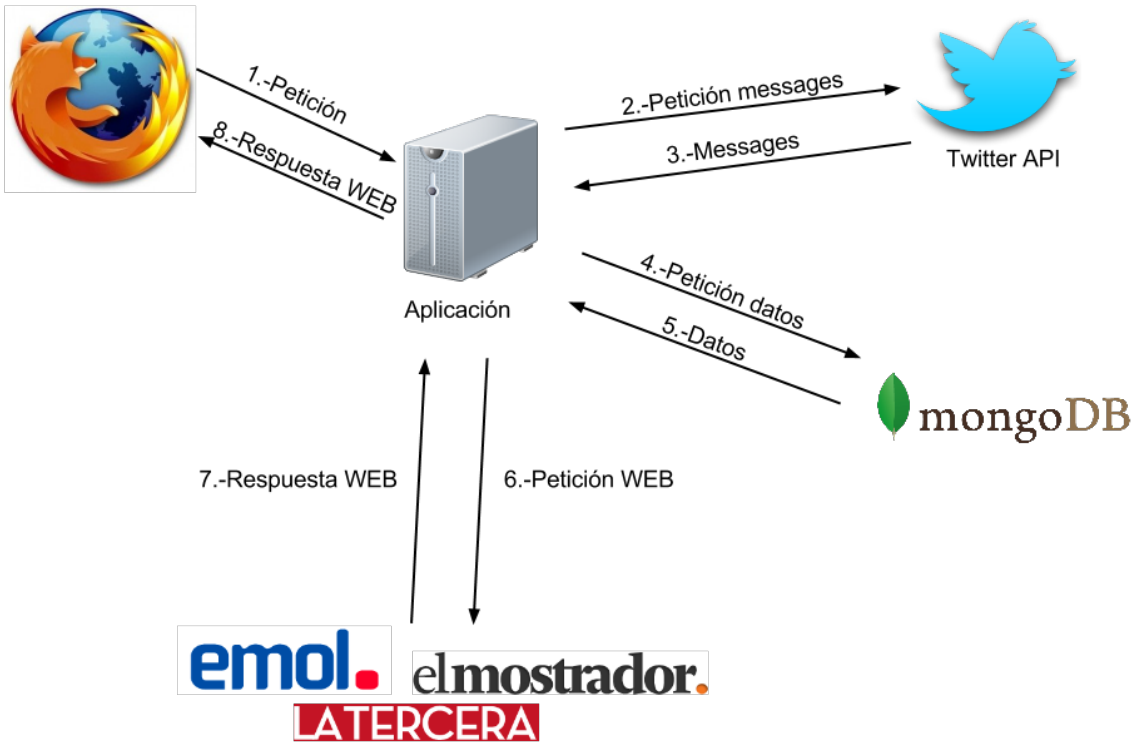

Figura 4.1: Diagrama de Peticiones a Servicios Externos

Figura 4.2: Diagrama módulo Obtención

obtener las URL de éstos, si es que tienen y, en dicho caso abrir la página WEB correspondiente a esa URL para obtener una imagen representativa junto con los metadatos que

pudieran ser necesarios. Obtención también cumple una segunda función, que es la de proveer un acceso simplificado a los datos, lo cual se logró utilizando lo que comunmente se conoce como *DAO(Data Access Object)*[15], el cual es un objeto que se encarga de suministrar una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos.

Para el desarrollo de este módulo se implementó una única clase llamada *Twitter_DAO*, la cual almacena la siguiente información de un tweet:

Id Identificador que le da Twitter a cada mensaje

User Nombre de usuario de quien escribió el tweet

Message Mensaje del tweet

URL URL que aparece en el tweet, si es que la hay; si no hay, es None

Image URL de donde se encuentra la imagen representativa del tweet

Keywords Palabras clave que aparecen en la página WEB relacionada con el tweet

Text Texto obtenido desde el enlace de la página WEB correspondiente al tweet

Massive Marca si el mensaje proviene de una fuente masiva de mensajes

Created_at Corresponde a un timestamp que marca el momento en que el tweet fue agregado a la base de datos

Anteriormente se pretendía guardar también el título y la descripción que aparecen en los metadatos de la página WEB pero se descartó esto dado que, haciendo pruebas, se vio que en la mayoría de los sitios éstos hacen referencia al sitio que publica la noticia y no a la noticia en sí; por lo tanto, no vale la pena guardarlos.

El constructor del objeto *Twitter_DAO* recibe un objeto de tipo *Status*, definido en la librería Python-Twitter; la dirección del *Host* donde se encuentra la base de datos y el nombre de la base de datos que está siendo usada. Luego, éste obtiene la URL asociada al tweet desde el mensaje utilizando una expresión regular y posteriormente revisa en la base de datos si

el objeto existía anteriormente. De ser así, simplemente se almacenan: la imagen, el texto y los *keywords* en variables dentro del objeto. Si el tweet no se encontraba en la base de datos, entonces, si existe una URL, éste ingresa a dicha dirección y trata de obtener los keywords, el texto y la imagen representativa de dicha URL. Para lograr esto se utilizó una librería externa llamada *lxml*, que permite parsear HTML o XML usando el lenguaje XPATH, el cual está diseñado para realizar esto de manera sencilla. Las palabras clave dentro de una página WEB se encuentran, si es que el sitio lo tiene implementado, en los encabezados o headers de la página en un tag de la forma que se aprecia en 4.3. La imagen en cambio puede aparecer de dos maneras distintas en los metadatos de la página [7]: la primera es en un tag de metadatos como el que se muestra en la Figura 4.4 (Este protocolo es descrito en [8]), y la segunda tiene una estructura como la que se ve en 4.5. Existen otras formas en las que pueden aparecer los keywords y las imágenes, pero éstas son las más comunes. Por último, para obtener el texto dentro de la página WEB se está obteniendo todo el texto que se encuentra dentro de los tag '*p*', los cuales encierran un párrafo y además se está obteniendo el texto que se encuentra dentro de los tag '*strong*', que son textos que se encuentran destacados.

```
1 <meta name="Keywords" content="palabras, separadas, por, comas"/>
```

Figura 4.3: Keywords en el encabezado

```
1 <meta property="og:image" content="URL de la imagen"/>
```

Figura 4.4: Keywords en el encabezado

```
1 <link rel="image_src" href="URL de la imagen"/>
```

Figura 4.5: Keywords en el encabezado

A continuación se explica de forma breve la funcionalidad de cada uno de los métodos que forman parte de este módulo:

get_twitter_messages Recibe una cuenta de usuario de tipo *Account*(descrito posteriormente) y el límite de mensajes que se desea obtener y devuelve una lista de objetos del

tipo *Twitter_DAO* con todos los datos descritos anteriormente.

save Inserta el objeto en la base de datos o lo actualiza si es que éste existía anteriormente.

get_messages_without_login Recibe un usuario de tipo *User*(descrito posteriormente) y el límite de mensajes que se desea obtener y devuelve una lista de objetos del tipo *Twitter_DAO* con todos los datos descritos anteriormente.

get_url Es una función auxiliar que recibe un texto y devuelve una URL, si es que hay alguna en el texto.

expire_tweets_before_date Esta función recibe una fecha y borra de la base de datos los tweets que fueron creados antes que dicha fecha.

get_representative_image_keywords Esta función recibe una URL, ingresa a dicha dirección y obtiene la imagen representativa, las palabras clave y el texto de dicha dirección.

check_massive_users Esta función revisa si el usuario que escribió el mensaje tiene muchos seguidores; si es así, entonces marca el mensaje como masivo.

4.1.2. Agrupación

En esta sección se describe el funcionamiento y desarrollo del módulo de *Agrupación*. Para una mejor comprensión de éste se incluye un diagrama explicativo (figura 4.6).

Este módulo se encarga de agrupar los tweets obtenidos con el módulo de obtención, de manera que los tweets que hacen referencia al mismo suceso o temas similares queden en el mismo grupo.

Para lograr esto, como se mencionó anteriormente, se decidió utilizar una aplicación de clustering llamada Cluto. A dicha herramienta se le debe pasar una matriz, la cual es generada con un script complementario a éste llamado *doc2mat*. Éste recibe un archivo con textos separados por líneas y genera un archivo que contiene dicha matriz o vector. La matriz contiene información acerca de los mensajes que se desea agrupar. Este archivo es leído y procesado más tarde por Cluto. Luego de esto se ejecuta la herramienta, la cual genera un archivo con la identificación del cluster al que pertenece cada tweet. Cluto además entrega

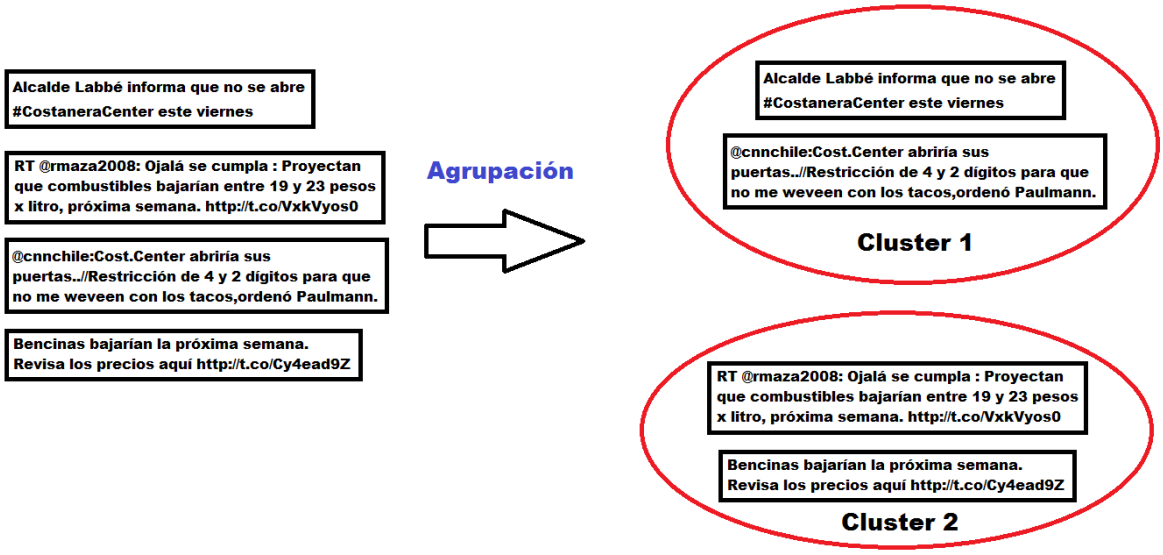

Figura 4.6: Diagrama módulo Agrupación

información acerca de las palabras más representativas de cada cluster, lo cual es utilizado después por el *Seleccionador* para elegir cuál es el tweet más representativo de cada cluster.

El módulo de *Agrupación* está implementado en un archivo llamado *agrupación.py*, el cual contiene una clase llamada *Clustering*. Dicha clase recibe en su constructor una lista con los tweets que se desea agrupar, el usuario que está utilizando la aplicación y de manera opcional puede recibir además la fuente que se quiere usar para agrupar los tweets. Esta fuente puede ser el mensaje de twitter, las palabras claves que aparecen en la URL o el texto que aparece en la página WEB y, por último, se puede seleccionar modelo de fila que se desea que utilice Cluto. Con respecto a la fuente que se utiliza para agrupar los mensajes, si se selecciona la opción de agrupar los mensajes usando para esto las palabras claves del texto de la página WEB correspondiente, entonces se utilizará dicha fuente con tweets que la tengan disponible. Para los que no tengan dicha fuente, se utilizará el mensaje para este fin.

Durante una etapa de pruebas se pudo notar que los mensajes no estaban siendo bien agrupados. Esto se debía a que el texto que estaba siendo utilizado para dicho propósito no se estaba limpiando previamente y por esta razón agrupaba los mensajes que contenían una URL, o los que contenían un tag de cierto usuario específico o ciertas palabras sin relevancia

en el texto llamadas comúnmente stopwords. Por esto se decidió realizar una limpieza de los datos por utilizar, previo a la realización del clustering. El proceso de purificación de los datos consiste en lo siguiente: primero se verifica si el texto contiene una URL; de ser así, se elimina ésta. Luego se remueven los tags de usuarios que pueden estar presentes en los mensajes de Twitter, es decir, se eliminan todas aquellas palabras que comienzan con '@'. Después se pasan los textos a minúsculas. Esto no tiene relevancia para el proceso de clustering en sí, sino más bien sirve para eliminar los stopwords que puedan comenzar con mayúscula. Luego de esto, se eliminan los tildes y caracteres no alfanuméricos tanto de los stopwords como del texto que se utilizará. Esto es importante, porque en el texto podría estar escrito una stopword sin tilde y al filtrar no se removería esta palabra, y la otra razón para eliminar los tildes y caracteres no alfanuméricos es que Cluto corta las palabras que contienen tildes de cualquier tipo. Por último, se eliminan todas las stopwords.

El módulo de *Agrupación* funciona de la siguiente manera:

1. Se limpia cada uno de los textos que serán utilizados para agrupar los mensajes e imágenes para que Cluto utilice únicamente las palabras que agregan valor a los mensajes, tal como fue explicado anteriormente.

2. Se genera un archivo que contiene los textos, separados por líneas. Éste se utilizará para agrupar los tweets. El nombre de dicho archivo corresponde al nombre de usuario que está utilizando la aplicación. Este archivo queda guardado en una carpeta temporal.

3. Utilizando el archivo anterior, se ejecuta la aplicación doc2mat para generar una matriz que luego será utilizada por Cluto. Dicha matriz queda guardada en la misma carpeta temporal bajo el nombre *(nombre de usuario).mat*.

4. Luego se ejecuta Cluto. Esta aplicación recibe como datos: el modelo de fila, el archivo que contiene la matriz y el número de clusters que se desea generar. Esto genera un archivo llamado *(nombre de usuario).mat.clustering.(numero de clusters)* el cual es un archivo que contiene un número en cada línea. Dicho número corresponde al grupo al que pertenece el texto que se encontraba en dicha línea. Además se genera un archivo

que contiene información adicional respecto a los clusters, tal como la relevancia de cada palabra dentro de cada cluster.

5. Por último, el módulo utiliza el archivo que contiene los números de cluster para agrupar los mensajes.

Una limitación que tiene Cluto es que se debe escoger la cantidad de grupos que se desea crear: no es posible dejar que la aplicación escoja el número de clusters más apropiado. Se propone como trabajo futuro implementar nuevamente este módulo utilizando alguna otra herramienta que escoja el número de clusters más apropiado dados los textos que se le entregan o implementar un algoritmo de agrupación que sirva para este propósito. Una estrategia conocida que puede ser implementada es la siguiente:

Dada una función de similitud de textos, tales como *similitud coseno TF-IDF*, *similitud de Monge-Elkan* o *similitud de q-grams* [17], por nombrar algunas, una lista de textos a agrupar y un margen de similitud. Se toma el primer texto de la lista. Como aún no existen grupos, entonces se crea uno donde su único elemento por ahora es este texto. Luego para cada otro elemento de la lista se compara su similitud respecto a cada cluster existente y, si algún cluster se encuentra dentro del margen de similitud, entonces dicho elemento es agregado al grupo más cercano (con el cual tenga más similitud, según la función de similitud). En caso de que ningún cluster se encuentre dentro del margen de similitud, entonces se crea un nuevo grupo y se agrega el elemento en cuestión. Al finalizar de recorrer todos los elementos se recomienda volver a realizar la comparación, dado que algunos elementos que pertenecen a algún cluster podrían ser más cercanos a otro cluster. Luego de repetir las comparaciones un número de veces significativo converge a una solución. Otro detalle que hay que tener en cuenta es que la función de similitud sirve para comparar dos textos, y no un texto con un cluster, por lo que hay que definir cómo se pretende comparar un texto con un cluster. Vale la pena revisar la publicación[12] donde se evalúan las diferentes formas de comparar textos.

A continuación se detalla la funcionalidad de cada uno de los métodos presentes en el módulo de *Agrupación*:

generate_documents_file Este método genera un archivo que contiene los textos que serán

utilizados para agrupar los tweets, recibiendo de manera opcional la fuente que se desea utilizar. Por defecto se utiliza el mensaje del tweet como origen del texto, pero en caso de que se seleccione una fuente distinta se intenta utilizar dicha procedencia de texto. Si el tweet carece de dicha fuente entonces se utiliza el mensaje del tweet para este propósito.

generate_mat_file Aquí se ejecuta el script doc2mat pasándole como parámetro el archivo a transformar.

generate_cluster_file Recibe el nombre del archivo que contiene la matriz junto con la cantidad de clusters que se desea generar y el modelo de filas. Con esto, ejecuta Cluto y devuelve el nombre del archivo que contiene los grupos a los que pertenece cada tweet.

generate_clusters Esta función se encarga de generar una lista de clusters, donde un cluster no es más que una lista de tweets. Para esto, utiliza el archivo generado por Cluto, que indica a qué cluster pertenece cada tweet.

clean_text Es una función que se encarga de limpiar un texto dado. Para esto, primero elimina la URL que pudiese estar contenida en el texto, la cual es recibida como parámetro; luego remueve los tags de usuarios, después pasa a minúsculas el texto, a continuación elimina las tildes y otros caracteres no alfanuméricos y, por último, se eliminan las stopwords del texto. El orden en el que se realizan estas acciones es importante, dado que la remoción de los tags de usuario y de la URL dependen de que los caracteres alfanuméricos no hayan sido removidos aún.

elimina_tildes Éste se encarga de eliminar los caracteres no alfanuméricos de un texto dado; los caracteres con tildes los deja sin éstas.

remove_user_tags Se encarga de remover todos los tags de usuarios que se encuentran en el texto, es decir, las palabras que comienzan con un '@'.

get_stopwords Este método entrega un arreglo con todos las stopwords que se desea eliminar de los textos. Éstos están en el archivo *stopwords.txt* que se encuentra en la carpeta *twittergrafico*. Por ahora sólo contiene stopwords en español, pero si se quisiera incluir palabras en otro idioma basta agregarlas separadas por espacios en dicho archivo.

4.1.3. Selección

En esta sección se detalla la implementación y función del módulo de *Selección*. Para una mejor comprensión de éste se incluye un diagrama explicativo (figura 4.7).

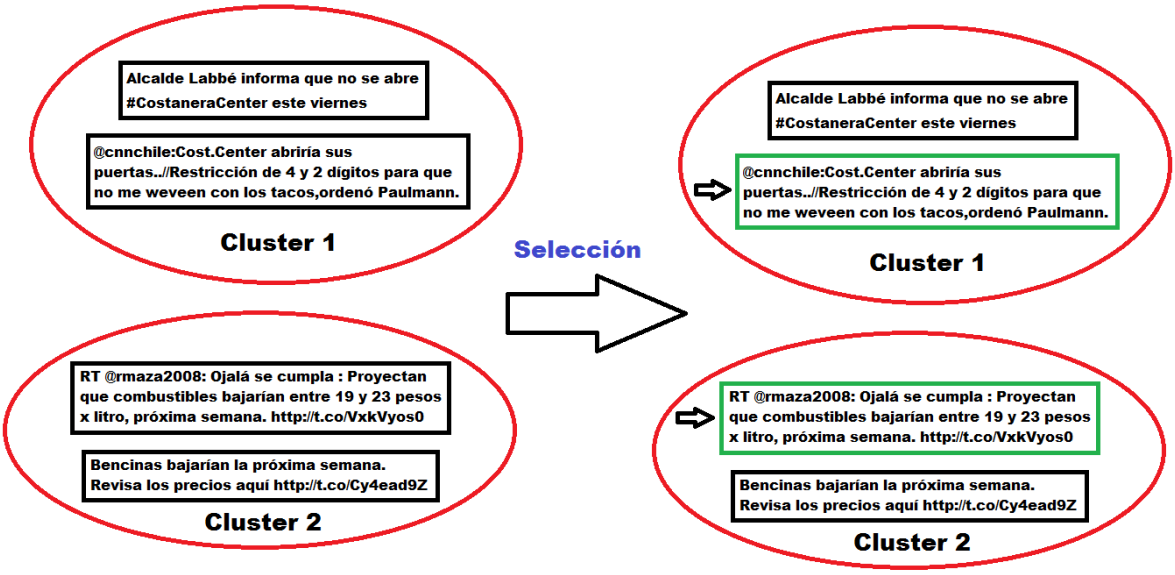

Figura 4.7: Diagrama módulo Selección

Este módulo se encarga de seleccionar, dentro de un cluster, el tweet más representativo, con la idea de mostrar únicamente las imágenes de los mensajes seleccionados y al hacer click sobre éstos se muestre el resto de las imágenes.

Para realizar esto, este módulo recibe los clusters de tweets que fueron previamente agrupados por el módulo de clustering. Además recibe el archivo que generó Cluto con información adicional respecto a estos grupos y, por último, la fuente de donde fue sacado el texto. Con esto el módulo sigue la siguiente estrategia para elegir el mensaje que represente de mejor manera al cluster:

1. El módulo lee el archivo con la información adicional respecto a los clusters y obtiene el porcentaje de importancia que Cluto da a las palabras que aparecen en cada cluster.

2. Teniendo los porcentajes de importancia de las palabras, se asigna a cada tweet un puntaje por cada palabra contenida en él, que aparezca en el listado de palabras de Cluto. El puntaje asignado corresponde al porcentaje que le asigna Cluto a dicha palabra multiplicado por 100.

3. Los puntajes de todos los tweets son normalizados al concluir la asignación de puntajes, es decir, el puntaje se divide por la cantidad de palabras que tiene el mensaje.

4. Se filtran los mensajes que no tienen imagen, y se les asigna puntaje 0.

5. Se selecciona, para cada cluster, el mensaje que tiene el mayor puntaje.

Se decidió multiplicar el puntaje asignado por Cluto por 100 dado que, luego al normalizar, se pierde información porque, al ser números muy pequeños, al normalizarlos no es extraño que Python aproxime estos números y puede pasar que quede más de un tweet con el mismo puntaje. Además, es conveniente normalizar los puntajes de los tweets, dado que éstos pueden tener distintos largos y, por lo tanto, un mensaje más largo es muy probable que contenga más palabras importantes que un mensaje más corto y, por esto, al normalizar, se le da más importancia al contenido del mensaje que a la extensión de éste, aunque esto puede hacer que se asigne un puntaje muy alto a tweets que contengan una única palabra o muy pocas palabras, pero cuya única palabra sea una palabra importante dentro del cluster. En la práctica no ha ocurrido que se elija un mensaje muy corto para representar al Tweet.

Dado que este trabajo de título está enfocado en representar de manera gráfica lo que ocurre en Twitter u otras redes sociales, no es de real importancia para este propósito que el mensaje que se muestra sea el más representativo del grupo de mensajes que se muestran, sino más bien es de gran importancia que el mensaje por mostrar tenga una imagen, y es por esta razón que se da mayor prioridad a los mensajes que contienen una ilustración para que estén a la vista en la página WEB representando al resto de los mensajes del cluster.

En el trabajo previo al desarrollo de la aplicación, se planteó la posibilidad de implementar una manera distinta de asignar el puntaje a los mensajes del cluster, la cual consistía en recopilar la frecuencia de todas las palabras que se encontraran dentro de un cluster, excluyendo las stopwords; sobre la base de estas frecuencias asignar puntajes a dichas palabras, y con

estos puntajes luego se puede asignar dicho puntaje a los mensajes en cuanto estas palabras estén contenidas en él. Pero en el transcurso del desarrollo del presente trabajo quedó de manifiesto que la importancia del mensaje que se despliega no radica en el contenido del mensaje, sino más bien en la imagen que se desea mostrar, y es por esta razón que se tomó la determinación de no implementar este procedimiento y, en vez de esto, dejarlo propuesto para un trabajo futuro en este mismo proyecto o uno paralelo.

Para el presente trabajo se consideró hacer un análisis de las imágenes que representan a los tweets con la intención de mostrar alguna imagen que no sea muy pequeña, pero se descartó esta idea por el hecho de que para realizar esto sería necesario descargar las imágenes que serán desplegadas y de esta manera poder ver el tamaño de éstas o realizar cualquier otro análisis; pero hacer esto utilizaría mucho ancho de banda por el lado del servidor y haría que la aplicación no fuera escalable a un nivel medianamente grande, por lo que se descartó dicha idea.

Este módulo está compuesto por dos clases. La primera corresponde al seleccionador, la cual es la clase principal, y la segunda es el filtro, que se encarga de revisar que los tweet contengan una imagen y de esta manera evitar que un mensaje sin imagen represente al cluster. A continuación se describe brevemente la funcionalidad que tiene cada método presente en el módulo:

obtain_representative_tweets Este método recibe los clusters de mensajes y los porcentajes de importancia que tienen las palabras dentro de un tweet, y dado esto devuelve una lista con los mensajes representativos de cada cluster.

parse_cluster_info La finalidad de este método es leer el archivo con la información de los clusters para obtener los porcentajes de importancia de las palabras dentro de cada uno de éstos, y luego retorna los respectivos porcentajes agrupados por cluster.

get_representative_tweet Ésta es una función auxiliar utilizada por el método *obtain_representative_tweets*, que toma un cluster de mensajes y los porcentajes de importancia de las palabras, y con esto asigna los puntajes a cada mensaje, luego los filtra

utilizando el método *image_filter* de la clase *Filter* y por último devuelve el mensaje que tenga el mayor puntaje.

normalize Este método se encarga de normalizar los puntajes de los mensajes, es decir, divide el puntaje de un tweet por la cantidad de palabras que tiene.

get_cluster_number Éste es un método auxiliar que, dada una linea de texto donde aparece el número de cluster, aplica una expresión regular para encontrar el número de cluster dentro del texto y lo retorna.

image_filter Éste se encarga de filtrar los métodos según tengan imagen o no la tengan. Además, revisa si la imagen que tiene se encuentra o no dentro de una lista de imágenes excluidas comparando la URL de ésta contra una lista negra.

4.1.4. Despliegue

En esta sección se explica cómo está implementado y qué función tiene dentro de la aplicación el módulo de *Despliegue*.

El módulo de despliegue corresponde al servicio WEB que se encarga de mostrar gráficamente el resultado de lo procesado en el resto de los módulos, y es por lo tanto el componente estructural de la aplicación, ya que cohesiona las funciones de los demás módulos presentes en la aplicación.

Para el desarrollo de éste se decidió utilizar un framework llamado *Bottle*, el cual es un framework liviano y rápido, que utiliza una interfaz simple y universal llamada *WSGI*, la cual sirve para interactuar con algún servidor WEB que la soporte, como, por ejemplo, *Apache*. Bottle, a diferencia de otros frameworks, no tiene una estructura *MVC(Model View Controller)*, sino ofrece una forma sencilla para definir las rutas del servicio WEB y ejecutar el controlador correspondiente a dicha ruta. Además ofrece soporte para diferentes motores de procesamiento de plantillas HTML, los cuales facilitan la generación de las vistas de la aplicación, tales como *Mako* [38]. A diferencia de otros frameworks, Bottle no ofrece herramientas que faliciten la interacción con la base de datos, pero, dado que los datos de los tweets son obtenidos de la API de Twitter, de la base de datos y de páginas WEB externas, de todas

maneras es necesario implementar un módulo propio para interactuar con estos datos.

El desarrollo de este componente está dividido en tres partes. La primera corresponde a la estructura de la aplicación, que se encuentra implementada en el archivo llamado *service.py* y que se encarga de implementar las acciones de los controladores, así como de dejar el servicio WEB esperando peticiones. La segunda parte de este componente corresponde a las plantillas HTML que serán cargadas por los controladores. Y, por último, la tercera parte corresponde a archivos estáticos, tales como *hojas de estilo CSS*, imágenes y archivos de código javascript.

Controladores

En cuanto a los controladores, cada uno se define en una función diferente, la cual debe estar precedida por un *decorador* llamado *route*, que pertenece al framework Bottle. El decorador recibe como parámetro la ruta que debe ser llamada por el explorador para que el controlador correspondiente sea ejecutado; además, opcionalmente, puede recibir el método utilizado por la petición. En el caso de este proyecto sólo se utilizan los métodos *GET* y *POST*. Cada uno de los controladores se encarga de obtener y procesar los datos necesarios para desplegar la vista correspondiente a dicho controlador.

Junto con la implementación de los controladores se creó un archivo llamado *lib.py*, el cual contiene métodos auxiliares que agrupan funcionalidades comunes de los controladores.

El primero de estos métodos es *render*, el cual es utilizado en caso de que un controlador necesite mostrar alguna plantilla en particular. Este método recibe como parámetro la ruta en la que se encuentra la plantilla, junto con las variables que el controlador debe pasarle a ésta. Luego render se encarga de introducir las variables que son globales para todas las páginas del sitio. Actualmente se agregan únicamente los datos del usuario que tiene su sesión iniciada en el sitio, pero eventualmente se pueden agregar más variables globales. Luego de esto, *render* se encarga de ejecutar Mako, el cual es una librería externa que se encarga de procesar los templates para generar el código HTML.

El segundo método auxiliar se llama *auth*, el cual es un decorador encargado de conceder

o denegar acceso a las páginas antes de la ejecución del controlador. Este decorador simplemente revisa que el usuario se encuentre con su sesión iniciada en el sitio. En caso de que esto suceda ejecuta el controlador correspondiente, pasándole como parámetro los datos del usuario que se encuentra accediendo al sitio. Lo anterior se hace en un método auxiliar y no en el mismo controlador, para evitar repetir el mismo código en varias partes. Además, en caso de que cambie la forma en que se trabajan las sesiones dentro del sitio se hace más sencillo cambiarlo.

La tercera función auxiliar es *passhash*, la cual introduce un hash a la contraseña, agregándole una sal, es decir, se toma la contraseña puesta por el usuario y se le agrega un texto y luego se aplica la función de hash, de manera que ésta sea más segura y difícil de descifrar. Luego, la función devuelve la contraseña codificada.

A continuación se detalla la funcionalidad de cada uno de los controladores presentes en la aplicación:

static Este controlador recibe como parámetro una ruta a un archivo estático, relativa a la carpeta donde se encuentran todos los archivos estáticos. Luego retorna dicho archivo.

home El controlador home se encarga de entregar la plantilla de la página de inicio, la cual es una página completamente estática, excepto por el hecho de que contiene, al igual que el resto de las páginas, la información de la sesión que se encuentra iniciada.

list Este controlador se encarga de obtener los últimos 50 mensajes de Twitter del usuario que tiene su sesión activa en la aplicación. Luego, éste filtra los mensajes que contienen una imagen y llama a *render*, pasándole la plantilla correspondiente junto con la lista de mensajes.

clustered Éste es llamado cuando el navegador realiza una petición de '*clustered_tweets*'. Al ejecutarse este controlador, éste obtiene los últimos 50 mensajes de Twitter del usuario que se encuentra conectado, luego utiliza el módulo de clustering para agrupar dichos mensajes y finalmente llama al módulo de selección para que seleccione los tweets representativos de cada grupo. Al concluir todo esto, éste llama a render para procesar

el template correspondiente, pasándole como parámetro los clusters y la lista con los tweets más relevantes de cada grupo.

keywords El controlador keywords es ejecutado cuando llega una petición desde el navegador de '*clustered_keywords*'. Éste realiza lo mismo que el controlador *clustered*, pero en vez de realizar el clustering utilizando los mensajes de Twitter, utiliza las keywords obtenidos desde la URL correspondiente al tweet, en caso de existir dichas keywords.

maxtf Este controlador realiza lo mismo que el controlador keywords, pero con la diferencia de que utiliza un modelo de filas en el proceso de clustering, que las escala de manera relativa a la frecuencia de sus términos.

search Este controlador hace lo mismo que el controlador de la lista, solo que obtiene los resultados de una búsqueda realizada utilizando la API de Twitter. Se han hecho algunas pruebas y se vio que los resultados obtenidos no son iguales a los obtenidos buscando en la página de Twitter. No se seguirá avanzando en este punto, dado que se aleja del enfoque que se quiere dar a este trabajo de título.

register Éste se encarga de desplegar la página de registro de cuentas.

register_account Este controlador se encarga de recibir los datos del registro de una cuenta en el sitio y realiza una pequeña confirmación de que los datos sean correctos, es decir, que ambas contraseñas sean iguales y que el largo de la contraseña sea mayor a 6 caracteres. Luego inserta el nuevo usuario en la base de datos y realiza algunos pasos que forman parte del proceso de autenticación y obtención de permisos, el cual será explicado más adelante, dado que es más complejo y comprende más de un controlador. Por último, el controlador redirige hacia una dirección de Twitter, la cual pedirá al usuario conceder acceso a sus mensajes y otros datos a la aplicación.

end_registration A este proceso se accede una vez que el usuario concede a la aplicación acceso a sus mensajes y otros datos. Con éste se completa la sincronización de la cuenta del usuario en Twitter con la cuenta en esta aplicación, y por lo tanto en adelante no es necesario volver a pedir permisos para acceder a sus datos. Finalmente este controlador redirecciona al usuario a la página principal del sitio. Los detalles relacionados con el proceso de autenticación y acceso a los datos serán explicados más adelante, dado

que comprenden más de un controlador y realizan peticiones a servicios de la API de Twitter.

get_login_url Este controlador se preocupa de iniciar la sesión del usuario. Para poder realizar esto, éste recibe un nombre de usuario y una contraseña, los cuales compara contra la base de datos. Luego, si son correctos, se asigna una *cookie* al navegador cuya llave lleva nombre 't' y cuyo valor es un identificador de sesión, el cual es generado y almacenado en la base de datos en el proceso de autenticación y obtención de acceso y cada vez que el usuario inicia sesión. Cada identificador de sesión es único para cada usuario y se genera utilizando un hash del *oauth_token*(explicado posteriormente) y caracteres generados al azar. El hecho de que este identificador quede guardado en una llave de nombre 't' y no con un nombre distinguible de manera más fácil es para tratar de evitar que la gente ingrese al sitio utilizando la cuenta de otro usuario. En un principio el identificador se generaba al momento de crear la cuenta y luego permanecía inalterado, lo que se convierte en un problema de seguridad, ya que cualquier usuario que conozca el identificador de sesión de otro usuario podría entrar al sitio identificado como este otro usuario. Es por esto que se modificó para que el identificador de sesión se genere cada vez que el usuario inicie sesión y se elimine cada vez que el usuario cierre sesión.

logout Este método, tal como su nombre en inglés lo indica, hace que el usuario cierre sesión con el sitio. Para esto elimina la cookie que contiene el identificador de sesión del navegador. Además, como se explicaba anteriormente, se elimina el identificador de sesión de la base de datos.

Un proceso dentro del contexto de los controladores de la aplicación que vale la pena explicar de manera más detallada es la autenticación y obtención de permisos para acceder a los datos de un usuario de Twitter. Lo que se persigue conseguir en este proceso completo es crear una cuenta de usuario en el servidor y sincronizarla con la cuenta de Twitter de dicho usuario, de manera que, al tener iniciada la sesión en la aplicación, ésta sea capaz de obtener los mensajes y otros datos del usuario a traves de la API. Para lograr la sincronización de ambas cuentas se implementó el protocolo OAuth, donde Twitter actúa como servidor y la presente aplicación actúa como cliente.

A continuación se explica paso a paso el proceso de autorización:

1. El usuario nuevo ingresa un usuario y una contraseña para crear una cuenta en el sitio; estos datos son enviados utilizando el método POST a '*register_account*', donde son recibidos por el controlador correspondiente.

2. Se verifica que los datos sean válidos y se crea un usuario en la base de datos. Este usuario contiene un nombre de usuario, un hash de su contraseña y una llave, la cual corresponde al hash del nombre de usuario más su contraseña. Esta llave será utilizada únicamente durante este proceso y tiene la finalidad de poder reconocer cuál es el usuario local que está sincronizando su cuenta con una cuenta de Twitter. No se utiliza el nombre de usuario directamente para este proceso, dado que cualquier usuario podría modificar este parámetro en su navegador y sincronizar su cuenta de Twitter con la de otro usuario existente en la aplicación. Esta llave se agrega a las *cookies* del navegador para poder saber más tarde cuál era el usuario con el cual hay que sincronizar las cuentas.

3. Luego, utilizando la librería Python-Twitter, se realiza una petición a la API de Twitter en la cual se le pasan como parámetros el *consumer_key* y el *consumer_secret*, los cuales son un par de claves que son asignadas por Twitter a la aplicación, una vez que esta es inscrita como una aplicación de Twitter. Además, se le pasa como parámetro una dirección donde Twitter debe redirigir más tarde. El servidor de Twitter devuelve luego algo llamado *oauth_token*, lo cual es una llave que le sirve a Twitter para identificar cuál es la aplicación de la cual proviene el siguiente request.

4. En este paso, se redirecciona al usuario hacia una página de Twitter, donde en la URL de la página va inserto el token recibido anteriormente. Este sitio pedirá al usuario iniciar sesión con su cuenta de Twitter con motivo de solicitar luego autorización para que la aplicación pueda acceder a sus mensajes y datos en general.

5. Al dar autorización para que la aplicación acceda a los datos de la cuenta de Twitter, éste redirecciona hacia la página '*end_registration*' de la aplicación, donde el controlador correspondiente recibe un parámetro llamado *oauth_token* y un *oauth_verifier*. La aplicación realiza otra petición a la API, pasando estas llaves como parámetros; la

API devuelve a esta petición un *oauth_token*, un *oauth_token_secret*, un *user_id* y un *screen_name*, entre otros. El *oauth_token* y el *oauth_token_secret* son un par de llaves con las que, junto al *consumer_key* y al *consumer_secret* correspondientes a la aplicación, se puede acceder a los mensajes y otros datos del usuario. El *user_id* corresponde al ID de usuario de la cuenta de Twitter y el *screen_name* corresponde al nombre de usuario de ésta. La aplicación busca al usuario en la base de datos utilizando la llave guardada anteriormente y le agrega los datos de *oauth_token*, *oauth_token_secret*, *user_id* y *screen_name*.

6. Por último, se elimina la llave utilizada para el registro de entre las cookies del navegador y agrega el identificador de ID del usuario, para iniciar la sesión del usuario. Luego se redirecciona al usuario a la página de inicio.

Plantillas

Esta segunda componente de la parte WEB del proyecto la componen las plantillas HTML. Éstas se encuentran dentro del proyecto en la carpeta *WEB/templates* y cada archivo dentro de esta carpeta corresponde a una plantilla distinta.

Los templates corresponden a lo que se conoce como vista en un patrón de estructura MVC(Modelo vista controlador). Estos son en esencia archivos que en su interior contienen codigo HTML mezclado con código python o *pseudo-python*, con la idea de que, al procesarlo con el procesador de templates que se esté utilizando, al pasarle las variables que se desea desplegar en la página, se genere código HTML puro. En estricto rigor, se podría generar cualquier tipo de documento o texto, pero en el caso de este proyecto y de proyectos WEB en general, se generan textos HTML. La sintaxis utilizada en los templates depende del procesador de templates que se esté utilizando. En este trabajo se decidió utilizar el Mako, por la sencilla razón de que éste permite, aparte de poder insertar variables y trabajar con condiciones y loops, utilizar herencia. La herencia permite tener un template que agrupa todos los componentes comunes a todas las vistas, y luego cada plantilla que es llamada por el controlador hereda lo definido en el template base e implementa lo correspondiente únicamente a dicha vista.

A continuación se describen las plantillas que forman parte del proyecto:

main.html Este es el template base que contiene la barra de menú superior(Figura 4.8), incluye las hojas de estilo correspondientes en el header y define una función a ser implementada para agregar elementos extra al header. Además incluye una función para ser implementada, donde debería ir el cuerpo de la vista. Estas funciones a implementar vienen por defecto definidas vacías para evitar que, de no implementar alguna de estas por alguna planilla que herede *main.html*, lance una excepción y no funcione la página. Éste además implementa el formulario de login, que aparece en el menú superior únicamente si el usuario no se encuentra con la sesión iniciada.

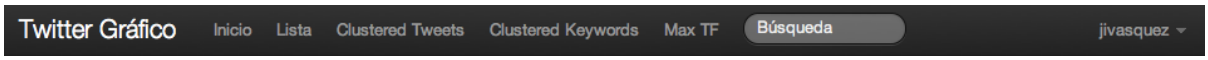

Figura 4.8: Barra Superior

home.html Esta plantilla es una página de inicio. Lo único que contiene es un texto que indica que es la página de inicio del sitio. Figura 4.9

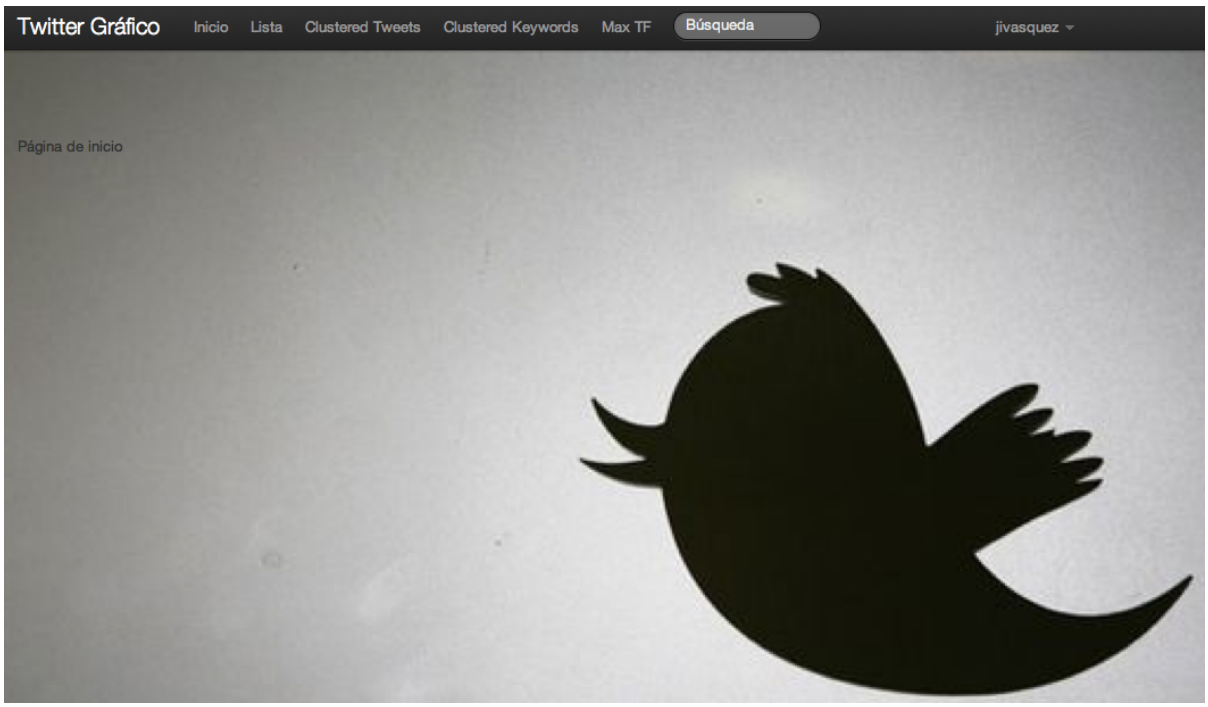

Figura 4.9: Página de Inicio

register.html Este template implementa un formulario para registrarse en el sitio. Los datos que se piden para poder registrarse en el sitio son: nombre de usuario y una contraseña.

Figura 4.10

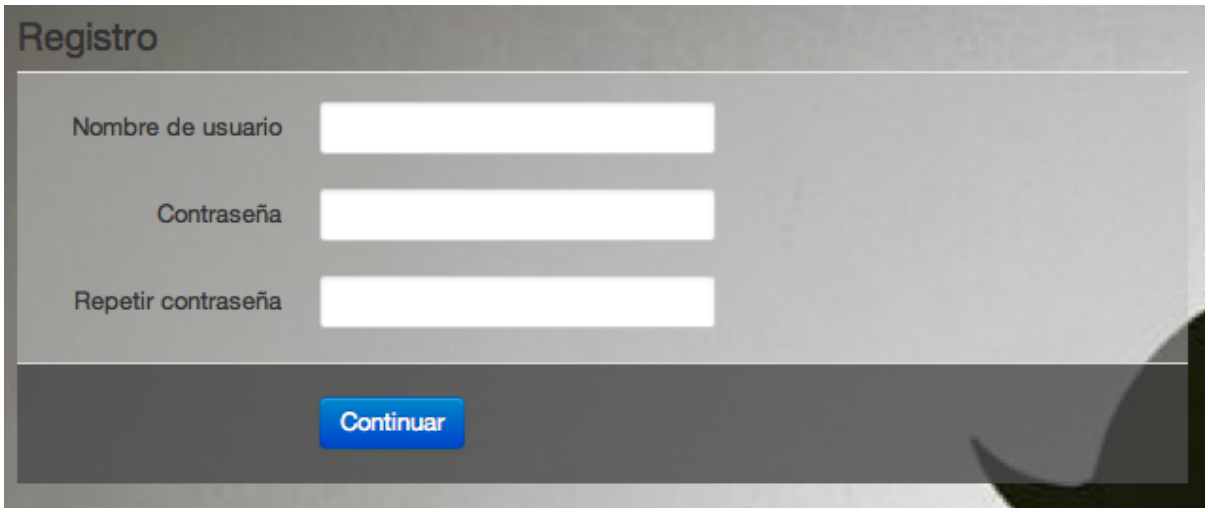

Figura 4.10: Página de Registro

list.html Es el template llamado por el controlador *list*. Éste muestra un collage con todas imágenes extraidas y al pasar el mouse por sobre la imagen se despliega un texto con el tweet correspondiente a ésta. Esta plantilla también es llamada por el controlador *search*. Figura 4.11

clustered_list.html Ésta es la plantilla llamada por los controladores *clustered*, *keywords* y *maxtf*. Dicho template muestra una imagen por cada cluster(la imagen que fue seleccionada previamente por el controlador) y al hacer click sobre la imagen se muestra el resto de las imágenes pertenecientes al grupo. Y, al igual que en la lista, si uno pasa el mouse por sobre la imagen se muestra un texto con el tweet correspondiente a la imagen. Figura 4.12.

Archivos Estáticos

Los archivos estáticos se encuentran en la carpeta *WEB/assets* y comprenden todos los archivos de imágenes, hojas de estilo CSS y archivos con código javascript presentes en el sitio WEB.

Dentro de los archivos se encuentran: la imagen de fondo, una hoja de estilo desarrollada para dar formato a algunos componentes del sitio y el resto lo componen librerías externas, las cuales se describen a continuación:

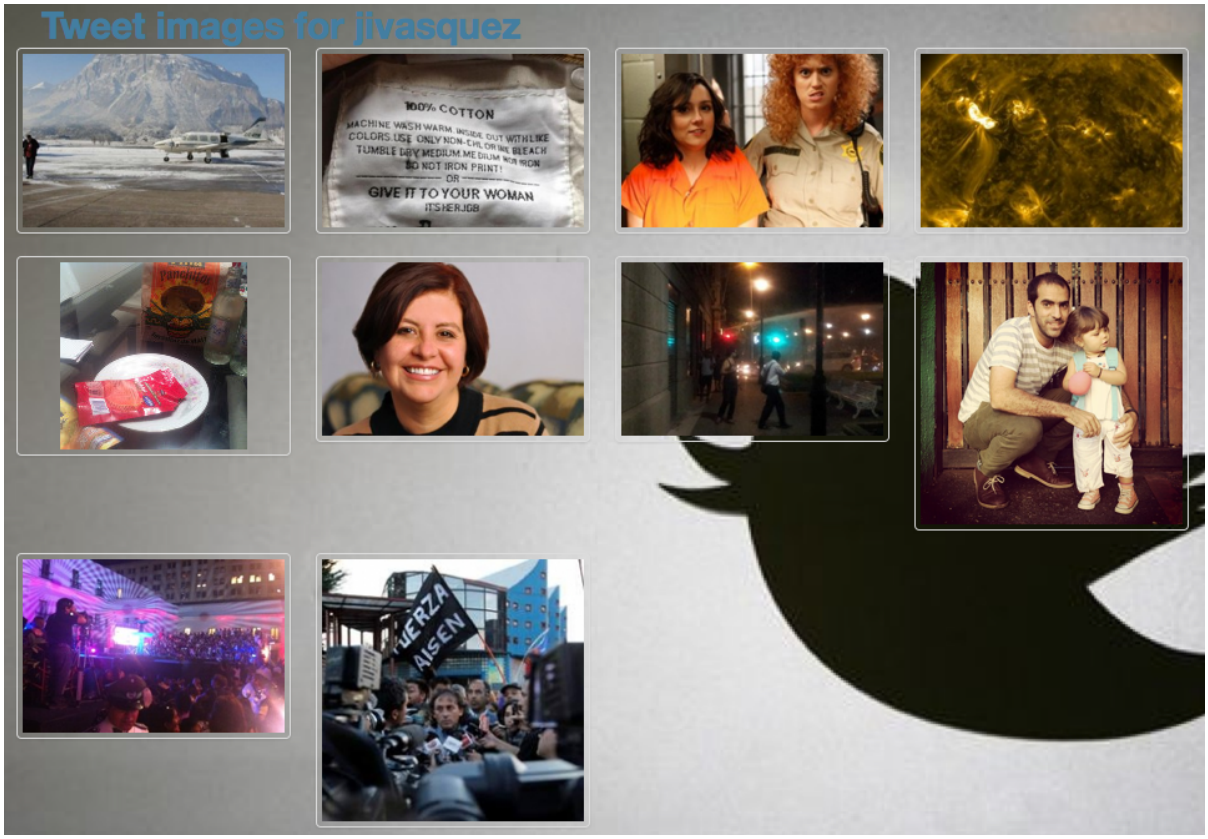

Figura 4.11: Página de Lista de Imágenes

Figura 4.12: Página de Tweets Agrupados

jQuery tools Esta es una librería externa que provee de herramientas para mejorar la interfaz de usuario de una página WEB. En particular, en este proyecto se utilizó, en un principio, *tooltip* y *overlay*. Tooltip hace que al pasar el mouse por sobre un elemento

que lo tenga implementado, se despliegue un pequeño cuadro de texto. Esto se utiliza en la página WEB para que, al pasar el mouse por sobre una imagen, se muestre sobre ésta el mensaje de Twitter correspondiente. En cuanto a Overlay, éste muestra, al hacer click sobre un elemento de la página, un cuadro con elementos WEB en primer plano, dejando el resto en segundo plano y oscurecido. Overlay se utilizó, en un principio, para que al hacer click sobre una imagen en alguna página que tenga clustering, se muestre el resto de las imágenes del cluster al que pertenece ésta. Luego se decidió descartar el Overlay que ofrece esta librería, dado que el texto que mostraba el mensaje representativo del cluster quedaba tapado por las imágenes. Además, cuando el grupo contenía muchas imágenes, había algunas que quedaban fuera de la ventana y por lo tanto no eran visibles. Es por esta razón que se decidió utilizar las *ventanas modales* implementadas en la librería de *Bootstrap*.

jQuery Es una biblioteca de javascript, que permite simplificar la manera de interactuar con los documentos HTML. Ésta es utilizada para poder implementar las funcionalidades de *jQuery tools*.

Bootstrap Corresponde a una completa serie de herramientas compuestas por código javascript y CSS, que ayudan a mejorar la interfaz de usuario, tanto en sus componentes como en sus interacciones, estilizando, ayudando en la usabilidad y brindando funcionalidades. En particular, para este trabajo de título se utilizó esta biblioteca para implementar la barra superior de la página, para desarrollar el formulario de inicio de sesión y para que las imágenes se muestren de manera ordenada, además de ordenar la estructura de la página y estilizar los componentes del sitio. Además, se utilizaron las ventanas modales que están implementadas en esta biblioteca para mostrar los clusters de imágenes, dado que es más ordenado y fácil de modificar que las ventanas modales de la librería jQuery tools. Cabe mencionar que se utilizó esta biblioteca porque es *open source*(código abierto) y es desarrollada por Twitter. Esto hace que el proyecto tenga una estructura y estilo similares a los que tiene Twitter.

4.2. Scripts Complementarios

Para mejorar la eficiencia tanto de espacio como de tiempo de carga de la página WEB, se desarrollaron dos scripts. El primero se encarga de obtener mensajes de fuentes que tienen más de 3000 seguidores; el segundo se encarga de borrar de la base de datos los *scripts* que lleven más de un día en la Base de Datos. Ambos scripts son ejecutados utilizando *Crontab*, el primero en intervalos de 5 minutos y el segundo en intervalos de 4 horas.

A continuación se detalla el porqué y cómo se implementó cada uno de estos script.

4.2.1. Script *get_tweets*

Este script se encarga de obtener mensajes constantemente de fuentes que tienen más de 3000 seguidores, con la intención de que, en el momento en que un usuario accede a la aplicación, ya haya una gran cantidad de los mensajes de su timeline cargados en la base de datos y de esta manera no sea necesario procesar todos los mensajes de este usuario. Al hacer esto se reduce en gran cantidad el tiempo de carga de la página.

Las fuentes de las cuales se obtienen los mensajes son aquellas que tienen más de 3000 seguidores y que han sido accedidas con anterioridad por algún usuario. Para lograr esto, la aplicación revisa por cada mensaje de Twitter, el usuario que escribió el mensaje y lo agrega a una lista de usuarios, junto con la cantidad de seguidores que tiene dicho usuario.

Para realizar esta tarea el script busca en la base de datos los usuarios con más de 3000 seguidores que han sido accedidos anteriormente. Luego, utilizando la función *get_messages_without_login* de *Twitter_DAO*, la cual se encarga de obtener mensajes desde cierta cuenta sin estar autenticado en la API de Twitter, recolecta dichos mensajes, los procesa y los agrega a la base de datos. Si llegase a haber un usuario masivo que tenga restringido el contenido que divulga, entonces este script no podría obtener sus publicaciones, pero por lo que se ha observado al hacer pruebas es que la mayor parte de los usuarios masivos no bloquean el contenido de sus mensajes.

4.2.2. Script *expire_tweets*

Este se encarga de eliminar los tweets después de un día desde que fueron agregados a la base de datos, con la intención de no llenar la base de datos. Dado que los mensajes que se despliegan en la página en su mayoría no tiene más de un par de horas desde su publicación, no es necesario guardar por mucho tiempo los mensajes que se despliegan.

Para realizar esto, el script calcula la fecha que era hace un día y llama a una función llamada *expire_tweets_before_date* perteneciente a *Twitter_DAO*, pasándole como parámetro la fecha límite hasta la que se desea borrar los mensajes. Dicha función se encarga de realizar un llamado a la base de datos en el que pide que se borren todos los mensajes anteriores a dicha fecha.

4.3. Casos de Uso

En esta sección se analiza un par de casos de uso donde se muestra cómo se utiliza la aplicación y los resultados que arroja tanto para la lista de imágenes como para las agrupaciones de mensajes. Todo esto siguiendo el orden lógico que se debe continuar.

4.3.1. Creación de Cuenta

El primer paso que se debe seguir es la creación de una cuenta en el sitio[1], para esto hay que hacer click en la palabra *Registrar* que se encuentra en en la barra superior. Luego, aparecerá un formulario como el que se vio anteriormente en la Figura 4.10, el cual debe ser llenado con un nombre de usuario y una contraseña, la cual debe contener al menos 6 caracteres.

Después se redirigirá a un sitio de Twitter, tal como se muestra en la Figura 4.13, el cual pedirá autorización para que *Twitter Gráfico* pueda acceder al timeline del usuario.

Al conceder permiso para acceder al timeline ya se habrá finalizado la creación de la cuenta y se podrá utilizar la aplicación.

[1]**Sitio:** http://twittergrafico.ignorelist.com

Figura 4.13: Página de Autorización

4.3.2. Inicio de Sesión

En caso tener una cuenta creada en el sitio, pero encontrarse sin una sesión iniciada, es necesario iniciar sesión para poder hacer uso de la aplicación. Para realizar esto, hay que hacer click sobre la palabra *Login*, que se encuentra en la barra superior de la página. Al hacer esto, aparecerá un cuadro de diálogo como el que se aprecia en la Figura 4.14, el cual debe ser rellenado con el nombre de usuario y contraseña de quien desee utilizar la aplicación y luego hacer click en el botón que dice *Iniciar Sesión*.

Figura 4.14: Ventana de Inicio de Sesión

4.3.3. Listado de Imágenes

El primer caso de uso es el de desplegar un listado con todas las imágenes que se obtengan al procesar los últimos 50 mensajes del timeline del usuario. Para realizar esto hay que hacer click sobre la palabra *Lista*, que se encuentra en la barra superior de la página. Al hacer esto se desplegará un collage con imágenes como el que se vió anteriormente en la Figura 4.11. Luego, si se hace click sobre alguna de estas imágenes aparecerá un cuadro mostrando dicha imagen con su respectivo mensaje, tal como se aprecia en la Figura 4.15.

Figura 4.15: Cuadro de Imagen

4.3.4. Imágenes Agrupadas

El segundo caso de uso que se muestra en este informe es el de esplegar imágenes y tweets agrupados por el mensaje que aparece en este. Para esto, es necesario encontrarse con una sesión iniciada y hacer click sobre *Tweets Agrupados* y luego sobre *Por Tweets*. Al hacer esto se desplegará un collage con imágenes y tweets como se pudo apreciar anteriormente en la Figura 4.12. Luego, al hacer click sobre alguna de las imágenes o algún texto aparecerá un cuadro de diálogo que muestra los demás tweets que pertenecen al mismo grupo, tal como se muestra en la Figura 4.16.

Figura 4.16: Ventana de Tweets Agrupados

Capítulo 5

Evaluación

Fueron realizadas una serie de evaluaciones y pruebas a la aplicación con la finalidad de evaluar y optimizar el rendimiento, la visualización y la calidad de ésta. A continuación se presentan los detalles de cómo fueron realizadas dichas pruebas, junto con los resultados obtenidos.

5.1. Prueba de Clusters

Para el desarrollo de este trabajo de título, como fue mencionado anteriormente, se utilizó una herramienta externa llamada Cluto para realizar el clustering de los mensajes. Dicha herramienta puede utilizar seis métodos diferentes para agrupar los mensajes, los cuales son: *rb*(bisecciones repetidas), *rbr*(bisecciones repetidas para refinamiento de *k-caminos*), *direct*(clustering de k-caminos directo), *agglo*(clustering aglomerativo), *graph*(clustering basado en particionamiento de grafos), *bagglo*(clustering aglomerativo parcialmente particional). Por esto se decidió hacer una prueba para verificar cuál de éstos agrupa de mejor manera los mensajes.

Para esta evaluación se tomaron los 50 últimos mensajes del timeline de un usuario y se agruparon usando cada uno de los métodos antes mencionados, utilizando los keywords de los mensajes como fuente para realizar el agrupamiento. Esto generó 12 clusters por cada uno de los métodos. Cada una de las agrupaciones fue evaluada por el autor del informe cualitativamente según cuan bien relacionados entre sí estuvieran los mensajes agrupados. Si los mensajes del grupo están bien relacionados entre sí, pudiendo haber a lo más un mensaje fuera de lugar, se consideró que el cluster se encuentra bien agrupado; si el cluster tiene algu-

nos elementos bien asociados entre ellos, pero hay bastantes que están fuera de lugar o que forman un segundo subgrupo de elementos dentro del mismo cluster, entonces se representa a dicho grupo como regularmente agrupado; en caso de que no se vea relación alguna en los elementos del grupo, o se formen muchos subgrupos (más de 2), entonces se considera como un cluster mal asociado, y aquellos grupos que contienen un único elemento se clasifican simplemente como clusters atómicos.

Los resultados obtenidos, los cuales se pueden ver en la Figura 5.1, son los siguientes:

	rb	rbr	direct	agglo	graph	bagglo
bien	5	5	5	2	6	3
regular	2	2	2	1	0	2
mal	5	5	5	3	5	2
atómico	0	0	0	6	1	5

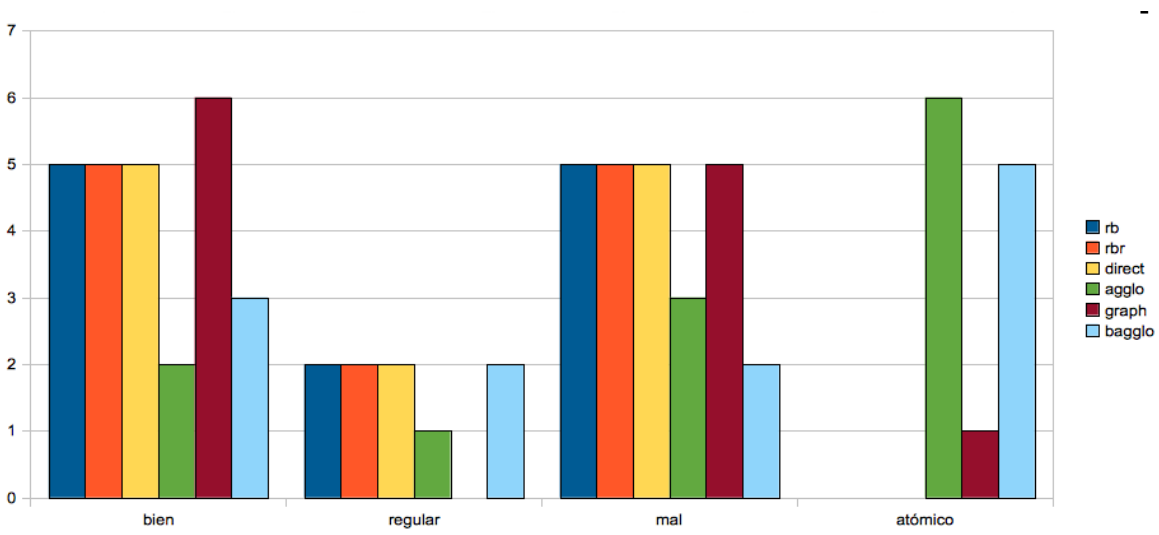

Figura 5.1: Prueba Métodos de Agrupación

Con los resultados de esta prueba se puede descartar ambos métodos aglomerativos dado que presentan un índice de clusters bien agrupados muy bajo en comparación con los demás métodos.

5.2. Prueba de Tiempo de Carga

Dado que esta aplicación debe hacer peticiones a una gran cantidad de sitios WEB diferentes para obtener las imágenes representativas de cada tweet, el tiempo de carga puede ser considerable. Por esto se decidió comparar el tiempo que demora la página en cargar al tener todos los datos cargados con anterioridad en la base de datos y cuánto tarda sin tener los datos con anterioridad. Los datos obtenidos, apreciables en el gráfico de la Figura 5.2 fueron los siguientes:

Sin base de datos	Con base de datos
84s	2,68s
96s	3,14s
72s	2,48s
58s	2,83s

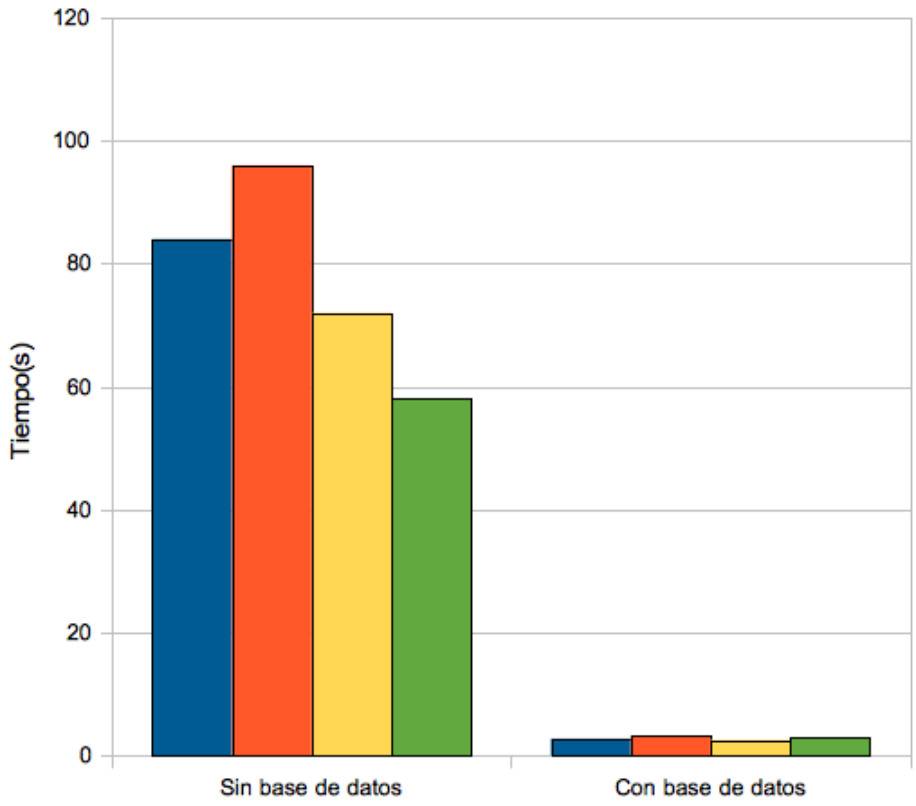

Figura 5.2: Gráfico del Test de Tiempo de Carga

Dados los resultados de esta prueba se puede ver que el tiempo de carga disminuye considerablemente con la incorporación de la base de datos. Es por esto que se debe poner

énfasis en el desarrollo del script que se encarga de precargar los mensajes de usuarios masivos en la base de datos.

5.3. Pruebas con usuarios

5.3.1. Encuesta

Con motivo de evaluar la aplicación, reconocer el valor que le dan los usuarios a los aspectos de ésta, ver qué tan intuitivo y fácil de utilizar es el sitio y poder plantear algunas mejoras para desarrollar en el futuro, se realizó una prueba con usuarios. En esta participaron 14 personas, las cuales fueron escogidas de entre los familiares, amigos y conocidos del alumno de manera tal que todos los usuarios tengan una cuenta en Twitter, y estos la utilicen en diversos grados de manera tal que la encuesta sea representativa respecto a la gran gama de personas que utilizan Twitter. Dicha prueba consistía en una serie de pasos que debían seguir los usuarios, los cuales van desde registrar una cuenta en el sitio, hasta revisar sus tweets en las diferentes secciones, y comparar estas secciones en diferentes aspectos. A medida que el usuario realiza los pasos que le eran indicados debía ir respondiendo una encuesta que contiene preguntas tanto abiertas como cerradas, las cuales fueron analizadas, llegando a los siguientes resultados:

La pregunta "¿Le parece atractiva e informativa la visualización de la información presentada?", tal como se puede apreciar en el gráfico de la Figura 5.3, muestra que un 71,43 % de los encuestados considera atractivo ver la información referente a sus mensajes de Twitter de manera gráfica, lo cual es un índice bastante positivo y con esto se puede concluir que a la vista de los usuarios la aplicación les agrega valor.

A los encuestados además se les preguntó el porqué consideran o no atractiva la visualización de la información presentada. A partir de lo que se puede sacar en limpio de las respuestas de quienes la consideran atractiva, podemos decir que la aplicación les permite encontrar algo interesante de manera más rápida que lo que se demorarían en Twitter. Además, los usuarios indican que ahorra tener que ver todos los enlaces y permite ver las fotos agrupadas en el mismo lugar. También porque es novedoso e interesante poder ver las fotos y luego, a partir de las fotos, descubrir los mensajes. De quienes no creen que la visualización de la información presentada sea atractiva se puede desprender lo siguiente: un usuario no comprendía de donde salían las imágenes y le tomó tiempo darse cuenta de cómo visualizar el mensaje detrás de dicha imagen. Además se sostiene que es muy escasa la cantidad de

imágenes que se despliega en el sitio. Por último, se comenta que la propuesta es interesante, pero que el diseño no es muy bueno; que falta ordenar las imágenes para que tengan un tamaño más homogéneo, no mostrar bordes blancos y ajustar los cuadros que muestran los mensajes, dado que los mensajes muy largos se salen de los márgenes de éstos.

De los comentarios surgidos a partir de esta pregunta, se puede concluir que la idea en sí parece atractiva a todos los encuestados y que lo que incomoda a algunos usuarios es la manera en la que se muestran los resultados y el orden de las imágenes dentro del sitio.

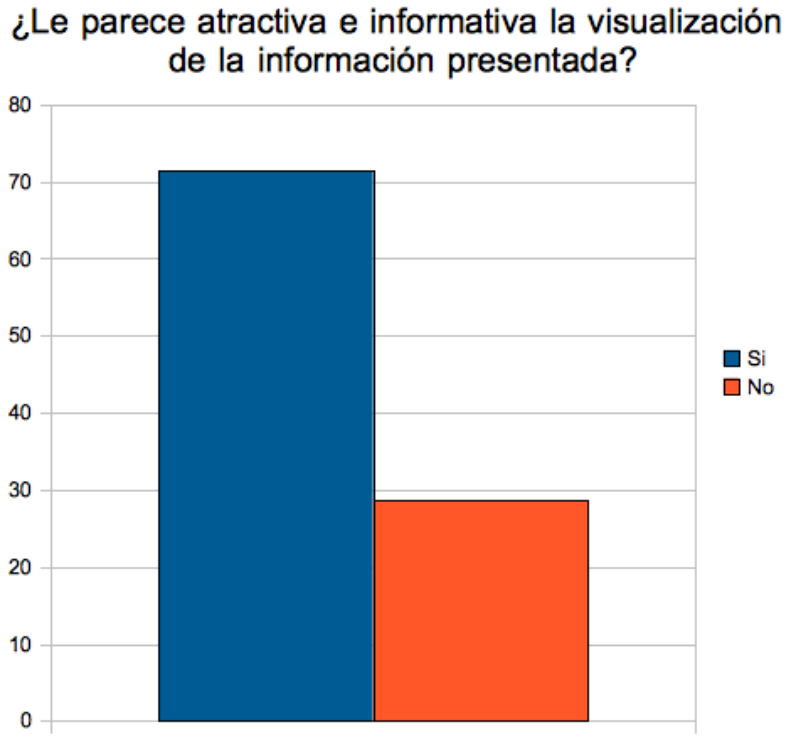

Figura 5.3: Pregunta N°1

La segunda pregunta que se hizo a los encuestados fue: “En su opinión, las imágenes que aparecen, ¿Son representativas del mensaje o tweet, que aparece al ubicar su mouse sobre la imagen?”, con la intención de saber si efectivamente las imágenes que se muestran van acordes a la noticia o evento que se está comentando en Twitter, dado que en muchos casos se muestra un logo institucional en vez de una fotografía del suceso. Como queda de manifiesto en el gráfico de la Figura 5.4, un 85,71 % de los usuarios que respondieron el sondeo consideran que las ilustraciones presentes en la página simbolizan de buena manera el contenido del texto, en tanto que sólo un 14,29 % cree que las imágenes que se despliegan no son representativas del mensaje correspondiente.

Junto con esta pregunta, se preguntó a quienes opinaron que las imágenes no simbolizan de buena manera el suceso del cual están hablando y éstas respondieron que no siempre es así, que a veces aparece una foto del autor de una frase y no del contenido del mensaje. Dado el alto índice que considera que las imágenes sí son representativas, se considera que no es preciso modificar, por ahora, el comportamiento del sitio de acuerdo a la manera en que se obtienen las imágenes. De todas maneras, es preciso definir, como trabajo futuro, una manera de agregar imágenes al filtro que se encarga de depurar logos institucionales, dado que actualmente este filtro funciona con una lista que se encuentra directamente en el código. Idealmente se debería generar una lista en la base de datos, la cual se vaya llenando en colaboración con los mismos usuarios de la aplicación.

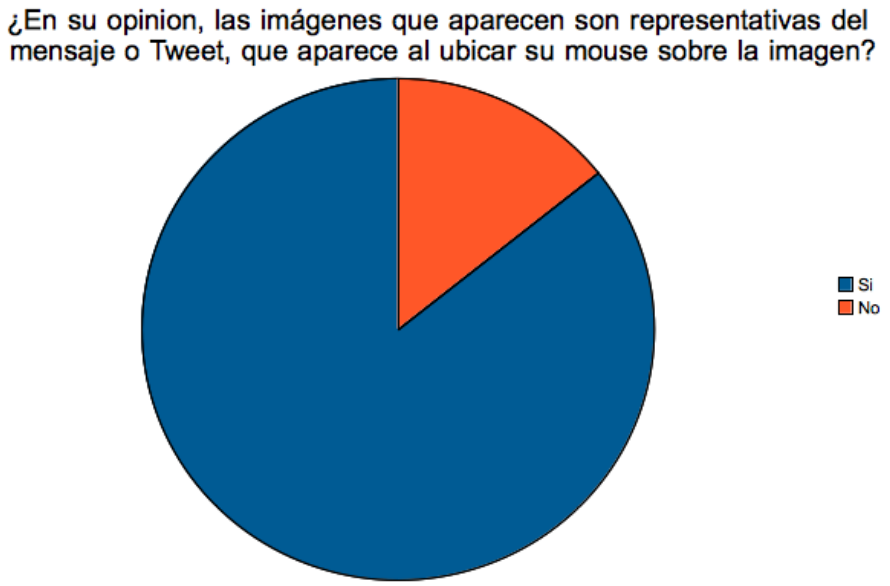

Figura 5.4: Pregunta N°2

La siguiente pregunta corresponde a la segunda parte de la encuesta, en la cual a los usuarios de la aplicación se les pidió que revisaran las imágenes agrupadas para responder las preguntas que seguían. La pregunta es "¿Considera usted que las imágenes agrupadas presentan alguna relación coherente entre si?". Los resultados obtenidos, tal como es apreciable en el gráfico de la Figura 5.5 muestran que un 14,29 % de los encuestados considera que las imágenes agrupadas tienen relación entre si, un 57,14 % considera que solo a veces se encuentran bien agrupadas y un 28,57 % cree que no están relacionadas entre sí.

Luego se pidió a los encuestados que argumentaran el porqué consideran que solo a veces se encuentran bien agrupadas o que simplemente no se encuentran bien agrupadas. De las respuestas dadas a esta pregunta se puede desprender lo siguiente: Quienes consideraban que algunos grupos presentan sus imágenes y mensajes bien agrupados comentaban que aparecian muchas veces grupos inconexos, o aparecían mensajes personales que no tenían relación alguna entre ellos. Además comentaban que aparece un grupo que agrupa todas las cosas que no tienen relación con el resto. Quienes habían respondido que no tienen relación las imágenes pertenecientes al mismo cluster dijeron que simplemente los grupos de imágenes son inconexos y que aparecían temas que no tenían relación alguna entre ellos. Esto ocurre porque Cluto genera un número de clusters estático, es decir, se le pide a Cluto que genere un número fijo de clusters y éste fuerza los resultados para que se genere esa cantidad de grupos, cuando lo ideal sería que se generara una cantidad dinámica de agrupaciones, dependiendo de la cercanía de sus mensajes. Es muy probable que quienes tengan una gran cantidad de mensajes personales tengan muchos mensajes inconexos; en cambio, quienes tengan más mensajes de fuentes periodísticas tengan grupos más coherentes.

Se proponen dos mejoras por realizar en un trabajo futuro para mejorar la calidad de los clusters: La primera es investigar y probar la implementación de clustering utilizando alguna otra herramienta que idealmente genere un número de clusters dinámico, o implementar un módulo de clustering propio utilizando el algoritmo propuesto en 3.1.2. La segunda mejora es quitar los mensajes personales o que no contengan una URL o una imagen del proceso de clustering; de esta manera se agrupan principalmente mensajes de índole noticiosa. Luego se pueden omitir los mensajes personales que fueron sacados de la fase de agrupación o

simplemente se pueden considerar como otro grupo más y mostrarlos de manera especial. La forma propuesta de distinguir un mensaje personal de uno noticioso es viendo la cantidad de seguidores que tiene quien escribe el mensaje. Actualmente esto ya se está realizando, por lo que no sería difícil separarles del resto.

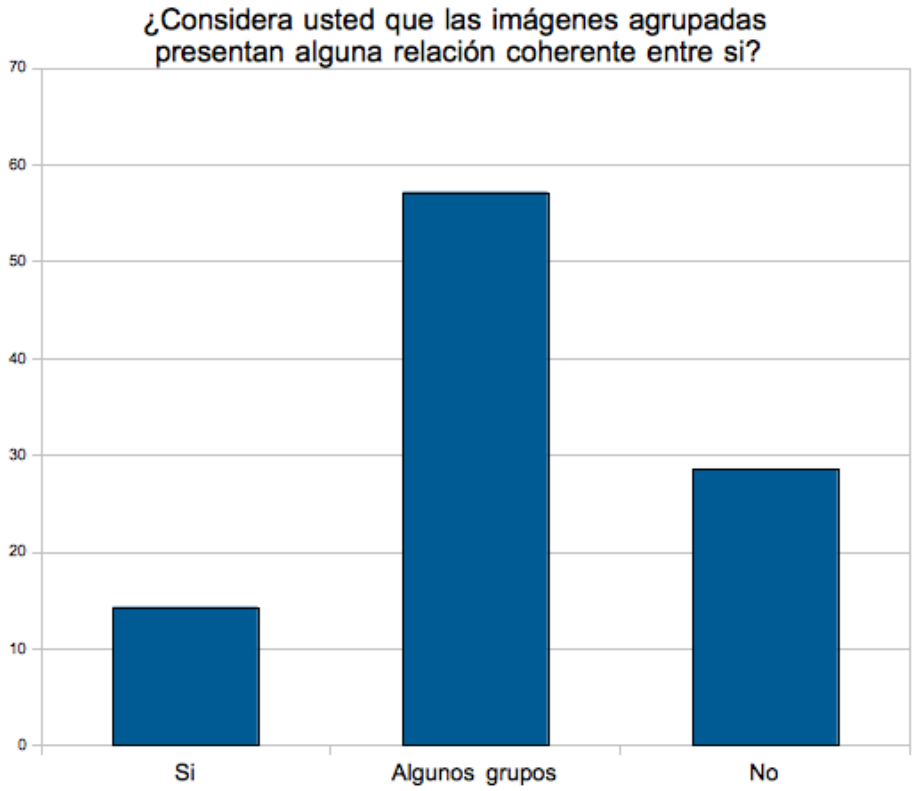

Figura 5.5: Pregunta N°3

La cuarta pregunta de la encuesta realizada es "¿Cree usted que la imagen principal (la que al hacer click muestra el resto de las imágenes del grupo) representa de buena manera al grupo de imágenes?". Se puede apreciar en el gráfico de la Figura 5.6 que tan solo un 21,43 % de los encuestados considera que la ilustración principal representa bien al conjunto de mensajes que pertenecen a su cluster; en cambio, un 78,56 % considera lo contrario. De este resultado se podría inferir que las imágenes que están siendo seleccionadas no representan bien al grupo o que los grupos que se están formando no tienen relación entre sí.

Como complemento y para poder sacar una conclusión más acertada, se preguntó a los encuestados el porqué consideran que las ilustraciones no simbolizan de manera adecuada al grupo al cual pertenecen. De las respuestas dadas por los usuarios se puede sacar en limpio las siguientes conclusiones: Algunos usuarios notan que las imágenes o noticias que aparecen dentro del texto no están relacionados y por ende ninguna ilustración podría representar al grupo. Otros usuarios opinan que da la impresión de que la imagen que aparece representando al grupo es simplemente la primera que aparece, es decir, que fue elegido algún tweet al azar. Además a veces aparecen grupos atómicos (con un solo mensaje) y por lo tanto es absurdo hablar de imágenes representativas para éstos. Por último, se comentó que algunas veces sí ocurría que las imágenes representativas estaban bien, pero que en grupos más heterogéneos no había un ícono que los pudiera simbolizar de buena manera, dado lo disímil de sus contenidos.

Para poder mejorar este aspecto de la aplicación habría que mejorar el método de agrupación, de manera que queden temas noticiosos marcados y entonces se pueda calibrar de mejor manera la elección del mensaje que represente a cada grupo, ya que no sirve de nada tratar de mejorar la selección de un ícono para el cluster si todos sus elementos hablan de un tema distinto.

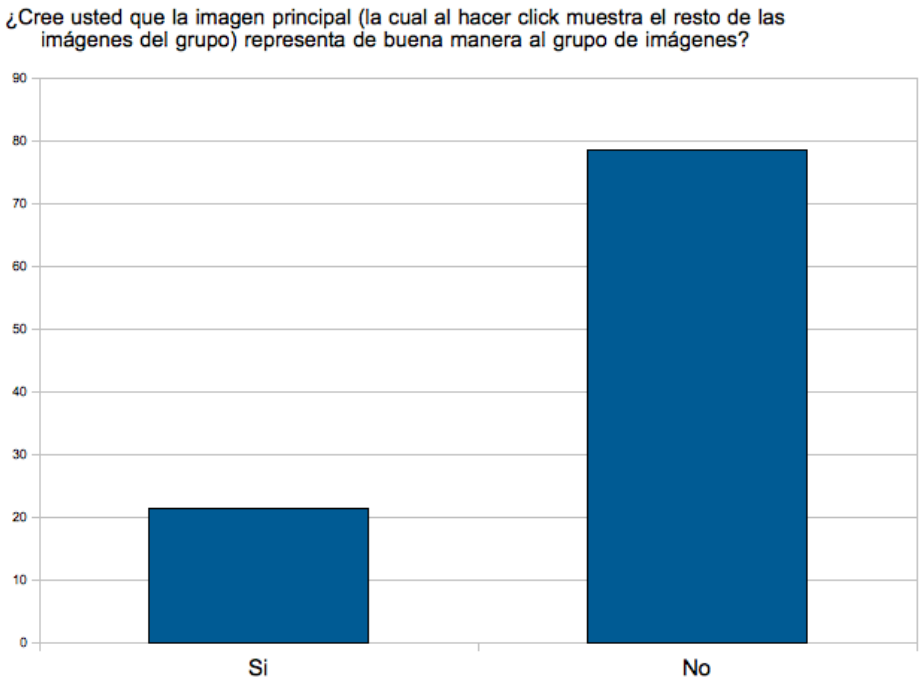

Figura 5.6: Pregunta N°4

En la quinta pregunta de este sondeo se preguntó a los participantes "¿Cuál de estos considera usted que se encuentra agrupado de mejor manera?" con la intención de saber si encontraban que dichas formas de agrupar elementos daban resultados diferentes y además ver cuál de éstas daba un mejor resultado.

De las respuestas obtenidas en esta consulta, se puede notar a partir del gráfico de la Figura 5.7 que un 42,86 % de los usuarios considera que los tweets agrupados por texto se encuentran agrupados de mejor manera, lo cual no es conclusivo, pero marca una clara tendencia que apunta a decir que éste es el método con el cual se obtienen mejores resultados, lo cual es esperable, dado que los textos de los enlaces correspondientes a los tweets son más extensos en cantidad de palabras relevantes que las keywords y que los tweets. De todas maneras habría que estudiar si Cluto toma en cuenta el largo de los textos al momento de compararlos, dado que utilizando esta fuente muchas veces se comparan textos extensos provenientes de la página WEB del enlace con textos cortos de tweets que no tienen un sitio WEB asociado.

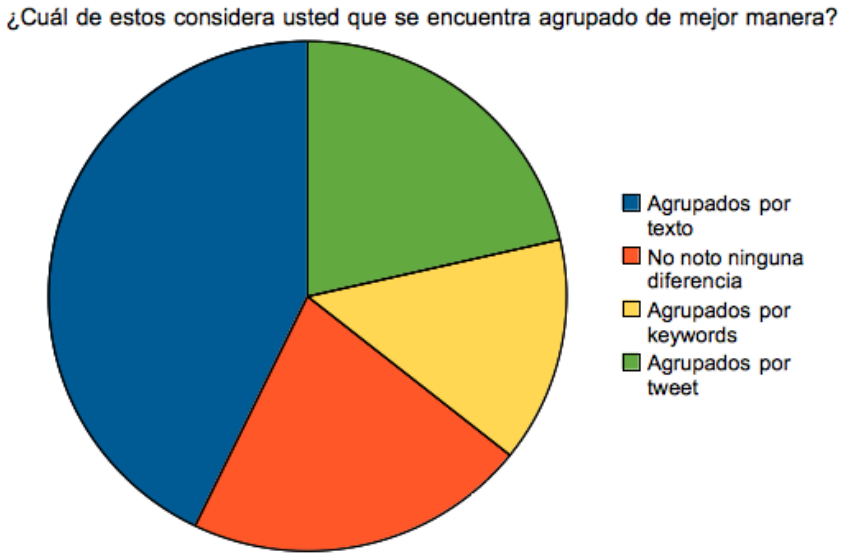

Figura 5.7: Pregunta N°5

La sexta pregunta de la encuesta es "¿Le parece útil poder revisar lo que ocurre en su cuenta de Twitter de manera gráfica?". Esta pregunta tenía la finalidad de analizar la importancia que dan los usuarios al poder inspeccionar de una forma alternativa y más visual lo que está ocurriendo en su cuenta de Twitter.

Tal como es posible ver en el gráfico de la Figura 5.8 sólo un 7,14 % de los encuestados considera que no es útil poder revisar lo que ocurre en su cuenta de Twitter de manera gráfica; en cambio, a un 92,86 % sí le parece útil tener la oportunidad de apreciar lo que pasa en su timeline como un collage de ilustraciones, lo cual nos indica que existe un potencial de gente que podría estar interesada en utilizar una aplicación que sea capaz de realizar la tarea de mostrarle de forma gráfica lo que ocurre en su cuenta. Es también un buen indicador de que la idea planteada va bien encaminada y es preciso mejorar los puntos débiles que pueda tener el sitio para hacerlo más atractivo a los usuarios.

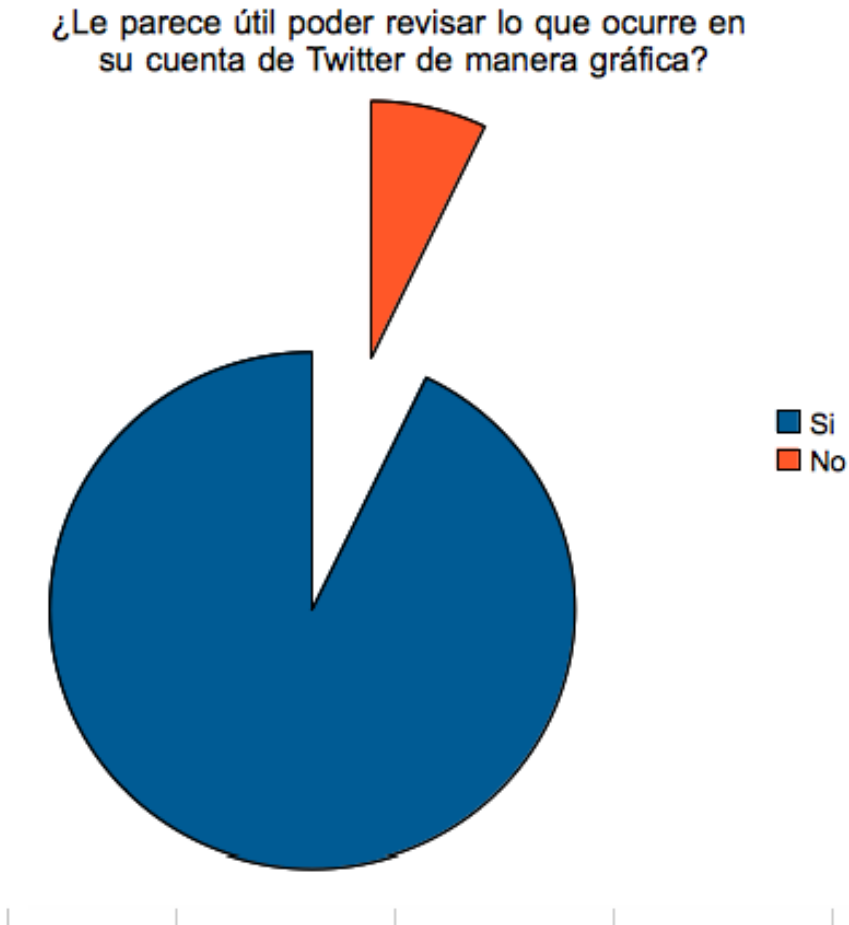

Figura 5.8: Pregunta Nº6

Luego en el presente sondeo se consultó "¿Le parece apropiada la cantidad de imágenes que se muestran?" con la intención de calibrar la cantidad de imágenes que se desea que aparezcan.

Los resultados obtenidos a partir de esta pregunta, como se puede apreciar en la Figura 5.9 son los siguientes: 14,29 % manifestó que sí era apropiada la cantidad de imágenes que se muestran, en cambio un 85,71 % dijo que debería mostrarse un mayor número de imágenes. Ninguno de los encuestados dijo que debiera haber menos ilustraciones.

La aplicación muestra las imágenes que puedan ser obtenidas a partir de los últimos 50 mensajes de Twitter, dado que la obtención de estas es muy lenta. Esto podría cambiarse y mostrar un número mayor de imágenes, ya que el script que precarga los mensajes de usuarios con gran número de seguidores hace que el despliegue de las imágenes no sea tan lento. Para mostrar un mayor número de ilustraciones se propone como trabajo futuro implementar alguna de las siguientes soluciones:

La primera forma es consultar al usuario cuántas imágenes desea ver. De esta manera se debe traer mensajes desde la API de Twitter hasta completar la cantidad de imágenes que se desea mostrar, para luego desplegarlas. Dada la forma en que está implementado el sitio, bastaría con modificar el módulo de obtención.

La segunda solución es añadir un botón que permita incluir más imágenes a las que se están mostrando, sin tener que recargar la página. Esta solución es más atractiva visualmente y su implementación es bastante similar a la otra. Lo que se propone es definir en una variable javascript la cantidad de ilustraciones que se están mostrando y luego al apretar el botón de siguiente se envía una petición *Ajax* (actualmente se está trabajando con Ajax, por lo cual esto no cambia mucho), pasando como parámetro el nuevo número de imágenes que se desea mostrar. Al concluir el procesamiento por parte del servidor y sea recibida una respuesta, se limpian las imágenes que se mostraban antiguamente y se agregan las imágenes procesadas más recientemente.

Para el desarrollo de ambas soluciones, se sugiere revisar la documentación que se encuentra en la página de Twitter respecto a cómo trabajar con los timelines [37].

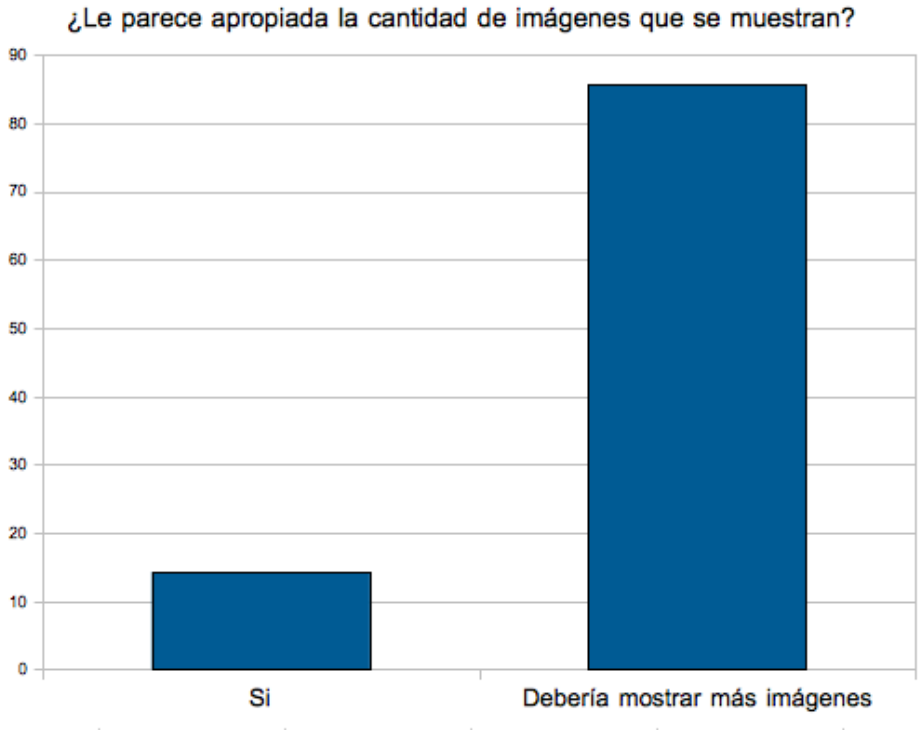

Figura 5.9: Pregunta N°7

La octava pregunta de la encuesta es "¿Usaría una aplicación así?". La única finalidad que tiene esta pregunta es medir el nivel de aceptación que tiene la aplicación en el círculo de usuarios encuestados. Como se puede apreciar en el gráfico de la Figura 5.10 , un 92,86 % utilizaría la aplicación, lo cual es un índice de aceptación muy bueno.

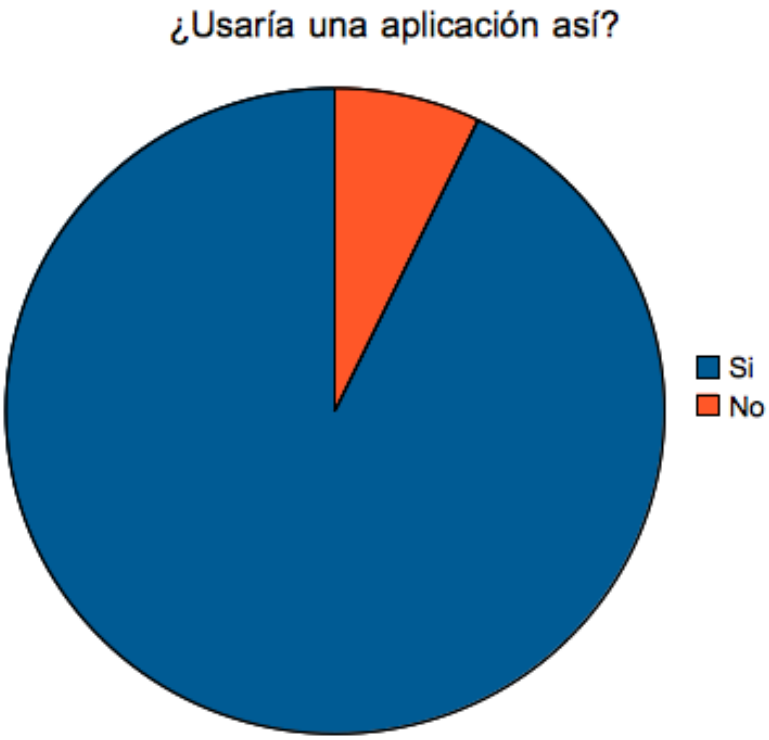

Figura 5.10: Pregunta N°8

Finalmente se pidió a los usuarios dejar sugerencias para mejorar el sitio. De éstas se pudo extraer las siguientes sugerencias:

- Clasificar las imágenes en tópicos, ya que los mensajes que se agrupan mensajes son muy distintos.
- Mostrar los trending topics.
- Agrandar las imágenes pequeñas para que no se vea tan desordenado y agrupar también por usuario. Mostrar quién es el autor de cada mensaje, dado que es confuso el no poder ver quién escribió cada mensaje. Esta sugerencia fue hecha por varios usuarios.
- Eliminar los tweets de índole personal, lo cual podría ser logrado, como fue dicho anteriormente, sacando todos los mensajes sin imagen o sacando todos los mensajes de usuarios con pocos seguidores.
- Se recomiendan otras mejoras visuales, tales como cambiar la imagen de fondo y el color de fondo, ya que la ilustración de fondo se ve pixelada.
- Utilizar una cuenta de google, facebook o twitter para registrarse, en vez de tener que crearse una cuenta en el sitio.
- Otra sugerencia bastante interesante que fue hecha, es el "aprender" de los gustos del usuario a partir de las imágenes en las cuales hace click. Con esta información se podría más adelante dar más prioridad a los temas más relacionados con los gustos de este usuario.
- Fue sugerido además que se visitaran los siguientes sitios para obtener ideas de cómo mejorar el despliegue visual: http://www.twitterfountain.com, http://visibletweets.com, http://flipboard.com

5.3.2. Estadísticas

Utilizando los datos obtenidos al realizar la encuesta se realizó un estudio estadístico con la finalidad de saber qué porcentaje del total de los mensajes que son desplegados por el sitio se encuentran cubiertos por el script *get_tweets*.

Como se puede apreciar en el gráfico de la Figura 5.11, un 37,73 % de los mensajes fueron escritos por usuarios con más de 3000 seguidores, lo cual no parece ser un porcentaje alto de mensajes cubiertos por el script en cuestión. Pero lo que se busca realmente al correr *get_tweets* es disminuir el tiempo que demora la carga de los mensajes que tienen URL, por lo que no hace sentido incluir en las estadísticas a los mensajes que no contienen enlace. Es por esto que se decidió analizar la cantidad de mensajes provenientes de usuarios a partir de aquellos mensajes que contienen URL únicamente.

De este segundo estudio de los resultados se puede observar en el gráfico de la Figura 5.12 que un 61,99 % del total de mensajes que contienen un enlace provienen de una fuente con más de 3000 seguidores y por lo tanto son tweets que de manera muy probable pasan a ser preprocesados por el script en cuestión. Este aumento tan abrupto en el porcentaje de mensajes provenientes de fuentes con más de 3000 seguidores al filtrar aquellos que tienen una URL contenida en el texto se debe a que aquellos usuarios con una cantidad reducida de adeptos tienden a escribir mensajes de índole personal, los cuales por lo general no contienen enlaces. En cambio aquellos usuarios con una gran cantidad de seguidores corresponden por lo general a medios periodísticos en gran medida y a figuras públicas, los cuales tienen la tendencia a escribir textos menos privados y de carácter más informativo. En particular, aquellos usuarios que corresponden a medios periodísticos escriben un volumen de tweets muy alto comparado con el resto de los usuarios y además, dentro de los mensajes redactados por éstos, la gran mayoría contiene un enlace que lleva hacia una página donde se detalla de manera más extensa el suceso que está aconteciendo.

El resultado obtenido es un índice que satisface la cobertura que se necesitaban para lograr mejorar los tiempos de carga del sitio. Aún así queda como trabajo futuro realizar pruebas para ver cuánto varía el índice de cantidad de mensajes que provienen de una fuente masiva al cambiar la cantidad de seguidores que definen si un usuario es masivo o no. Además, revisar cuánto afecta este cambio al rendimiento del procesador.

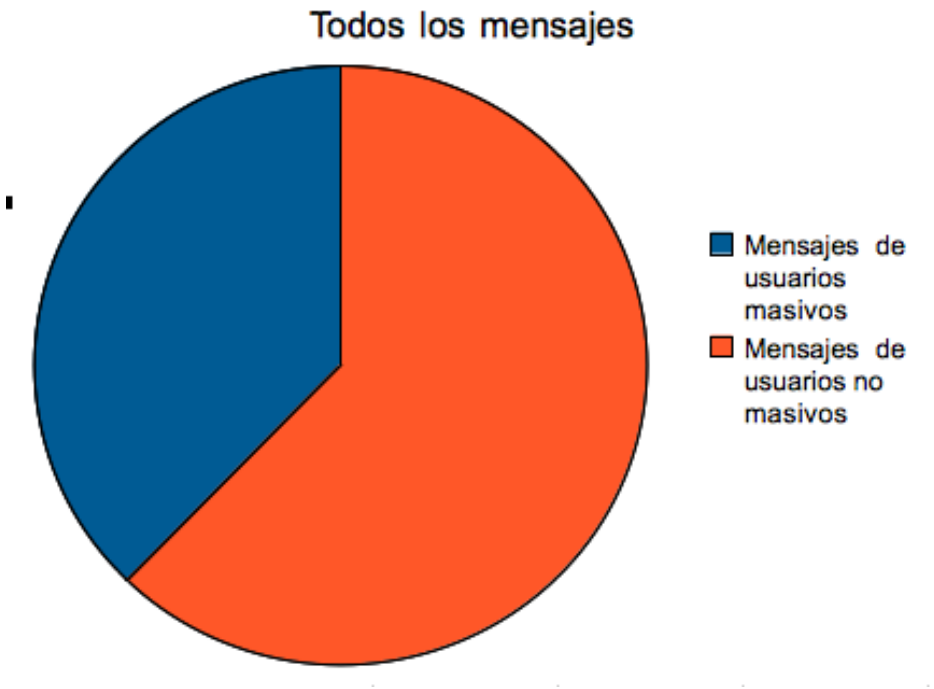

Figura 5.11: Mensajes cubiertos por el script(Total de mensajes)

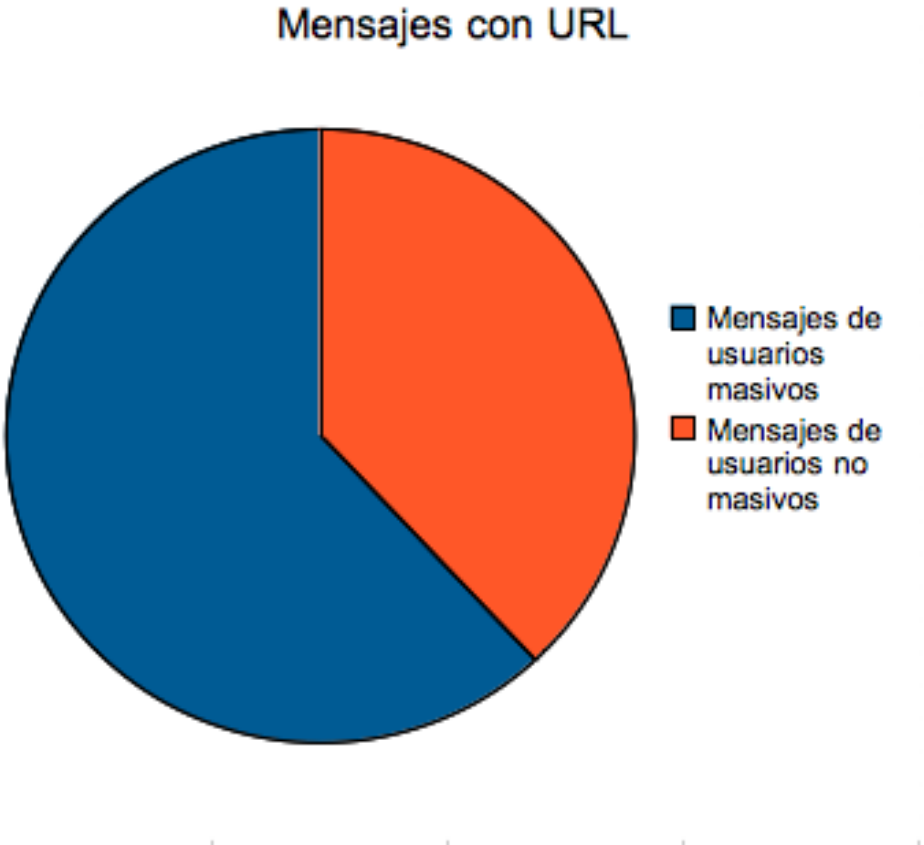

Figura 5.12: Mensajes cubiertos por el script(Mensajes con URL)

Capítulo 6

Conclusiones

Se consiguió desarrollar un prototipo de aplicación WEB que permite a los usuario revisar lo que ocurre en su timeline de Twitter de manera gráfica, mostrando una nube de imágenes que, al hacer click sobre ellas, despliega información adicional de estas. Si se está viendo el listado de imágenes se puede apreciar el mensaje correspondiente de donde proviene la imagen y si se está en algún modo de clustering de mensajes se puede ver el grupo completo de ilustraciones al cual pertenece dicha imagen.

Se logró implementar de manera eficiente la agrupación automática de imágenes utilizando para esto tres fuentes de información diferentes: el texto del mensaje, las keywords del sitio WEB al que enlaza el mensaje y el texto de dicha página. Junto con esto se planteó un algoritmo pendiente de desarrollar que tiene por fin mejorar los resultados obtenidos con la herramienta Cluto.

Se desarrolló además un módulo que se encarga de seleccionar una imagen representativa de cada cluster a partir de los textos que se tienen y utilizando filtros de imágenes no deseadas. Junto con la implementación de esta solución, se dejó esbozado un algoritmo de selección alternativo que establece una mejora en la obtención de una ilustración representativa del grupo.

Además, junto con el desarrollo de la aplicación, se realizó una encuesta entre los usuarios de la cual surgieron una serie de propuestas para mejorar tanto la estética de ésta, como algunos aspectos de su funcionalidad. Entre dichas propuestas se encuentran: Mejorar los

filtros de imágenes. Dar la opción de desplegar un número mayor de ilustraciones. Mostrar trending topics. Homogeneizar los tamaños de las imágenes. Eliminar tweets que hablan de temas personales. Mostrar el nombre del autor del mensaje, entre otras.

Por último, se planteó una serie de propuestas a realizar en un trabajo futuro para mejorar el funcionamiento de esta aplicación. Dichas propuestas son las siguientes: desarrollar un módulo que encuentre imágenes idénticas entre sí; reimplementar el módulo de *Clustering* de manera que sea capaz de generar una cantidad de clusters dinámica independiente del número de mensajes que se tomen. Para esto se puede implementar el algoritmo que se dejó propuesto con dicho fin en el capítulo del desarrollo y reimplementar el módulo de *Selección*, desarrollando el algoritmo que se dejó propuesto en el capítulo de desarrollo y compararlo con el módulo actual.

Capítulo 7

Referencias

[1] TWITTER. Twitter Libraries. [en línea] <http://dev.twitter.com/pages/libraries> [consulta : 30 abril 2012]

[2] TWITTER. Twitter API. [en línea] <http://dev.twitter.com> [consulta : 30 abril 2012]

[3] HTML PARSER PROJECT. HTML Parser Documentation [en línea] <http://htmlparser.sourceforge.net> [consulta : 30 abril 2012]

[4] S.C. CHEN. PHP Simple HTML DOM Parser [en línea] <http://simplehtmldom.sourceforge.net> [consulta : 30 abril 2012]

[5] GORDON PETRIE, PAUL and DAVID. Python and HTML Processing [en línea] <http://www.boddie.org.uk/python/HTML.html> [consulta : 30 abril 2012]

[6] DAN BIKLE. Hpricot [en línea] <http://hpricot.com/> [consulta : 30 abril 2012]

[7] STACKOVERFLOW. Sharing a link on facebook: how to get thumbnails? [en línea] <http://stackoverflow.com/questions/5968670/sharing-a-link-on-facebook-how-to-get-thumbnails> [consulta : 30 abril 2012]

[8] FACEBOOK. The Open Graph protocol [en línea] <http://ogp.me/> [consulta : 30 abril 2012]

[9] PAUL HAYMAN. URL Regular Expression [en línea] <http://www.geekzilla.co.uk/view2D3B0109-C1B2-4B4E-BFFD-E8088CBC85FD.htm> [consulta : 30 abril 2012]

[10] WIKIPEDIA. Base de datos relacional [en línea] <http://es.wikipedia.org/wiki/Base_de_datos_relacional> [consulta : 30 abril 2012]

[11] THE PYTHON-TWITTER. Python Twitter [en línea] <http://code.google.com/p/python-twitter> [consulta : 30 abril 2012]

[12] HILA BECKER, MOR NAAMAN, and LUIS GRAVANO. Learning Similarity Metrics for Event Identi

cation in Social Media. In Proceedings, the Third ACM International Conference on Web Search and Data Mining (WSDM 2010), February 2010, New York, NY.

[13] HILA BECKER, MOR NAAMAN, and LUIS GRAVANO. Selecting Quality Twitter Content for Events. In Proceedings, International Conference on Weblogs and Social Media. (ICWSM 2011), July 2011, Barcelona, Spain.

[14] IDÉE. PixMatch [en línea] <http://ideeinc.com/products/pixmatch> [consulta : 30 abril 2012]

[15] WIKIPEDIA. Data Access Object [en línea] <http://es.wikipedia.org/wiki/Data_Access_Object> [consulta : 30 abril 2012]

[16] WIKIPEDIA.[en línea] <http://es.wikipedia.org/wiki/Extract,_transform_and_load> [consulta : 30 abril 2012]

[17] IVÁN AMÓN y CLAUDIA JIMÉNEZ. Funciones de Similitud sobre Cadenas de Texto: Una Comparación Basada en la Naturaleza de los Datos [en línea] <http://www.bdigital.unal.edu.co/2033/4/71644758.20104.pdf> [consulta : 30 abril 2012]

[18] KARIPIS LAB. [en línea] <http://glaros.dtc.umn.edu/gkhome/views/cluto> [consulta : 30 abril 2012]

[19] KARIPIS LAB. [en línea] <http://glaros.dtc.umn.edu/gkhome/files/fs/sw/cluto/doc2mat.html> [consulta : 30 abril 2012]

[20] WIKIPEDIA. Analizador Sintáctico [en línea] <http://es.wikipedia.org/wiki/Analizador_sintáctico> [consulta : 30 abril 2012]

[21] WIKIPEDIA. XPath [en línea] <http://es.wikipedia.org/wiki/XPath> [consulta : 30 abril 2012]

[22] WIKIPEDIA. XML [en línea] <http://es.wikipedia.org/wiki/Extensible_Markup_Language> [consulta : 30 abril 2012]

[23] WIKIPEDIA. HTML [en línea] <http://es.wikipedia.org/wiki/Microblogging> [consulta : 30 abril 2012]

[24] WIKIPEDIA. HTML [en línea] <http://es.wikipedia.org/wiki/HTML> [consulta : 30 abril 2012]

[25] WIKIPEDIA. Algoritmo de agrupamiento [en línea] <http://es.wikipedia.org/wiki/Algoritmo_de_agrupamiento> [consulta : 30 abril 2012]

[26] WIKIPEDIA. Palabras vacías [en línea] <http://es.wikipedia.org/wiki/Palabras_vacías> [consulta : 30 abril 2012]

[27] WIKIPEDIA. Expresión regular [en línea] <http://es.wikipedia.org/wiki/Expresión_regular> [consulta : 30 abril 2012]

[28] WIKIPEDIA. Web Server Gateway Interface [en línea] <http://en.wikipedia.org/wiki/Web_Server_Gateway_Interface> [consulta : 30 abril 2012]

[29] WIKIPEDIA. Modelo Vista Controlador [en línea] <http://es.wikipedia.org/wiki/Modelo_Vista_Controlador> [consulta : 30 abril 2012]

[30] WIKIPEDIA. Decorator Pattern [en línea] <http://en.wikipedia.org/wiki/Decorator_pattern> [consulta : 30 abril 2012]

[31] WIKIPEDIA. Función Hash [en línea] <http://es.wikipedia.org/wiki/Función_Hash> [consulta : 30 abril 2012]

[32] WIKIPEDIA. Cookie (informática) [en línea] <http://es.wikipedia.org/wiki/Cookie_(informática)> [consulta : 30 abril 2012]

[33] WIKIPEDIA. Application programming interface [en línea] <http://en.wikipedia.org/wiki/Application_programming_interface> [consulta : 30 abril 2012]

[34] WIKIPEDIA. OAuth [en línea] <http://es.wikipedia.org/wiki/OAuth> [consulta : 30 abril 2012]

[35] WIKIPEDIA. Ventana (informática) [en línea] <http://es.wikipedia.org/wiki/Ventana_(informática)> [consulta : 30 abril 2012]

[36] WIKIPEDIA. Cron (Unix)[en línea] <http://es.wikipedia.org/wiki/Cron_(Unix)> [consulta : 30 abril 2012]

[37] Twitter. Working with Timelines [en línea] <https://dev.twitter.com/docs/working-with-timelines> [consulta : 30 abril 2012]

[38] MAKO. Mako Templates for Python [en línea] <http://www.makotemplates.org> [consulta : 30 abril 2012]

Capítulo 8

Anexos

8.1. Encuesta

A continuación se presenta la encuesta realizada a los usuarios:

Prueba Twitter Gráfico

Para poder mejorar la utilidad de la aplicación Twitter gráfico, se ha diseñado esta pequeña prueba en la que se deberán seguir algunos pasos dentro de ésta e ir respondiendo el cuestionario a medida que va completando los pasos. Cabe mencionar que para realizar esta prueba es necesario tener una cuenta en Twitter.

Para comenzar con la prueba, se le pide que ingrese a la siguiente página: http://twittergrafico.ignorelist.com.

Luego diríjase a "Registrar" para crear una nueva cuenta, aquí debe poner un nombre de usuario, una contraseña y repetir dicha contraseña. No necesariamente deben ser su usuario y contraseña de Twitter.

Al presionar continuar se abrirá una página de Twitter que le pedirá iniciar sesión, en caso de que no tenga su sesión de Twitter iniciada, y dar permisos a Twitter Gráfico para acceder a sus tweets. Al hacer esto será redirigido a la página de Twitter gráfico, donde comenzaremos a realizar las pruebas.

Primer paso: Presione "Lista" en el menú, ésto hará que se muestre una serie de imágenes. Esto puede tardar varios segundos. Al hacer click sobre ellas, o posar el mouse encima, se puede leer el tweet correspondiente a dicha imagen.

* Required

¿Le parece atractiva e informativa la visualización de la información presentada? *

Si

No

¿Por qué?

¿En su opinion, las imágenes que aparecen son representativas del mensaje o Tweet, que aparece al ubicar su mouse sobre la imagen? *

Si

No

¿Por qué?

Responder si es que crees que las imágenes no representan de manera adecuada los mensajes.

Segunda parte

Presione "Tweets agrupados por tweets" en el menú, esto hará que se muestre una serie de imágenes y textos. Al igual que en el listado anterior, esto puede tardar varios segundos. Al hacer click en alguna de las imágenes se abrirá una pequeña ventana dentro del navegador mostrando imágenes relacionadas. Y al posar el mouse sobre una imagen se puede leer el Tweet correspondiente a dicha imagen

¿Considera usted que las imágenes agrupadas presentan alguna relación coherente entre si? *

Si

No

Algunos grupos

¿Por qué?

Responda esta pregunta si cree que no están bien agrupados o si cree que solo algunos están bien agrupados

¿Cree usted que la imagen principal (La que al hacer click muestra el resto de las imágenes del grupo) representa de buena manera al grupo de imágenes? *

Si

No

¿Por qué?

Responder sólo si su respuesta anterior fue no

Tercer paso

Entre a "Tweets agrupados", "Keywords agrupados" y "Texto agrupado". Luego responda las siguientes preguntas

¿Cuál de estos considera usted que se encuentra agrupado de mejor manera? *

Agrupados por tweet

Agrupados por keywords

Agrupados por texto

No noto ninguna diferencia

¿Le parece útil poder revisar lo que ocurre en su cuenta de Twitter de manera gráfica? *

Si

No

¿Le parece apropiada la cantidad de imágenes que se muestran? *

Si

Debería mostrar más imágenes

Debería mostrar menos imágenes

¿Usaría una aplicación así? *

Si

No

¿Que recomendaciones haría para mejorar esta aplicación?