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

HERRAMIENTAS DE ANÁLISIS DE OPINIÓN EN REDES SOCIALES VIRTUALES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

IVÁN PLIOUCHTCHAI

PROFESOR GUÍA:
NELSON BALOIAN TATARYAN

MIEMBROS DE LA COMISIÓN:
MAURICIO PALMA LIZANA
GUSTAVO ZURITA ALARCÓN

Trabajo parcialmente financiado por Solid IT Limitada

SANTIAGO DE CHILE
MARZO 2014

Resumen

La masividad del uso de las redes sociales ha crecido explosivamente en los últimos años. Resulta interesante conocer la opinión que expresan los usuarios en Twitter para realizar estudios de mercado, popularidad de marcas, candidatos presidenciales, etc. Este trabajo tiene por objetivo desarrollar un software que permita hacer análisis de opinión en Twitter. Este software se utilizó para estudiar la opinión sobre los candidatos a presidente en el año 2013 en Chile.

Se estudiaron dos técnicas utilizadas para obtener el sentimiento asociado a un texto: Método Estadístico y Método Ontológico. El primer método requiere de un gran volumen de datos (textos de los que se conoce si expresan una opinión positiva o negativa) para entrenar el algoritmo. Se eligió el método ontológico, para el que se construyen manualmente reglas para identificar el sentimiento. Para aplicar estas reglas, se procesa el texto libre usando la librería FreeLing, que construye un árbol de dependencia de las palabras que componen el texto. Dicho árbol permite agrupar el sujeto con los correspondientes adjetivos, verbos, etc de las oraciones. La ontología construida consiste en patrones detectables en los arboles de dependencia, con palabras claves que pueden ir en las distintas posiciones del patrón. Hubo problemas con la librería FreeLing que no procesa correctamente texto mal escrito, como es el caso típico de los Tweets. Se tuvo que hacer un preprocesamiento al texto para ayudar a FreeLing a procesar el texto. Al hacer el análisis de los Tweets de los 7 días anteriores a la segunda vuelta, se obtuvo una popularidad del 61 % para Bachelet (obtuvo 62 % en las elecciones) y un 39 % para Matthei (38 % en las elecciones), resultado que también es cercano a las estimaciones de Brandmetrics.

Otra funcionalidad desarrollada es la identificación de la posición geográfica del usuario, y por lo tanto sus Tweets, a partir del dato que él indica en el perfil de usuario. Este es un campo de texto libre. El texto se trata de calzar con una serie de expresiones regulares, que están asociadas con las regiones de Chile. Se validó la técnica desarrollada comparando los resultados obtenidos con los datos por GPS para aquellos Tweets para los que estaban disponibles, obteniendo cerca de un 90 % de acierto. Sin embargo, sólo a alrededor de la mitad de los Tweets se les puede identificar la localidad usando esta técnica, lo que de todas formas es mejor que cerca del 2 % de los Tweets que tienen la información del GPS.

Se analizó también el uso de Twitter en función de la hora del día, observando la máxima actividad en la noche, durante y después de los noticieros.

Tabla de contenido

1. Introducción **1**
- 1.1. Objetivos . . . 3
- 1.2. Metodología . . . 4

2. Estado del arte **5**
- 2.1. Identificación de sentimiento . . . 5
  - 2.1.1. Métodos basados en la cantidad de palabras positivas y negativas . . . 5
  - 2.1.2. Métodos estadísticos . . . 6
  - 2.1.3. Métodos basados en reglas . . . 7
  - 2.1.4. Métodos mediante una ontología . . . 7
  - 2.1.5. Uso de una ontología para analizar texto . . . 8
  - 2.1.6. Comparación de los enfoques de clasificación del sentimiento . . . 9
- 2.2. Intentos de predecir los resultados de elecciones . . . 10

3. Twittermeter: la aplicación a desarrollar **11**
- 3.1. Descripción general del sistema a desarrollar . . . 11
- 3.2. Requerimiento de Usuario . . . 12
- 3.3. Requerimiento de Software . . . 12

4. Desarrollo del trabajo **14**
- 4.1. Elección de la plataforma para la implementación y ejecución del software . . . 14
- 4.2. Obtención de mensajes desde Twitter . . . 15
  - 4.2.1. API de Twitter . . . 15
  - 4.2.2. Refinamiento de los Tweets obtenidos en el stream . . . 16
  - 4.2.3. Implementación del cliente . . . 17
- 4.3. Construcción de Ontología . . . 18
  - 4.3.1. Elecciones presidenciales en Chile . . . 18
  - 4.3.2. Patrones identificados . . . 19
  - 4.3.3. Validación de patrones identificados para la ontología . . . 20
- 4.4. Motor de análisis de sentimiento usando la Ontología construida . . . 21
  - 4.4.1. Librerías de análisis de texto . . . 21
  - 4.4.2. Problemas enfrentados con FreeLing . . . 22
  - 4.4.3. Preparación de texto previo a la librería . . . 23
  - 4.4.4. Motor de reglas definidas en XML . . . 23
  - 4.4.5. Motores específicos para cada tipo de patrón de ontología . . . 24
- 4.5. Identificador de preferencia de voto . . . 27

4.6. Localización geográfica de los Tweets . . . 27
4.6.1. Tweets con GeoTag . . . 27
4.6.2. Localidad en perfil de usuarios . . . 28
4.6.3. Identificación de la localidad del usuario . . . 28
4.6.4. Credibilidad del dato obtenido . . . 29
4.7. Visualización de datos obtenidos . . . 30
4.7.1. Precálculo de resultados . . . 31
4.7.2. Gráficos generados . . . 31
4.7.3. Implementación de la publicación de resultados para el usuario . . . 34
4.8. Diseño de la aplicación construida . . . 37
4.8.1. Arquitectura . . . 37
4.8.2. Modelo de datos . . . 39

5. Análisis e interpretación de resultados obtenidos 41
5.1. Menciones de candidatos . . . 41
5.1.1. Validación de relevancia de Tweets . . . 41
5.1.2. Actividad diaria en Twitter . . . 41
5.2. Segregación geográfica de los datos . . . 43
5.3. Mensajes en apoyo y rechazo . . . 44
5.4. Usuarios que apoyan y rechazan . . . 46
5.5. Comparación con Brandmetrics . . . 47

6. Conclusiones 49
6.1. Posibilidades del uso de los datos recolectados . . . 50
6.2. Trabajos futuros . . . 51

7. Bibliografía 52

Apéndices 56

A. API de FreeLing 56
A.1. Instalación de FreeLing . . . 56
A.2. Procesamiento del texto . . . 57
A.3. Estructura del árbol de dependencia de FreeLing . . . 59

B. Script que copia páginas del servidor a Google Drive 60

C. Datos de validación del método propuesto para identificar la ubicación de los usuarios de Twitter 63

Índice de figuras

1. Gráfico que muestra la cantidad de menciones versus el tiempo. 32
2. Gráfico que muestra la cantidad de menciones de los candidatos el 23 de Noviembre de 2013 en la Región de Antofagasta. 33
3. Gráfico que muestra la cantidad de menciones de apoyo/rechazo de las candidatas de la segunda vuelta entre el 9 y el 15 de diciembre del 2013. 33
4. Directorio de Google Drive con los archivos para el sitio web público. 35
5. Foto de parte de la página web pública del sistema desarrollado. 36
6. Herramienta para comparar localidad obtenida mediante expresiones regulares de información del perfil y mediante GPS. Se muestran los datos para la Región Metropolitana. 37
7. Arquitectura final de la aplicación. Flujo de datos. 38
8. Modelo Relacional de la base de datos. 40
9. Actividad en Twitter sobre los candidatos el 2 de octubre del 2013. 42
10. Gráfico de comparación de la ubicación de los usuarios obtenida desde el perfil y usando GPS. 43
11. Todos los Tweets con GPS recolectados. 44
12. Estimación de intención de voto por las candidatas. 46
13. Tendencia observada por Brandmetrics. 47

Capítulo 1

Introducción

La popularidad de las redes sociales virtuales en los últimos años creció explosivamente aumentando la cantidad de personas que participan en ellas y el tiempo invertido en éstas. Para ser más específico, la cantidad de usuarios registrados en Twitter aumentó en 40 % en el segundo semestre del año 2012, llegando a 288.000.000 usuarios activos al final de dicho período. Gracias a la masificación de los dispositivos móviles, es muy fácil emitir opiniones sobre los temas cotidianos o de contingencia. A fines del 2012 el 57 % de los usuarios activos entraban a Twitter desde dispositivos móviles [1].

Conscientes de la masividad del uso de las redes sociales, y en particular Twitter, muchas empresas, organizaciones y personas intentan llegar al público masivo a través de ellas. Un ejemplo de esto es la cadena de restaurantes Mr. Jack en Chile, que tiene una estrategia de presencia en las redes sociales [2].

Potencialmente, existen otras aplicaciones utilizables por distintos tipos de organizaciones y que tienen relación con el objeto de la presente memoria y cual es el análisis de la opinión pública a través de Twitter, clasificando los mensajes de acuerdo a su carga positiva o negativa. Así, por ejemplo, los canales de televisión podrían contar con una herramienta complementaria al People Meter, que les permitiría contar con mejor información para sus decisiones de inversión.

Para validar las estrategias de marketing en línea usadas por las empresas, ellas buscan medir el impacto que generan sus cuentas de Twitter y Facebook. Existen variados servicios gratuitos y pagados que permiten visualizar de forma cómoda algunas estadísticas sobre la actividad relacionada con ciertas palabras claves o cuentas en Twitter (por ejemplo Tweet Archivist [3], TwentyFeet [4], Socialbakers [5] y Brandmetric [6]).

Las opiniones emitidas por los usuarios pueden tener, o no un sentimiento (positivo o negativo) asociado. El articulo "Twitter Sentiment Classification using Distant Supervision" [7] define el sentimiento cómo "una sensación personal positiva o negativa". Se presentan tam-

bién en ese articulo, ejemplos de Tweets con sentimientos. Positivo: "Jquery is my new best friend.", negativo: "History exam studying ugh." y neutro: "just landed at San Francisco".

Para los efectos de este estudio, clasificará la opinión, reduciéndola a uno de tres tipos de sentimientos: Positivo, Negativo, Neutro. La percepción pública se obtiene mediante el análisis estadístico de las opiniones relativas a un tema.

La herramienta Oracle Rightnow CX Cloud Service [8] tiene la capacidad de obtener mensajes escritos en las redes sociales, blogs y otras fuentes en internet relativos a algunos términos de búsqueda predefinidos. Esta información después puede ser clasificada según el sentimiento del emisor para encontrar aquellos mensajes, que son más relevantes. [9]

Otro uso interesante de las opiniones escritas en Twitter es el intento de predecir los resultados de una votación. Existen estudios como el de Tumasjan et al. [10] y Daniel Gayo-Aveyo [11], que usan Twitter para intentar predecir las elecciones.

Thomas R. Gruber, en su articulo A Translation Approach to Portable Ontology Specifications [12] define "una especificación de un vocabulario de representación para un dominio compartido del discurso - definiciones de clases, relaciones, funciones y otros objetos - se llama ontología".

Dado que Twitter es una red social global, los Tweets escritos por habitantes de distintas áreas del mundo se mezclan. Si se desea obtener la opinión de los habitantes de ciertos lugares geográficos específicos, es necesario usar distintas técnicas para averiguar el lugar desde donde escribe el emisor. Esto es más complejo de lo que se podría pensar ya que Twitter no provee la localización de la gran mayoría de los Tweets en forma transparente.

Se estudiaron en este trabajo distintas formas de implementar una herramienta de obtener la percepción pública geolocalizada sobre ciertos conceptos que tiene la población que participa en la red social Twitter. En particular, se realizó un estudio de las opiniones sobre los candidatos presidenciales en las elecciones en Chile en 2013.

Esto presentó dos desafíos principales:

1. Generar una ontología que permita clasificar un mensaje de Twitter como una opinión positiva, neutra o negativa. Esta técnica evita que el sistema deba ser entrenado.
2. Obtener la geolocalización de Tweets, lo que es necesario para filtrar sólo las opiniones relevantes a un país o localidad. Se hizo un estudio de la confianza de la técnica usada para identificar la localidad de los usuarios de Twitter.

Lo que diferencia lo realizado en este trabajo de título de las herramientas de análisis ya existentes es:

- La gran mayoría de las herramientas de análisis de opinión usan un método estadístico,

y no una ontología. El beneficio de usar esta la técnica con ontología consiste en que no es necesario entrenar el algoritmo con una gran cantidad de datos (los algoritmos estadísticos deben ser entrenados con millones de Tweets para funcionar correctamente), lo que simplifica la puesta en marcha.

- Se implementó una ontología usable en el contexto de las elecciones presidenciales. Esto permite usar el programa en el contexto chileno. Todos los artículos encontrados relacionados con el análisis de sentimiento asociado a un texto se basan en el análisis de texto escrito en inglés.
- Se implementó un sistema de análisis sectorizado geográficamente. Esto para poder comparar las opiniones en distintas regiones o ciudades de Chile. No se han encontrado descripciones ni implementaciones de este tipo de análisis por lo que se trata de una innovación en el área.

Se espera que el presente trabajo, sirva como introducción, o como un primer acercamiento al tema para tener una vista general de lo relativo a minería de datos en Twitter para alguien que quiera ahondar en el tema con el objetivo, por ejemplo, de realizar una tesis de magíster, doctorado, o implementaciones de herramientas de análisis de sentimiento en las redes sociales virtuales.

En el futuro se planea usar la herramienta obtenida en este trabajo para brindar el servicio de análisis de opinión sobre marcas y popularidad de canales y programas de televisión a empresas Chilenas.

1.1. Objetivos

A continuación se listan todos los objetivos establecidos para este trabajo, mencionando ente paréntesis las secciones dónde se explica la forma de cumplir dicho objetivo.

1. Determinar la metodología más apropiada para identificar automáticamente el sentimiento positivo o negativo presente en un mensaje de Twitter (Sección 2.1).
2. Determinar los requisitos mínimos que deberá tener la aplicación que permita visualizar las estadísticas sobre los candidatos, generadas a partir de los mensajes de Twitter (Sección 3).
3. Desarrollar una herramienta para obtener mensajes desde la red social Twitter (Sección 4.2).
4. Desarrollar e implementar la metodología elegida para identificar el sentimiento (Sección 4.3).
5. Investigar e implementar una forma de determinar la ubicación de los autores de los Tweets (Sección 4.6).
6. Disponibilizar los resultados al público (Sección 4.7.3).
7. Estudiar la variación en la actividad de los usuarios para determinar los momentos

idóneos para publicar una información en las redes sociales y así llegar a la mayor cantidad de público (Sección 5.1.2).

8. Comparar los resultados obtenidos con Brandmetrics [13] y con las Elecciones Presidenciales (Secciónes 5.5 y 5.4).
9. Documentar la arquitectura de la aplicación desarrollada para que el trabajo pueda ser continuado fácilmente en el futuro (Sección 4.8).

1.2. Metodología

La metodología de trabajo para cumplir con los objetivos propuestos consistió en:

- El estudio previo general de la problemática y la bibliografía
- La implementación de pequeñas pruebas de concepto que permitieron validar la utilidad de las decisiones tomadas.
- La integración de las componentes nuevas con las demás. Este y el punto anterior se realizó en forma iterativa completando y mejorando la funcionalidad de cada componente del software.
- La generación de gráficos y su análisis

Se comenzó a recolectar datos desde Twitter el 19 de agosto del 2013. Un poco antes de la primera vuelta presidencial, el 8 de noviembre, se publicó una página que permitió ver las estadísticas de las menciones de los candidatos por región (sin el análisis de sentimiento). Antes de la segunda vuelta, se publicó el sistema que mostraba las fracciones de mensajes de apoyo y de rechazo hacia cada una de las candidatas. Actualmente (diciembre del 2013) esta página muestra la información del día de las elecciones, el 15 de noviembre.

Capítulo 2

Estado del arte

Existe una variedad de artículos que describen las técnicas utilizadas para clasificar textos según el sentimiento expresado en ellos.

2.1. Identificación de sentimiento

Se estudiaron distintos métodos que se pueden utilizar para identificar el sentimiento expresado en un texto. A continuación se explican estas técnicas, para luego hacer un análisis comparativo de estas.

2.1.1. Métodos basados en la cantidad de palabras positivas y negativas

Estos métodos requieren de listados de palabras para las que se conoce si estas son positivas, o negativas y la fuerza del sentimiento que expresan. El sentimiento que representa cada palabra se expresa cómo un valor numérico. A continuación, se calcula la puntuación de un trozo de texto sumando los valores de sentimiento de cada una de las palabras que lo componen. El valor obtenido en este cálculo representa la positividad, neutralidad o negatividad de una opinión. Esta técnica es aplicada por Yu y Hatzivassiloglou [14] para identificar si un texto describe un hecho, o expresa una opinión.

Una técnica similar es utilizada por Wiebe [15] para identificar la subjetividad u objetividad de una oración, basandose en si esta contiene, o no, al menos un adjetivo subjetivo. Esto,

finalmente, permite identificar si lo expresado en un texto es una opinión, o la descripción de sucesos, o hechos.

2.1.2. Métodos estadísticos

Se han encontrado y estudiado distintos artículos que detectan el sentimiento de los Tweets basándose en algoritmos entrenados con grandes volúmenes de datos.

Los métodos estadísticos de evaluación del sentimiento de un texto se basan en algoritmos entrenados sobre una gran cantidad de textos (por ejemplo Tweets) de los que se sabe el sentimiento asociado. Por ejemplo, se usan redes neuronales para estos efectos. Más adelante, en la sección 2.1.6, se discutirán las ventajas y desventajas de este enfoque.

El artículo "Thumbs up? Sentiment Classification using Machine Learning Techniques" de Pang et al. [16] es uno de los primeros intentos de usar un algoritmo entrenado para clasificar los Tweets según el sentimiento. El entrenamiento de los algoritmos se hace usando reseñas sobre las películas. Estas reseñas tienen asociadas calificaciones (en cantidad de estrellas) que son un indicador de que tan positivo o negativo es el sentimiento del autor que escribió la reseña.

El artículo "Twitter Sentiment Classification using Distant Supervision" de Alec Go et al. [7] se basa en el articulo anterior, pero tiene la innovación de entrenar el sistema usando Tweets y determinando en forma automática los sentimientos usando las caritas felices ":)" o tristes ":(" (y sus variantes). Los resultados obtenidos por estos autores llegan a alrededor del 80 % de efectividad. La complejidad de la técnica usada por Go, Bhayani y Huang es la necesidad de entrenar el el sistema con una gran cantidad de datos (ellos usaron 1,600,000 Tweets). El articulo es de interés pues en el se explican las técnicas utilizadas para limpiar los Tweets y así reducir sus características. Por ejemplo, se pueden reemplazar todas las "menciones" (nombres de usuario de otros participantes de la red antecedidos con un '@') y los URL por un *placeholder*. Con esto se limpia el texto de las palabras que a priori se sabe que no conllevan un sentimiento asociado, pues corresponde un sujeto en la oración.

Albert Bifet y Eibe Frank en su articulo llamado "Sentiment knowledge discovery in twitter streaming data" [17] hablan sobre la forma de trabajar con la API de Twitter y hacen un estudio de la efectividad de los algoritmos entrenados para el análisis de sentimiento, obteniendo al rededor de un 80 % de efectividad. Además proponen el uso de una ventana de tiempo, en la que se calculará la nota de sentimiento. Esto es necesario pues al existir un flujo constante de Tweets se desea evaluar la evolución del sentimiento respecto de los distintos términos de interés.

La gran mayoría de los estudios encontrados se basan en el artículo "Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews" de Peter D.

Turney [18], en el se explica la forma de poner una nota a un texto en función de la aparición de palabras positivas y negativas en la cercanía de las palabras que describen el objeto de interés.

La técnica usada en estos artículos es diferente a la propuesta para este trabajo, que implica generar y usar una ontología.

2.1.3. Métodos basados en reglas

Existen variados métodos basados en distintos tipos de reglas que se deben aplicar a un texto, o parte de el para clasificarlo. En general, cualquier método que use algún tipo de reglas definidas manualmente puede considerarse cómo un método de esta familia.

Un ejemplo de método basado en reglas es el descrito por Kan en su exposición en la conferencia ROMIP 2011 [19], que obtiene una calificación para un texto completo, sin considerar los sujetos, considerando palabras positivas y negativas, junto a ponderadores asociados a negaciones y conjunciones.

Otro método, que está explicado en el artículo "Sentiment Analysis: A Combined Approach" de Prabowo y Thelwall [20] identifica sujetos en una oración e intenta aplicar reglas de secuencias de palabras al mismo texto para obtener una calificación definida por dicha regla. Un ejemplo de oración es: "Laptop-A is more expensive than Laptop-B.". La regla que identifica con un sentimiento negativo al sujeto Laptop-A de la oración es:

$$\{\# \wedge more \wedge \text{e}xpen\text{s}ive \wedge than \wedge ?\} \Rightarrow \{-\}$$

Este método también toma en cuenta las negaciones para invertir el sentido de las palabras.

2.1.4. Métodos mediante una ontología

Un enfoque similar al de las reglas para saber si un texto expresa una opinión positiva o negativa es el de usar una ontología. Se trata de tener un diccionario que relaciona términos y contextos junto con el dato de si la palabra es positiva o negativa en dicho contexto.

El trabajo "Ontology based combined approach for Sentiment Classification" de Shein Khin Phyu Phyu [21] explica un procedimiento que permite obtener el sentimiento asociado a una opinión sobre una película mediante el uso de una ontologia. El procedimiento explica la necesidad de separar el texto en cada una de las oraciones que lo componen, para luego

analizar cada una de ellas. En cada oración se identifican las partes de ella, así cómo los nombres, verbos y adjetivos. Esta información es usada para identificar si la oración pertenece al dominio de interés, para finalmente buscar palabras claves que permitan identificar el sentimiento asociado al sujeto de la oración. El autor compara su algoritmo con uno similar, pero que no identifica si la oración pertenece al dominio de interés. Los resultados obtenidos con la técnica propuesta por Khin Phyu Phyu muestran una mejora en los aciertos llegando a al rededor del 85 %.

"Sentiment Learning on Product Reviews via Sentiment Ontology Tree" de Wei y Gulla [22] explica la importancia de tener un Arbol de Ontología de Sentimientos, que consiste en relacionar conceptos, de los más generales a los más específicos en forma de un árbol para identificar el tema tratado en un trozo de texto. Por ejemplo, para el caso de una cámara fotográfica, la calidad de la imagen es un subconcepto de ella. Si se logra identificar en el texto la mención positiva de la calidad de imagen, solamente se podrá asegurar que la opinión es positiva hacia la cámara si se logra identificar en el texto que dicha calidad de imagen está en el contexto de una cámara fotográfica. El uso de esta técnica permite descartar opiniones que si bien tienen palabras claves que se pueden relacionar con el tema de interés, no se refieren a ese tema.

Se ha encontrado un sólo artículo que propone el uso de una ontología para la obtención del sentimiento de un Tweet. Se trata de "Ontology-based sentiment analysis of twitter posts" de Kontopoulos et al. [23]. Este artículo es muy reciente, habiendo sido publicado en el año 2013. Los autores usan la ontología y un análisis semántico del texto para poder clasificar el sentimiento de Tweets como "The screenplay was wonderful, although the acting was rather bad" donde están presentes dos sujetos a los que se asocian dos sentimientos opuestos. El análisis se hace sobre Tweets escritos en inglés y usando un servicio de análisis de sentimientos en la nube llamado OpenDover. Además, en este artículo se deja de lado la georeferencia de los Tweets.

2.1.5. Uso de una ontología para analizar texto

Existen variadas librerías que permiten simplificar y estructurar un texto escrito en lenguaje fluido, para que este sea procesable por un programa. Una de las librerías es FreeLing [24], que promete un buen soporte del idioma castellano. Esta librería es capaz de identificar las partes de una oración para identificar los sujetos, los verbos, adjetivos, etc. Ademas, construye un árbol de dependencia entre las palabras, que permite identificar que palabras están relacionadas entre ellas en una oración y cuales no. En particular, permite identificar qué verbos y adjetivos están asociados a qué sujetos.

2.1.6. Comparación de los enfoques de clasificación del sentimiento

El enfoque basado en calcular la puntuación de una oración mediante la suma de los sentimientos asociados a cada una de las palabras que la componen no permite identificar el sujeto al que se refiere la opinión, resultando completamente inútil en caso de que la oración contenga más de un sujeto de interés. Esta técnica tampoco considera la negación de parte de las palabras de una oración, lo que puede cambiar el sentimiento identificado de la oración a uno diametralmente opuesto al real.

Para el óptimo funcionamiento del enfoque estadístico, el algoritmo debe ser entrenado con un gran volumen de datos. La cantidad necesaria de datos puede ser difícil de conseguir. Este método es general y tolerante a cambios de temas, permitiendo analizar textos de distintas temáticas sin tener que reconfigurar o reentrenar el sistema. Un cambio importante del contexto o del idioma de los textos a analizar requiere volver a entrenar el algoritmo.

Un algoritmo basado en una ontología, en cambio, no debe ser entrenado. La ontología se construye manualmente considerando el contexto e idioma del texto. Esto reduce el tiempo de puesta en marcha del sistema. Si cambia el contexto o idioma de los textos a analizar, será necesario ajustar o rehacer la ontología.

El método estadístico es normalmente entrenado con datos generados por muchas personas (que evaluaron manualmente la positividad o negatividad del sentimiento en muchos textos). Gracias a esto el algoritmo genera un resultado similar al de una persona 'promedio'. En el caso de la ontología, esta se construye por un equipo reducido de personas (o una sola persona), por lo que los resultados generados por el algoritmo pueden estar cargados hacia la opinión personal del que desarrolló la ontología.

Las diferencias principales entre el método basado en reglas y en ontología está en que el método basado en reglas no necesariamente considera al sujeto de la frase u oración; el método basado en ontología se basa en construir un árbol de dependencia para identificar las partes léxicas de una oración, lo que permite relacionar los adjetivos con los sujetos y tener una garantía de que el contexto de las palabras es el adecuado. Reduciendo de esta forma los falsos positivos y negativos que se puedan identificar.

Para este trabajo se opta por usar una ontología por ser una técnica menos usada en mensajes cortos cómo los de Twitter y por existir pocos datos (pocos Tweets en español chileno) para poder entrenar satisfactoriamente un algoritmo estadístico.

2.2. Intentos de predecir los resultados de elecciones

Variados artículos científicos describen los intentos de predecir los resultados de las elecciones usando los mensajes en Twitter. Todos ellos han mostrado la imposibilidad de hacer una predicción confiable.

Todos los estudios encontrados se basan en el método estadístico para evaluar la positividad o negatividad del sentimiento asociado a los Tweets.

Tumasjan et al. [10] en su artículo explican que usaron palabras claves que expresan sentimientos para identificar si un Tweet tiene una opinión positiva o negativa, sin embargo, no relacionaron a qué sujeto se refiere ese sentimiento. Estos autores también identificaron que existen usuarios que son más o menos interesados en la politica, algunos publicando una sola vez, unas pocas veces, llegando a más de 80 Tweets por usuario, en una muestra de al rededor de 70,000 Tweets.

Un articulo muy interesante es el de Daniel Gayo-Avello llamado "I Wanted to Predict Elections with Twitter and all I got was this Lousy Paper" [11]. Aquí se explica por qué es imposible predecir las elecciones usando Twitter. Los principales argumentos son que nadie ha logrado predecir una elección usando Twitter, sino que todos los análisis existentes se hacen a posteriori, muchas veces ajustando e interpretando los resultados para obtener un resultado positivo. Se ignoran los factores demográficos de los que escriben, ya que los usuarios de Twitter tienen un perfil específico: acceso a internet, a tecnología, cierto rango etario y de educación, etc. Por otro lado, el artículo propone un conjunto de reglas a las que hay que atenerse en caso de querer predecir una elección: definir como se cuentan los votos, es decir, cómo se decide por quién va a votar qué persona; poder filtrar correctamente la propaganda y la desinformación, etc.

Es necesario recalcar que el presente trabajo no tiene como objetivo predecir los resultados de las elecciones presidenciales 2013 en Chile, sino que sólo se pretende usar el tema ya que los Tweets relacionados muchas veces tienen asociado un sentimiento positivo o negativo marcado.

Capítulo 3

Twittermeter: la aplicación a desarrollar

Para cumplir con los objetivos establecidos para el presente trabajo, se decidió implementar una aplicación que permita recolectar la información, procesar y visualizarla. En este capítulo se explican las características buscadas en este software.

3.1. Descripción general del sistema a desarrollar

Se quiere implementar un sistema que permita hacer análisis de los mensajes que mencionan conceptos (sujetos) comparables entre ellos. Estos sujetos pueden ser marcas comerciales, canales de televisión, eventos, personas, etc. En este caso particular, el sistema debe estar orientado al análisis de las menciones de los candidatos a presidente de Chile para el período 2014 - 2018, cuyas elecciones se realizaron en noviembre y diciembre del año 2013.

Las características buscadas incluyen:

- El sistema debe ser capaz de mostrar en gráficos cómo varía la cantidad de menciones de los candidatos en el tiempo.
- El sistema debe ser capaz de mostrar la fracción de mensajes positivos (de apoyo) y negativos (de rechazo) hacia cada uno de los candidatos.
- El sistema debe mostrar la fracción de usuarios que prefieren a cada uno de los candidatos.
- Debe existir un gráfico que permita ver la cantidad de mensajes emitidos mencionando a alguno de los candidatos desde las distintas regiones de Chile.
- Se desea tener una página web pública que permita ver estas estadísticas.

3.2. Requerimiento de Usuario

Requerimientos funcionales:

1. Debe existir una manera de obtener mensajes relacionados con los candidatos presidenciales para la elecciones en Chile para el año 2014 desde la red social Twitter.
2. Debe existir una serie de páginas web que permitan ver distintos gráficos de los datos obtenidos y generados.
3. Debe existir un gráfico que muestre para cada candidato un histograma de menciones durante un período de tiempo, en que cada punto represente la cantidad de menciones en un período de tiempo corto.
4. La página debe mostrar la cantidad de opiniones positivas y negativas por cada candidato
5. La página debe mostrar la fracción de usuarios de Twitter que apoyan a cada uno de los candidatos (preferencia de voto)
6. La página deberá mostrar la fracción de menciones de los candidatos por región de Chile, en un período.

Además se desean los siguientes requerimientos no funcionales:

1. La página web debe ser rápida y usable.
2. Se debe estudiar la confiabilidad de los datos de región obtenidos.
3. Se debe estudiar la confiabilidad de los datos de opinión obtenidos, comparándolos con una muestra analizada manualmente.
4. El sistema debe considerar todos, o la gran mayoría de los Tweets emitidos mencionando a los candidatos.

3.3. Requerimiento de Software

El software a desarrollar debe tener las siguientes funcionalidades:

1. Sistema de obtención de Tweets: Se deberán recolectar Tweets relacionados con los 9 candidatos de la primera vuelta y los 2 de la segunda vuelta.
2. Identificar al autor de cada uno de los Tweets.
3. Georreferenciar el Tweet cuando sea factible (posición GPS).
4. Almacenar los datos obtenidos de manera local
5. Procesar los Tweets identificando la opinión del usuario como positiva, negativa o desconocida (neutra).

6. Páginas web que permitan visualizar distintos gráficos.
7. Un sistema que genere estadísticas periódicamente para ser usadas en los gráficos de las páginas para el usuario.
8. Generación y visualización de gráficos de cantidad de menciones por candidato versus el tiempo.
9. Generación y visualización de gráficos de cantidad de opiniones positivas y negativas sobre los candidatos.
10. Generación y visualización de un gráfico que muestre la preferencia de voto de los usuarios de Twitter que han escrito sobre algún candidato en un período determinado. Se deberá consolidar la opinión de todos los Tweets de un usuario para obtener su preferencia de voto.
11. Página y gráficos para ver la cantidad de menciones de los candidatos por región de Chile durante un día.
12. Identificar la ubicación (región) de un Tweet en función de la información del perfil del usuario que lo emite.
13. Página que permita comparar datos obtenidos por adivinador de región del usuario con los datos del GPS en caso de existir.

Capítulo 4

Desarrollo del trabajo

Este trabajo se realizó en forma iterativa implementando primero prototipos minimos que demostraban la viabilidad de la solución elegida, haciendo correcciones en caso de ser necesario. Con el tiempo la funcionalidad se fué mejorando y completando para cumplir con los objetivos propuestos.

4.1. Elección de la plataforma para la implementación y ejecución del software

Se consideraron distintas opciones de plataformas para la implementación de la aplicación requerida. Se deseaba implementar una interfaz web para poder visualizar los datos, por lo que las principales opciones consideradas fueron Python, PHP y Java EE, que permiten implementar aplicaciones web fácilmente.

La primera opción descartada fué PHP ya que es una plataforma poco eficiente para el procesamiento de grandes volúmenes de datos, estimándose un overhead elevado agregado por la plataforma en la ejecución de la aplicación.

Tanto Python, como Java EE tienen librerías que permiten abstraerse del acceso y manejo de la capa de persistencia, lo que simplifica la implementación de las componentes que manejan los datos generados por la aplicación. Además existen implementaciones de APIs para acceder a Twitter para ambos lenguajes de programación.

El factor determinante para elegir la plataforma fué la mayor experiencia personal en el uso de Java, Java EE y JBoss, por lo que esta fue la plataforma finalmente elegida. Se estimó que con esta decisión se evitaría malgastar el tiempo en aprender algunas funcionalidades de

Python con las que se tiene menos experiencia.

Para la persistencia de los datos obtenidos desde Twitter se decidió usar una base de datos PostgreSQL.

Se definió que el software se ejecutará en un PC disponible basado en un procesador Intel Atom y 4 GB de RAM, con Ubuntu Linux como sistema operativo.

4.2. Obtención de mensajes desde Twitter

Una parte absolutamente necesaria para cumplir con los objetivos definidos es poder recolectar los mensajes desde Twitter. Para poder usar los datos de esta red social, se estudió la API pública existente, se eligió el enfoque que se iba a tomar y se planificó cómo se iban a refinar los Tweets para utilizar sólo aquellos que son relevantes para el tema de las elecciones presidenciales en Chile del año 2013.

4.2.1. API de Twitter

Twitter permite obtener Tweets de dos formas distintas: Usando búsquedas y escuchando un stream.

4.2.1.1. Búsqueda

La búsqueda a través de la API de Twitter es bastante limitada. Permite obtener sólo hasta mil resultados, los que se deben obtener en bloques (compaginamiento). Por esto, no es posible obtener una gran cantidad de datos para analizar. Debido a la inalcanzabilidad de muchos Tweets relevantes, esta opción se descartó.

4.2.1.2. Stream

Twitter permite suscribirse a un stream de Tweets que incluyen las palabras de interés deseadas. Para esto es necesario abrir una conexión con los servidores de Twitter y suscribirse a los términos buscados. Twitter envía los Tweets nuevos que contienen estos términos por dicha conexión en formato JSON. La cantidad de mensajes que se pueden obtener tienen un

límite para evitar usos no justos del sistema. En caso de exceder un límite de Tweets en un tiempo, se recibe un mensaje especial desde Twitter indicando que no se han recibido todos los mensajes generados en el sistema.

En el foro oficial para desarrolladores con Twitter [25] se puede encontrar información que dice que en caso de que los términos comprendan menos del 1 % de todos los Tweets del sistema, se recibirán todos ellos. En caso de superarse este limite, se recibirá sólo como máximo el 1 % de los Tweets (del universo) seleccionados al azar. En términos prácticos, esto significa que si se filtra por términos específicos y no muy comunes, se recibirán todos los Tweets que incluyen estos términos.

La API de Twitter exige tener un conjunto de tokens públicos y secretos para suscribirse a un stream y permite una sola conexión por dirección IP y por aplicación. Esto implica que es necesario suscribirse a un stream que incluya todos los términos de interés y, de ser necesario, separar los Tweets según los términos que incluyen en forma programática dentro de la aplicación. Esto descarta la posibilidad de tener distintos streams, uno por cada candidato, conectados a la aplicación. El temor de superar el 1 % de Tweets no se se justificó pues la única vez que este límite fue superado fue durante uno de los debates presidenciales televisados y los dias de la primera y segunda vuelta de las elecciones.

4.2.2. Refinamiento de los Tweets obtenidos en el stream

El stream de Twitter entrega una cantidad considerable de Tweets que no son relevantes. Estos se pueden caracterizar por contener los términos buscados pero en un contexto diferente al esperado. Por ejemplo, al buscar por los apellidos de los candidatos aparecen Tweets relacionados a otras personas conocidas que tienen el mismo apellido (El candidato *Claude* y el actor *Jean-Claude Van Damme*).

Existen distintas formas de descartar Tweets poco relevantes en forma automática:

- Tweets que contengan ciertas palabras claves (lista negra). Se implementó esta funcionalidad obteniendo mejores resultados. Sin embargo, la lista negra tiende a crecer mucho y a hacerse muy específica, por lo que este método por sí mismo no es suficiente.
- Tweets que no estén escritos por una persona que hable en un idioma relevante. Twitter entrega la posibilidad de obtener el idioma del usuario que escribió un Tweet. Si el idioma del usuario no es el español, su Tweet puede ser descartado. Se implementó este filtro. Efectivamente se descartan los Tweets escritos en idiomas distintos al castellano. Esta medida mejora la relevancia de los mensajes, pero se pierden algunos Tweets que sí se refieren al tema deseado. Se estima que esta pérdida fué de menos de un 10 %.
- Descartar Tweets según la localidad del usuario. La API de Twitter permite saber la localidad del usuario. Esta es la localidad que el usuario configura voluntariamente en su perfil. Se trata de un campo de texto libre. Se ha observado que una gran cantidad de

usuarios son honestos al configurar este dato. En el caso de Chile, los usuarios indican 'Chile', 'Santiago', 'Stgo', 'Valparaiso', 'Valpo', etc. Este filtro fué descartado en un comienzo debido al tamaño del diccionario que se debía tener para filtrar adecuadamente los Tweets. Aún teniendo este diccionario, se descartaría más de la mitad de todos los Tweets que sí son relevantes para el tema de interés.

- Tweets que estén geotageados y el geotag no pertenezca a Chile. Esta funcionalidad no se implementó. Sin embargo, la pequeña cantidad de Tweets geotageados hace pensar que la utilidad de este filtro es baja.
- Tweets que estén escritos por personas no Chilenas. Para esto se debe tener una lista de los usuarios Chilenos de Twitter. Para generar dicha lista se pueden buscar aquellos usuarios que sean seguidores de ciertas cuentas de Twitter típicas para Chile, como algunas cuentas de noticias, o personas de interés. Además estos usuarios deben ser seguidos por una cantidad considerable de otros usuarios que se sabe que son Chilenos. Esta funcionalidad no se implementó, sin embargo es viable y se conocen implementaciones que han resultado exitosas .
- Otra forma de identificar a usuarios Chilenos, o con interés en Chile es revisando el historial de Tweets de dicho usuario en búsqueda de Tweets geotageados, o check-ins en foursquare. Esta forma no fue implementada.

4.2.3. Implementación del cliente

Existen implementaciones en distintos lenguajes de programación que usan la API de Twitter para obtener información desde esta red social. En particular se usó la implementación de la API de Twitter para Java llamada Twitter4J. Esta librería oculta el proceso de creación de la conexión, la autenticación y la suscripción al stream además de implementar otras funcionalidades como obtener la información del perfil de un usuario, realizar búsquedas historiales en Twitter, etc.

La librería fue usada satisfactoriamente en la implementación del recolector de Tweets simplificando el proceso de comunicación con Twitter de forma considerable.

4.2.3.1. Palabras relevantes e irrelevantes

Se definieron 3 listas de términos en la aplicación para obtener los mensajes desde Twitter e identificar si son relevantes, o no para el tema deseado.

- **Términos relevantes suscritos**: Estos son términos relacionados con cada uno de los candidatos a presidente. Se agregan a los términos a los que se suscribe la aplicación para recibir todos los Tweets que los contengan. Esta lista incluyó los nombres y apellidos de los candidatos, nombres de las cuentas oficiales en Twitter de los candidatos, hashtags

populares que se usaban para referirse a los distintos sujetos del estudio y sobrenombres con los que las personas se referían a ellos.

- **Términos relevantes no suscritos**: Estos términos son relevantes, pero no se agregan a la lista de suscripción del stream. Esto porque pueden ser palabras comunes que harían que llegarán muchos Tweets no relevantes. Sin embargo, la aplicación relaciona cada una de estas palabras con alguno de los sujetos de interés. Esta lista incluyó los nombres de pila de los candidatos y algunos sobrenombres que no son lo suficientemente únicos para poder suponer que se refiere a alguno de los candidatos.
- **Lista negra**: Estas son palabras que no se incluyen en la lista de suscripción, pero su presencia en alguno de los Tweets recibidos implica que dicho mensaje no es relevante para la aplicación y debe ser descartado. Esta lista se fue construyendo con el tiempo observando los Tweets obtenidos. Se incluyeron palabras que no son relevantes para el tema específico que se deseaba estudiar, pero que típicamente estaban presentes en los mensajes no relevantes. Por ejemplo, se incluyó "Van Damme" para descartar los Tweets referidos al actor y que el sistema interpretaba como referentes al candidato Claude, u"Obama" para descartar Tweets sobre Michelle Obama, que se asociaban con la candidata Bachelet.

4.2.3.2. Almacenamiento de los datos

Los Tweets relevantes obtenidos se fueron almacenando en una base de datos, guardando además los metadatos de interés: Nombre del usuario, Ubicación GPS, Localidad declarada en el perfil de usuario, fecha y hora. La sección de Arquitectura muestra el modelo de datos utilizado.

4.3. Construcción de Ontología

Se construyó una ontología específica para las elecciones presidenciales en Chile en el año 2013. Esta consiste solamente en las reglas que permiten identificar si un mensaje de Twitter tiene un sentimiento positivo o negativo.

4.3.1. Elecciones presidenciales en Chile

Para comprender correctamente la construcción de la ontologia, es necesario conocer el contexto en el que se enmarca. Durante el año 2013 se llevaron a cabo las elecciones de presidente para gobernar el país entre los años 2014 y 2018.

La primera vuelta de las elecciones tuvo 9 candidatos: Franco Parisi, Marcel Claude, Ricardo Israel, Marco Enríquez-Ominami, Roxana Miranda, Michelle Bachelet, Evelyn Matthei, Alfredo Sfeir y Tomás Jocelyn-Holt.

La segunda vuelta tuvo a Michelle Bachelet y Evelyn Matthei como candidatas.

4.3.2. Patrones identificados

Revisando manualmente los Tweets, se observó que muchos de sus textos que tenían un marcado sentimiento positivo o negativo hacia el candidato caían en una de las categorías explicadas a continuación.

Existe una infinidad adicional de patrones, pero estos son más complejos y se consideró poco eficiente implementarlos ya que son muy específicos, no reutilizables y aplicables a muy pocos mensajes.

4.3.2.1. Sinónimo con sentimiento

Estos son principalmente los hashtags que expresan claramente el apoyo o rechazo hacia un candidato:

- #YoVoto7 - Apoyo hacia la candidata Evelyn Matthei
- #ChaoEvelyn - Rechazo hacia la candidata Evelyn Matthei
- #JovenesXBachelet - Apoyo hacia la candidata Michelle Bachelet
- #YoVotoMEO - Apoyo hacia el candidato Marco Enríquez-Ominami
- etc.

4.3.2.2. Patrón lineal

En los árboles de dependencia descritos en la sección 2.1.5 se pueden identificar patrones "lineales", como por ejemplo, "Vamos Bachelet", donde el verbo "ir" queda como padre del sujeto "Bachelet". Se implementó un diccionario de palabras que al estar asociadas con cualquiera de los sinónimos que se pueden asociar a un candidato, representan un sentimiento positivo o negativo hacia este.

Algunas reglas de este patrón:

Ejemplo de texto	Infinitivo de palabra clave	Sentimiento
Vamos Parisi	ir	positivo
Voto Claude	votar	positivo
Bachelet mentirosa	mentir	negativo

4.3.2.3. Patrón Verbo-Unión

Esta es una forma donde un verbo une una palabra con alguno de los sujetos (candidatos). Por ejemplo, "Yo voto MEO", al ser convertido en un árbol de dependencia queda con el verbo "votar" como la raíz, y "yo" y "MEO" como los hijos. Este, claramente, es un Tweet de apoyo hacia el candidato Marco Enríquez-Ominami.

Ejemplos de esta regla:

Ejemplo de texto	verbo	palabra	Sentimiento
Me sumo a Claude	sumar	me	positivo
Sfeir es el mejor!	ser	mejor	positivo
Matthei lo hace mal	hacer	mal	negativo

4.3.2.4. Negación

Es común identificar alguno de los patrones anteriormente explicados pero en forma negada. Generalmente la negación queda como uno de los ancestros del nodo donde comienza el patron. En caso de encontrarse un "no", "nunca", etc, se debe invertir el sentimiento identificado de la frase.

4.3.3. Validación de patrones identificados para la ontología

Para validar el criterio utilizado para generar la ontología, se hizo una selección de 20 Tweets relacionados con los candidatos y se encuestó a usuarios de Internet sobre si estos Tweets tenían una opinión expresada, si esa opinión era positiva o negativa y el nombre del candidato. La encuesta fue respondida por 35 personas. Al rededor del 90 % de los encuestados coincidieron en sus respuestas.

Se comparó el resultado de esta encuesta con el resultado esperable al usar la presente ontología, obteniendo un 20 % (4 Tweets) a los que sería imposible detectar un sentimien-

to debido a que la estructura de dependencia es más compleja que las consideradas para la construcción de esta ontología. El resultado esperable de 5 Tweets (25 %) no coincidió pues los Tweets incluían sarcasmo, elemento incontrolable y no considerada en este trabajo. Finalmente, los demás Tweets coincidieron, tanto para aquellos que no tenían una opinión expresada, como para las opiniones positivas y negativas. Este resultado esta dentro de lo esperable y valida el criterio utilizado para crear la ontología.

4.4. Motor de análisis de sentimiento usando la Ontología construida

Para automatizar la identificación de sentimiento en los Tweets, se usa la ontología construida. Esta ontología consiste en reglas que son aplicadas mediante una pieza de software para obtener el resultado en forma automática. Esta sección explica la implementación de este módulo.

4.4.1. Librerías de análisis de texto

Para analizar el texto mediante una ontología, es necesario procesar el texto para obtener sus partes lingüísticas como el sujeto y el predicado.

Existen librerías que implementan esta funcionalidad. Se instalaron, estudiaron y probaron dos librerías de análisis lingüístico: GATE [26] y FreeLing [24]

4.4.1.1. GATE

Se trata de un software libre que permite desarmar texto plano. Sin embargo, su soporte para el idioma castellano es muy limitado. Es posible desarrollar las reglas para el idioma que se desee, pero esto es muy trabajoso y quedaría fuera del alcance de la presente memoria.

4.4.1.2. FreeLing

Es un software libre hecho en C que puede funcionar como una librería (existe un *wrapper* para java) o como un servidor. La librería recibe como entrada texto plano y entrega como resultado el mismo texto desarmado identificando las partes de las oraciones. Esta librería

incluye soporte para castellano por defecto y ha mostrado eficacia en las pruebas de concepto realizadas. Por esta razón se eligió esta librería para la implementación del módulo que ejecuta la ontología.

4.4.1.3. Elección de librería

Debido a que FreeLing ya está correctamente configurado y entrenado para funcionar con el lenguaje castellano, se decidió usar esta librería. El procedimiento de instalación y API de esta librería se describe en el apéndice A.

4.4.2. Problemas enfrentados con FreeLing

En un comienzo se usó la librería en una aplicación *standalone* Java, pudiendo procesar texto ingresado sin problemas.

A continuación, se intentó integrar la librería con el servidor JBoss para analizar los mensajes a penas estos llegan al sistema. Este intento no resultó debido a la complejidad de configuración del ambiente donde se ejecuta la librería. Se deben declarar algunas variables de entorno para que FreeLing funcione correctamente. Y no se encontró una forma viable de hacer esto en JBoss.

Para ahorrar tiempo, se decidió hacer una aplicación separada, que tome los Tweets desde la base de datos y los procese, guardando los resultados para que puedan ser accedidos por los módulos de visualización de estadísticas.

En condiciones reales, donde los Tweets son escritos con errores ortográficos, usando abreviaciones, sin usar puntuación adecuada, uso incorrecto de mayúsculas y minúsculas y con símbolos no alfanuméricos, la librería FreeLing tuvo muchos problemas para *parsear* el texto y generar los árboles de dependencia. Para resolver este problema, se aplicó un preprocesamiento al texto antes de pasárselo a la librería e implementando un sistema de detección de falla de la librería, que ignora el resultado y marca el Tweet en la base de datos para no volver a revisarlo.

En un principio se planeaba usar la información de las etiquetas EAGLES entregadas por FreeLing para obtener el sujeto, y el adjetivo de la oración, pero en pruebas reales esta información resultó completamente inservible debido el gran desacierto de la librería para obtener estos datos. Esto, probablemente, se debe al aún no suficientemente completo diccionario que usa la librería FreeLing. Finalmente, la etiqueta se ignoró y se usaron las palabras en su forma básica ($wordLemma$) sin importar si eran usadas como sujeto, verbo, adjetivo, etc.

4.4.3. Preparación de texto previo a la librería

Para evitar que FreeLing se confunda con los elementos extraños que la gente escribe en el texto, como por ejemplo, caritas felices, direcciones de internet, menciones a otros usuarios, saltos de línea, etc. se aplicó un conjunto de conversiones al texto para dejarlo en una forma más correcta. Las modificaciones se detallan a continuación:

- Se agregó un espacio después de los símbolos ‘,’ (coma), ‘:’ (dos puntos), ‘;’ (punto y coma), ‘.’ (punto).
- Se reemplazaron los símbolos ‘#’, ‘@’, ‘”’ (comilla simple), ‘”’ (comillas dobles) por espacios.
- Se reemplazaron los paréntesis de llave y corchetes ‘[]’ por paréntesis redondos.
- Se quitaron paréntesis repetidos. Por ejemplo, “((” se transformó en “(”.
- Se reemplazaron las URL de internet por un placeholder “URL”.
- Se reemplazaron las caritas felices y tristes por un espacio.
- Se quitaron puntos y espacios repetidos, reemplazandolos por uno solo.
- Se quitaron todos los saltos de línea.
- Finalmente, se agregó un punto final en caso de no existir y un salto de línea. Con esto se indica a FreeLing que el texto se terminó.

4.4.4. Motor de reglas definidas en XML

En un comienzo se decidió implementar un sistema flexible para programar las reglas de la ontología. Este sistema describe cada regla como un XML, que representa un árbol que tiene la misma forma que el patrón en el texto buscado. Cada nodo del XML representa una o varias palabras que se deben buscar en el nodo del árbol que describe el Tweet. Los nodos internos de cada nodo del XML representan a los hijos. Cada nodo, además, indica si la búsqueda del termino debe hacerse solamente en el nodo inmediato, o descendiendo por el árbol en forma recursiva hasta encontrar la palabra o una hoja que no representa la palabra. Si se logra calzar la plantilla con el árbol del Tweet, se le asigna el valor correspondiente a esa regla al texto respecto del sujeto identificado.

A continuación, un ejemplo de regla definida en XML:

```
<Rule name="Verbo unión yo v sujeto" value="1">
   <Node recursive="True">
      <word>votar</word>
      <word>apoyar</word>
      <word>felicitar</word>
```

```
      <Node recursive="True">
         <Subject/>
      </Node>
      <Node recursive="False">
         <word>yo</word>
      </Node>
   </Node>
</Rule>
```

La impelementación de esta motor de ontología resultó bastante complejo y poco eficiente, ya que permitía mucha flexibilidad en las reglas definidas. Por eso, se optó por implementar una forma más simple del motor, que sólo soporte los tres tipos de reglas de ontología ya explicadas.

4.4.5. Motores específicos para cada tipo de patrón de ontología

Como se mencionó en la sección anterior, se optó por implementar un motor más simple, que sólo soporte un conjunto limitado de tipos de reglas, pero más eficiente.

Para este motor se optó por describir las reglas en un archivo, en que los datos están separados por línea, cada línea del archivo describe una regla y sus partes son separadas por ';'. El motor, luego de obtener el árbol de dependencia, llama los métodos que implementan las reglas con el árbol como parámetro. El retorno de estas funciones es un objeto que tiene la información de las reglas que calzaron con el árbol, el sujeto y el sentimiento encontrado.

4.4.5.1. Objeto de resultado del análisis

Los sentimientos encontrados por los distintos métodos que buscan el sentimiento en el texto se guardan y retornan en un objeto especial de la clase *SentimentReport*.

Los atributos de la clase son:

- subject: Contiene una representación del sujeto al que se refiere el texto. En este caso, el o la candidata de las elecciones.
- ruleName: Un String con el tipo de la regla que identificó el sentimiento.
- sentiment: Un número de tipo d*oubl*e que contiene el valor del sentimiento encontrado. Un valor mayor que cero indica un sentimiento positivo o de apoyo y uno menor que cero indica un sentimiento negativo o de rechazo.

4.4.5.2. Sinónimo con sentimiento

El archivo que contiene las reglas tiene en cada línea, separados por punto y coma: La palabra a buscar, el candidato con el que está asociado y el sentimiento correspondiente.

Algunos ejemplos de reglas definidas en el archivo:

```
#HashTag;Subject;sentiment
YoMarco4;Marco Enríquez-Ominami;1
SiTuQuieresChileCambia;Marco Enríquez-Ominami;1
Vota1Parisi;Franco Parisi;1
JovenesXBachelet;Michelle Bachelet;1
NOaBachelet;Michelle Bachelet;-1
NoAMatthei;Evelyn Matthei;-1
nomasderecha;Evelyn Matthei;-1
ChaoEvelyn;Evelyn Matthei;-1
ChaoBachelet;Michelle Bachelet;-1
```

Esta regla es la más simple. Se trata de simplemente recorrer todo el árbol y comparar cada palabra con cada una de las palabras de las distintas reglas. En caso de encontrar una de estas palabras, se retorna un nuevo objeto de resultado que contiene el nombre del candidato y el valor del sentimiento.

En una posterior optimización, esta regla se comenzó a aplicar al texto del Tweet y no al árbol. Así se evitó el lento proceso de generación de este árbol, mejorando la velocidad de procesamiento.

4.4.5.3. Patrón Lineal

El archivo de configuración en cada línea incluye dos palabras separadas por ';', siendo una de ellas "@Subject" para indicar cualquier forma de llamar a un candidato, un "True" o "False" indicando si las palabras tienen que ir inmediatamente una después de la otra, o pueden haber otras palabras entre medio, un "True" o "False" indicando si las palabras se pueden cambiar de orden y finalmente, el sentimiento asociado a la regla.

Ejemplos de reglas:

```
#First;Second;inmediate(boolean);invertible(boolean);Sentiment
@Subject;grande;True;True;1
votar;@Subject;False;False;1
@Subject;mentir;False;True;-1
@Subject;presidente;False;False;1
apoyar;@Subject;False;True;1
```

Para cada regla, el algoritmo busca en el árbol de dependencia la primera palabra y la segunda, en caso de que la regla sea invertible. A continuación revisa si esta palabra es ancestro de la segunda palabra clave (o primera en caso de que la regla sea invertible). Esta revisión se hace en el hijo inmediato, o descendiendo recursivamente en caso de que i*nmediat*e sea *False*.

En caso de que en el camino recorrido en el árbol se haya encontrado la palabra "no", el valor de sentimiento obtenido se multiplica por -1, invirtiendo su valor de apoyo a rechazo y de rechazo a apoyo.

4.4.5.4. Patrón Verbo-Unión

Esta regla se describe con el verbo, luego @Subject, la palabra clave y el sentimiento asociado, tal como se muestra en el siguiente ejemplo:

```
#Verb;@Subject;Word;Sentiment
sumar;@Subject;me;1
ser;@Subject;mejor;1
ser;@Subject;peor;-1
ser;@Subject;mentiroso;-1
```

Para cada verbo, el algoritmo recorre el árbol de dependencia hasta encontrar el verbo o hasta llegar a una hoja. A continuación busca entre los descendientes de este nodo a alguno de los sujetos y la palabra definida en la regla. En caso de encontrar estas palabras, se retorna el objeto de la clase *SentimentReport* con el sujeto encontrado y el sentimiento detectado. Al igual que en el caso de la regla de Patrón Lineal, en caso de encotrar una negación, se invierte el valor del sentimiento detectado.

4.5. Identificador de preferencia de voto

Este módulo obtiene el candidato favorito de cada uno de los usuarios que emitieron alguna opinión en un rango de fechas para, finalmente, calcular la cantidad de personas que apoyan a cada candidato.

El algoritmo es bastante simple y se describe a continuación:

- Se obtienen todos los usuarios que escribieron algo en el rango de fechas y se detectó un sentimiento en el texto.
- Se itera obteniendo todos los Tweets con sentimiento para el usuario en el rango de fechas.
- Se cuentan las cantidades de opiniones positivas sobre cada candidato y se selecciona aquel que haya tenido la mayor cantidad de opiniones positivas. Este es el candidato favorito del usuario de Twitter.
- Se suma 1 a la cuenta de personas que apoyan al candidato resultante.

4.6. Localización geográfica de los Tweets

Es de gran interés saber desde dónde se está emitiendo una opinión. Con esta información se puede medir la efectividad de campañas publicitarias sectorizadas, medir la popularidad de un candidato, producto, o marca en las distintas áreas, etc.

4.6.1. Tweets con GeoTag

Twitter permite a los usuarios incluir su posición por GPS junto con los mensajes. Esta posición, de estar disponible, viene en formato de latitud y longitud.

Si bien ésta es la forma más precisa y simple de saber cuál fue la ubicación del Tweet, la cantidad de usuarios que tienen activada la opción de compartir su ubicación es muy baja. Entre todos los Tweets obtenidos durante los casi 4 meses, que se recolectaron datos para este trabajo, solamente el 2 % de los Tweets incluyeron esta información. Esta es una fracción muy baja, por lo que se desea tener una forma alternativa de obtener la localidad de los mensajes.

4.6.2. Localidad en perfil de usuarios

La API de Twitter permite obtener desde el perfil de un usuario la localidad que él indica. Se trata de un campo de texto libre que algunos usan para indicar la información de su ubicación y otros como un campo de texto libre, en el que se escribe cualquier palabra, mensaje, símbolo, o una simple combinación de caracteres. Los usuarios que no hacen mal uso de este campo de texto, indican su ubicación con distinto nivel de precisión: algunos indican el país, otros la región, o la comuna, o la ciudad. Algunos incluso indican su dirección exacta con nombre de calle y número de casa.

Es importante notar que esta ubicación es del usuario de Twitter, y no necesariamente del lugar desde el que fue emitido, pero hay fuertes razones para suponer que si alguien pone que es de una ciudad, pero no está en ella, al menos tiene un interés en esa ciudad.

Se estableció como objetivo para esta parte saber si es posible usar esta información para separar los Tweets según la región de Chile a la que pertenecen sus autores. Para lograrlo, se almacenó esta información del perfil en la base de datos para su posterior análisis.

4.6.3. Identificación de la localidad del usuario

Se decidió hacer la separación de los Tweets según la región de Chile a la que pertenecen sus autores. Para lograr esta separación, se creó un sistema de reglas basado en *expresiones regulares*. En este sistema, cada región tiene un conjunto de expresiones regulares, que en caso de calzar con la ubicación declarada por el usuario, determinan la pertenencia de ese usuario a esa región.

Además de las 15 regiones de Chile, se agregó el área Chile y Todo el mundo. Las regiones son principalmente representadas por los nombres de las comunas que las componen y algunas ciudades más grandes ubicadas en ellas.

Algunos ejemplos de reglas definidas:

Área	Expresión regular
Todo el mundo	.*
Chile	.*[CcSs]hile.*
Chile	.*CL.*
Chile	.*[Cc]hilito.*
Chile	.*CHILE.*
XV	.*[Aa]rica.*
XV	.*[Pp]arinacota.*
XV	.*[Pp]utre.*
XV	.* XV .*
I	.*[Tt]arapac[aá].*
I	.*[Ii]quique.*
I	.*[Aa]lto [Hh]ospicio.*
RM	.*[Ss]antiago.*
RM	.*[Vv]itacura.*
RM	.*[Ss]tgo.*

4.6.4. Credibilidad del dato obtenido

Para validar la credibilidad, o no credibilidad de la localidad obtenida a partir del dato en el perfil del usuario, se implementó un sistema que ayuda a comparar este dato con los datos del GPS, en caso de existir.

La complejidad de hacer la comparación es averiguar a que localidad pertenecen las coordenadas GPS. Para obtener este dato, se usó una funcionalidad que ofrece Google a través de su API llamada GeoCode [27]. El uso de esta API es muy simple, ya que consiste en una llamada GET a la `http://maps.googleapis.com/maps/api/geocode/xml?sensor=true&latlng=LAT,LON`, donde LAT y LON son los valores numéricos de la latitud y longitud respectivamente.

La respuesta de los servidores de Google es un XML con toda la información geopolítica de dicha coordenada. Se incluye el nombre del país, región, comuna, ciudad e incluso la dirección en caso de existir.

Esta API tiene un límite de un máximo de 2,500 consultas en un periodo de 24 horas. Al momento de la implementación de esta funcionalidad ya se tenía un gran volumen de datos almacenados por lo que para completarla con los datos del GeoCode, se implementó un proceso demonio que se ejecuta cada 15 minutos y realiza 17 consultas, logrando un total de 2448 consultas diarias que quedan dentro del límite permitido. La información obtenida se almacena en la base de datos para poder ser usada libremente por otros módulos de la

aplicación.

4.7. Visualización de datos obtenidos

Se implementaron distintas visualizaciones de los datos obtenidos y generados con el funcionamiento de la aplicación.

Las visualizaciones se implementaron como páginas web generadas dinámicamente usando JSF. Los gráficos se dibujan mediante el uso de la librería Google Charts [28] y el estilo gráfico de algunas páginas fue implementado usando el popular CSS Bootstrap [29].

Para hacer los gráficos en Google Charts, basta definir un div vacío en la página. Un javascript ubicado en el encabezado de la página web es el encargado de dibujar el gráfico en dicho div. A continuación, un ejemplo de este jacvascript:

```
<script type="text/javascript">
  google.load("visualization", "1", {packages:["corechart"]});
  google.setOnLoadCallback(drawChart);
  function drawChart() {
    var data = google.visualization.
                          arrayToDataTable(${viewCachedStats.chartData});
  var options = {
    title: 'Tweets'
  };
  var chart = new google.visualization.
                          LineChart(document.getElementById('chart_div'));
  chart.draw(data, options);
  }
</script>
```

La definición de los arreglos inicializados con datos en javascript es igual a la manera en la que *EL* de de JSP imprime estas estructuras hacia la página web. Es por esto, que basta con imprimir el arreglo usando $\{viewCachedStats.chartData\}$. En caso de las estadísticas precalculadas, el *String* con datos que se genera debe seguir el mismo formato, es decir, una matriz es un arreglo de areglos. Cada arreglo se encierra en corchetes '[' y ']' y sus elementos se separan por una coma ','.

4.7.1. Precálculo de resultados

La gran cantidad de datos que maneja la aplicación hace que la preparación de estos para la visualización tome mucho tiempo. Por ejemplo, la generación de un gráfico que muestra la cantidad de menciones por candidato versus el tiempo para un día (al rededor de 20,000 Tweets) se demora al rededor de 3 minutos. Este tiempo es lineal en función de la cantidad de Tweets totales del periodo a visualizar y dista de ser interactivo, por lo que se decidió precalcular los datos periódicamente y almacenarlos para poder desplegarlos rápidamente.

Se crearon tareas programadas en el servidor para ejecutarse durante la noche. Cada tarea debe calcular una estadística diferente, al terminar guarda el resultado en una tabla especialmente creada en la base de datos para estos fines. El resultado guardado se entrega directamente al javascript de la página web que dibuja el gráfico lo que hace que la respuesta de las páginas web sea casi inmediata.

4.7.2. Gráficos generados

4.7.2.1. Histograma de menciones por candidato en un período

Los datos para este gráfico se generan siguiendo el algoritmo que se detalla a continuación. Se desean ver los datos entre los momentos T_0 y T_f con un delta de ΔT entre punto y punto.

- Establecer el valor de la variable T_1 en T_0 y de T_2 en $T_0 + \Delta T$
- Crear un arreglo de mapas *Data*. Cada elemento del arreglo representará un intervalo de tiempos, cada mapa tendrá como llave el nombre de un candidato y como valor asociado la cantidad de menciones para ese candidato en ese intervalo de tiempo. Se inicializa en 0.
- Crear un arreglo de Strings *Intervalos* dónde se guardarán los intervalos de tiempo. Estos serán las etiquetas del eje X del gráfico.
- Iterar mientras $T_2 < T_f$
  - Seleccionar y obtener desde la base de datos todos los Tweets emitidos entre los momentos T_1 y T_2
  - Para cada Tweet *tw* identificar los candidatos que son mencionados en él.
  - Incrementar la cuenta de menciones para el candidato para el intervalo de tiempo actual en el arreglo de mapas *Data*.
  - Agregar a *Intervalos* "$(T_1 + T_2)/2$" en formato de texto.
  - $T_1 = T_1 + \Delta T$ y $T_2 = T_2 + \Delta T$
- Transformar los datos generados en un *String* al formato exlicado en la sección 4.7.

En caso de ser necesario, se puede agregar además la cantidad de Tweets totales en cada intervalo de tiempo.

En lafigura 1 se puede ver un ejemplo de gráfico generado.

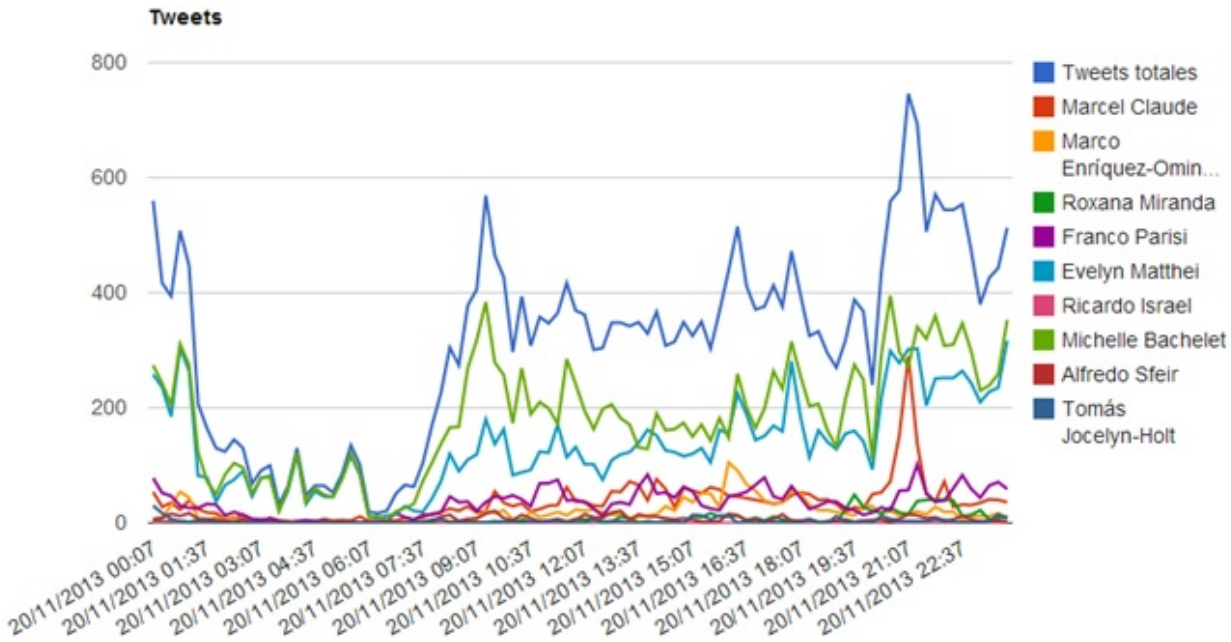

Figura 1: Gráfico que muestra la cantidad de menciones versus el tiempo.

4.7.2.2. Menciones de candidatos por región

Este tipo de gráficos obtienen Tweets para un período de tiempo y a continuación detectan la localidad de cada uno de ellos. Con estos datos se construyen y dibujan gráficos de torta para cada región. Los datos de cada uno de los gráficos muestran la cantidad de Tweets que mencionan a cada uno de los candidatos, como se puede ver en la figura 2 para la Región de Antofagasta, el 23 de noviembre del 2013.

4.7.2.3. Apoyo y rechazo de candidatos por período

Este gráfico sólo se implementó para la segunda vuelta de las elecciones, debido a que el buscador de sentimientos se pudo implementar y hacer funcionar a fines de noviembre.

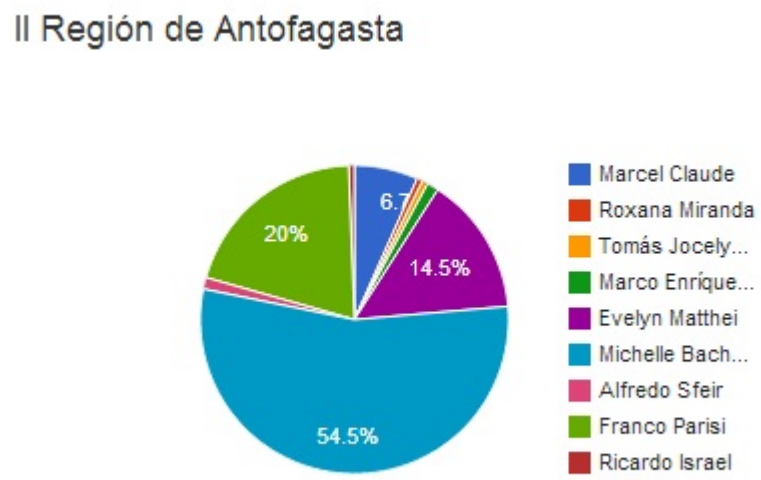

Figura 2: Gráfico que muestra la cantidad de menciones de los candidatos el 23 de Noviembre de 2013 en la Región de Antofagasta.

Este tipo de gráfico se basa en el resultado del módulo que busca sentimientos en los Tweets. Consiste en obtener los datos de un período (por ejemplo, 1 día) y contar la cantidad de mensajes en apoyo, y rechazo para cada candidato. Un ejemplo de dicho gráfico se puede ver en la figura 3.

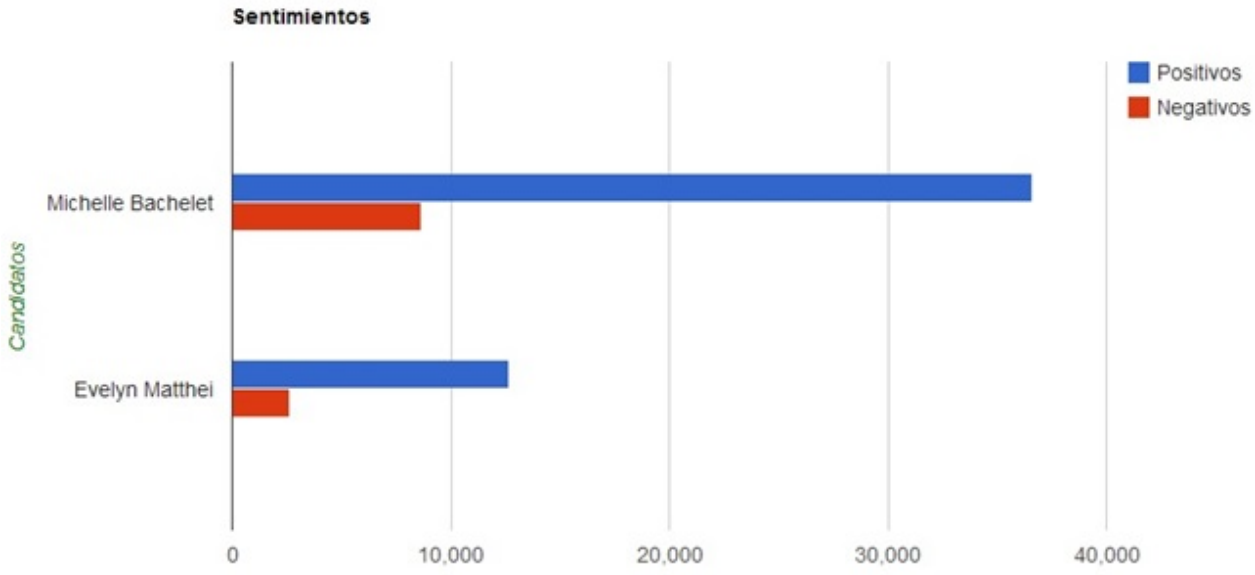

Figura 3: Gráfico que muestra la cantidad de menciones de apoyo/rechazo de las candidatas de la segunda vuelta entre el 9 y el 15 de diciembre del 2013.

4.7.2.4. Preferencia de voto

Por razones de tiempo no se alcanzó a implementar este gráfico para ser visible por el público antes de las elecciones, sin embargo si se implementó de una manera experimental. Se sigue el procedimiento explicado en 4.5 y se guardan los datos en un arreglo de arreglos conteniendo los nombres de los candidatos en la primera posición de cada arreglo interno y la cantidad de personas que prefieren a cada uno en la segunda posición. En la figura 12, de la sección 5.4 se puede ver un ejemplo de dicho gráfico.

4.7.3. Implementación de la publicación de resultados para el usuario

La herramienta explicada en este trabajo se ejecutó en un servidor pequeño y con una conexión a Internet hogareña. Para evitar sobrecargas e interrupción del servicio debido al acceso concurrente a la página web de unos pocos usuarios, se decidió implementar un sistema de Caché de páginas que hospedaría la web en un servidor más potente al que pudieran acceder varios usuarios al mismo tiempo.

Para implementar esta solución se decidió usar una característica de Google Drive que permite hospedar páginas web estáticas [30]. Este sistema permite almacenar en una carpeta de Google Drive páginas html, imágenes y otros recursos y abrirlos usando la URL `https://googledrive.com/host/CÓDIGO_CARPETA_COMPARTIDA/ARCHIVO`, en esta dirección *CÓDIGO_CARPETA_COMPARTIDA* es un código que se obtiene al compartir una carpeta de google drive (haciéndola pública) y *ARCHIVO* es el nombre de un archivo ubicado en esa carpeta.

En la figura 4 se puede ver la carpeta que se tiene para hospedar las páginas para el público.

Para no tener que actualizar las páginas manualmente varias veces por día, se implementó un sistema que actualiza las páginas automáticamente una vez por hora utilizando Google Script [31]. Este sistema permite crear programas en javascript, que se ejecutan en los servidores de Google. Además, se puede programar la ejecución automática y periódica de estos programas.

El programa desarrollado se conecta a la URL desde la que es visible el servidor principal de Twittermeter y descarga las páginas necesarias para guardarlas en archivos temporarios. Una vez descargadas las páginas, se reemplazan los archivos htm definitivos por los archivos nuevos. En caso de fallar la descarga de las páginas, el código no toca los archivos htm existentes anteriormente, por lo que si bien la página no se refresca con los datos nuevos, no deja de existir y los usuarios no verán un error 404.

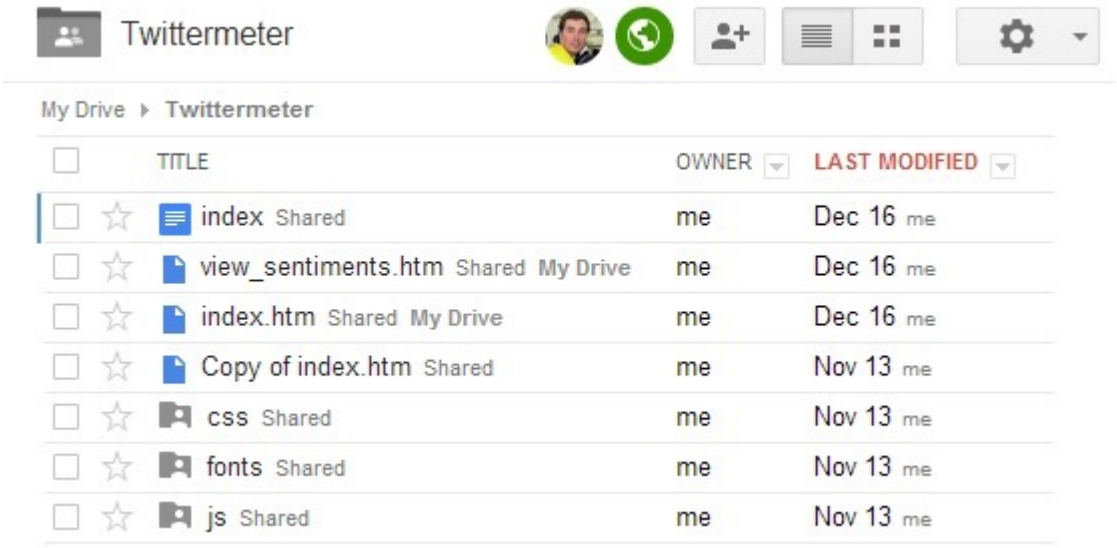

Figura 4: Directorio de Google Drive con los archivos para el sitio web público.

Este sistema se aplicó para dos páginas de estadísticas: la de cantidad de menciones por región y la de opiniones de apoyo y rechazo por candidato.

El código de este script se puede ver en el Apéndice B: Script que copia páginas del servidor a Google Drive.

La URL en Google Drive donde quedó disponible la página es `https://googledrive.com/host/0B7WZb1MN64OvdWdoXzJwblRwYWs/`. Para poder compartirla con más facilidad, se creó una pequeña página en servidor al que se tenía acceso, que simplemente redirigía a esta url. La url que se divulgó es `www.solidit.cl/twittermeter`.

La figura 5 muestra una foto de la página web disponible para el público.

Se instaló un script para contar la cantidad de visitas a esta página y se divulgó a través de redes sociales. Se hizo una publicación antes de ambas vueltas electorales llegando a tener alrededor de 400 visitas a las páginas.

4.7.3.1. Comparación de localidad por perfil y por GPS

Para estudiar la confiabilidad del dato de localidad obtenido mediante las expresiones regulares, se implementó una visualización que muestra en un mapa puntos de todos los Tweets que tienen información de GPS y pertenecen a una región parametrizable. En este mapa se colorean de color verde aquellos puntos que coinciden en el dato del GPS y en el dato obtenido por expresiones regulares y se colorean en rojo aquellas para las que el dato no coincide. La página permite indicar la región para la que se desean obtener los datos (este dato es el obtenido mediante expresiones regulares) y el texto a buscar en el XML de

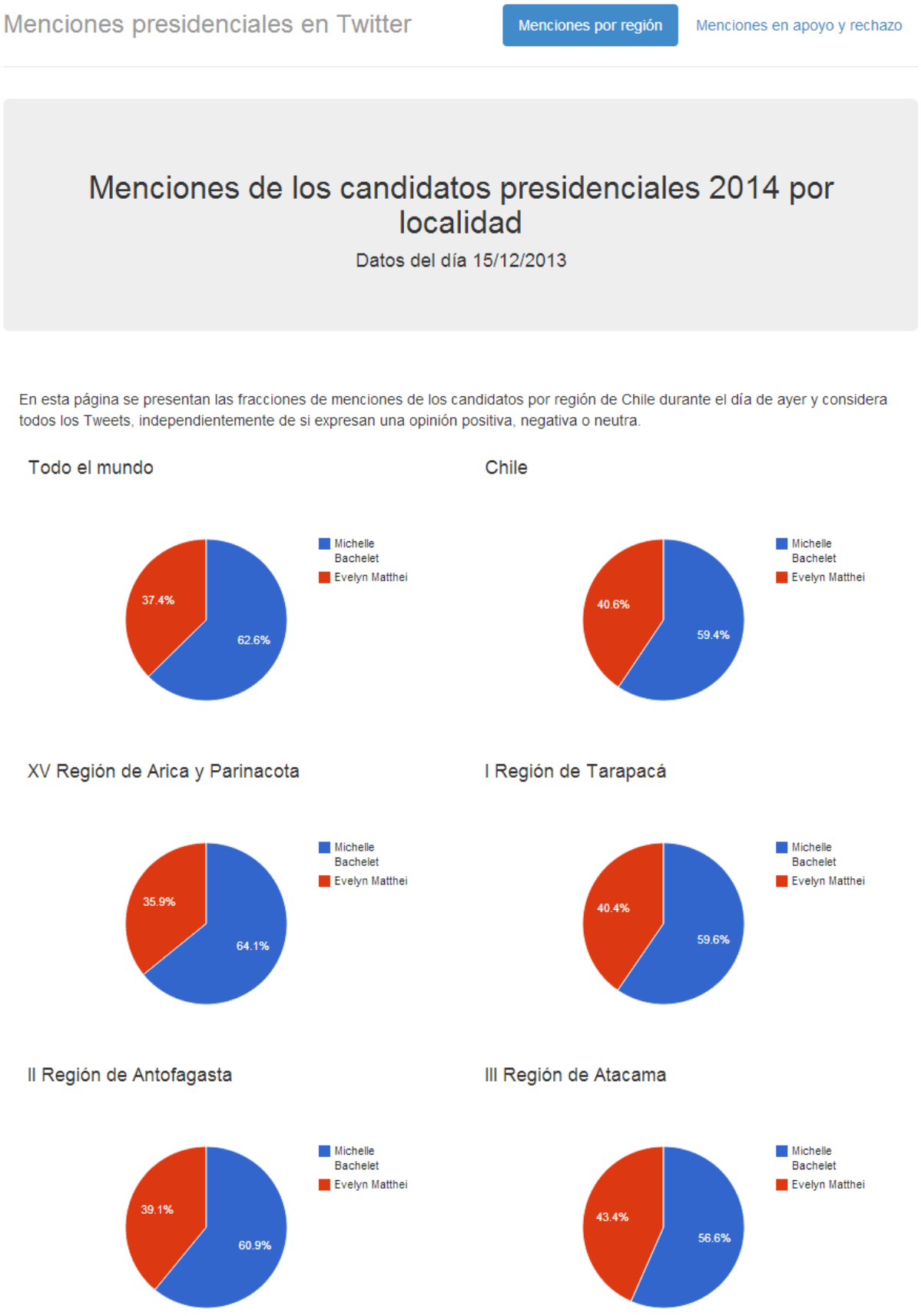

Figura 5: Foto de parte de la página web pública del sistema desarrollado.

GeoCode, obtenido desde Google a partir de la información del GPS. Esta visualización usa la API de Google Maps [32]. La figura 6 muestra un ejemplo de la página que permite hacer el análisis.

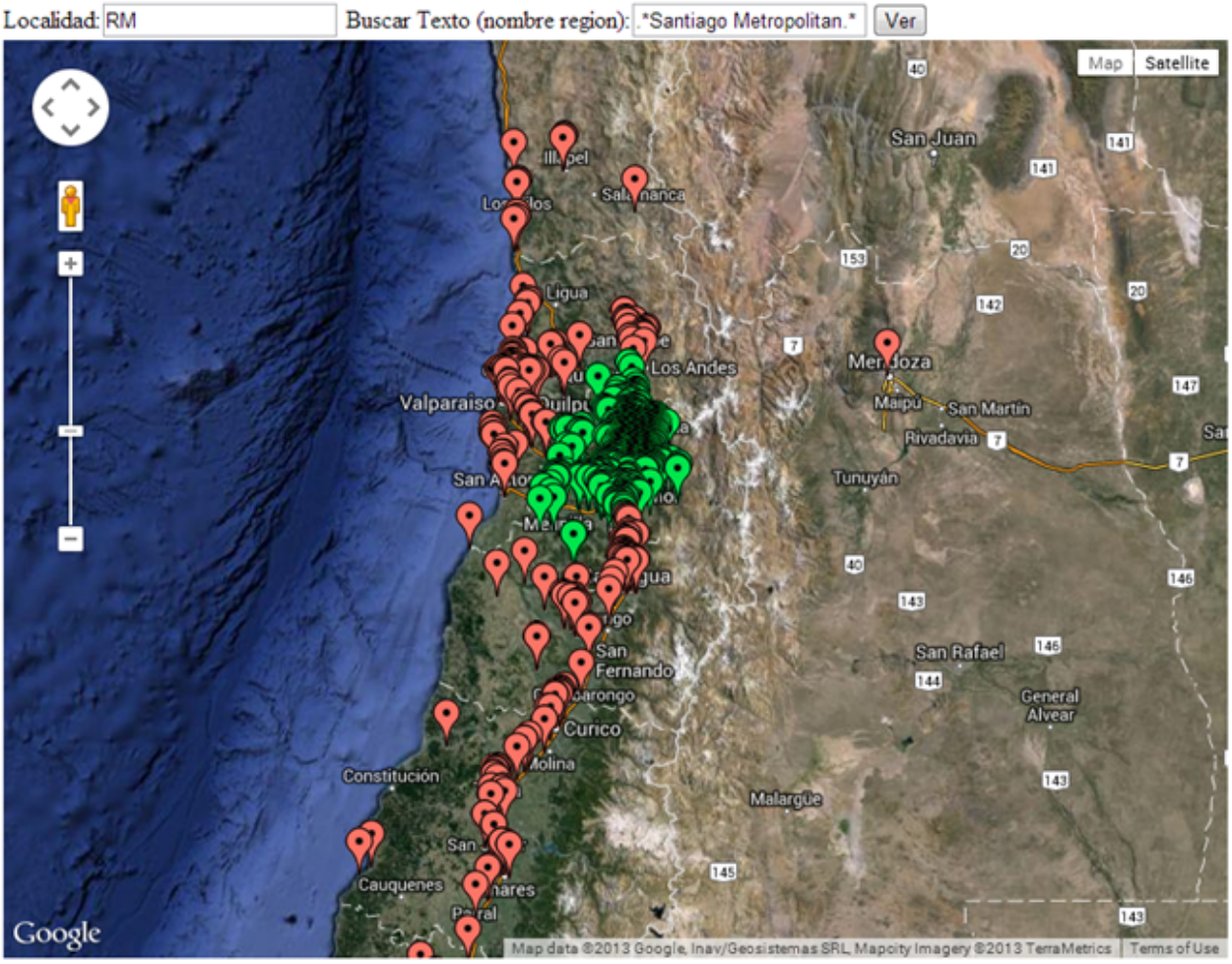

Figura 6: Herramienta para comparar localidad obtenida mediante expresiones regulares de información del perfil y mediante GPS. Se muestran los datos para la Región Metropolitana.

4.8. Diseño de la aplicación construida

4.8.1. Arquitectura

Como ya se explicó anteriormente, la aplicación esta programada en Java y consiste en dos partes. Una de ellas se ejecuta sobre un servidor JBoss y la otra es una aplicación Java Estándar. La segunda aplicación es la encargada de ejecutar la ontología para obtener el apoyo o rechazo hacia el sujeto mencionado en el mensaje del Tweet.

La comunicación entre estas aplicaciones se hace a través de la base de datos. La aplicación sobre JBoss obtiene los Tweets y los deja en la base de datos, además de generar estadísticas y las visualizaciones de los gráficos. La aplicación estándar toma los datos de los Tweets desde la base de datos, los analiza y guarda el resultado obtenido en la misma base de datos.

El esquema de la arquitectura y el flujo de datos se puede ver en la figura 7

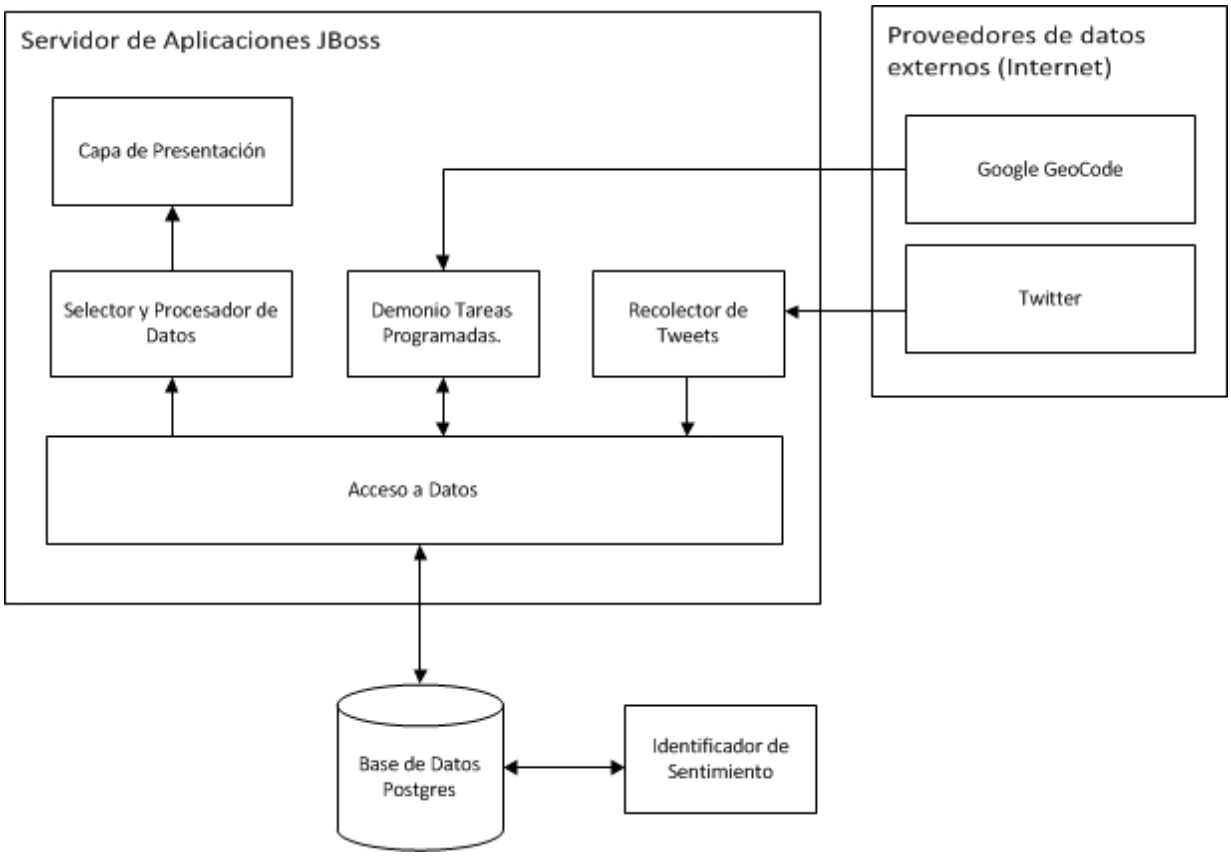

Figura 7: Arquitectura final de la aplicación. Flujo de datos.

4.8.1.1. Escalabilidad

Considerando el gran volúmen de datos que genera Twitter, es necesario tener presente la escalabilidad de la aplicación desarrollada. La componente que más poder de procesamiento requiere es el módulo de Identificación de sentimiento. Dado el diseño de componentes mostrado, es fácil ver que es posible escalar horizontalmente, ejecutando, el módulo en cuestión en otras maquinas en forma paralela y haciendo pequeños ajustes para que los datos a procesar por cada instancia del módulo no se intersecten entre ellos.

De ser necesario, se pueden ejecutar instancias adicionales del JBoss con la aplicación instalada, que se encarguen de una parte del stream del twitter. Para esto, sería necesario configurar cada instancia con un set de palabras claves a suscribirse distinto. Además sería necesario realizar un cambio en la implementación para comprobar que un Tweet (que tiene

un identificador único) no haya sido guardado ya en la base de datos por otra de las instancias antes de realizar su persistencia.

4.8.2. Modelo de datos

Se eligió la base de datos PostgreSQL para la capa de persistencia.

La base de datos contiene todos los datos obtenidos y generados por la aplicación. Sus tablas y atributos más importantes se muestran en el diagrama 8.

Las tablas que componen el sistema son:

- **jobs**: Permite al programa funcionar con más de un tema al mismo tiempo (por ejemplo: candidatos y canales de TV), asociando cada uno de los sujetos y Tweets con el trabajo (job).
- **configuration**: Contiene datos necesarios para la conexión con Twitter.
- **Tweets**: Guarda los Tweets recolectados por la aplicación.
- **users**: Guarda los usuarios de twitter.
- **subject**: Guarda una lista de los sujetos de interes de la aplicación. En este caso, los nombres de los candidatos presidenciales.
- **term**: Contiene los términos suscritos, no suscritos y de lista negra para el filtro de Tweets relevantes. Cada entrada está relacionada con el subject.
- **location_groups**: Contiene la lista de las regiones junto con un id.
- **location_regexp**: Contiene la lista de las expresiones regulares y el id del grupo (región) con la que se asocia.
- **sentiment**: Relaciona un Tweet y un Subject y contiene el valor del sentimiento encontrado.
- **tweet_sentiment**: Guarda una representación textual del árbol de dependencia. Sirve para propósitos de depuración.
- **statistics_cache**: Aquí se guardan los datos para los gráficos precalculados.

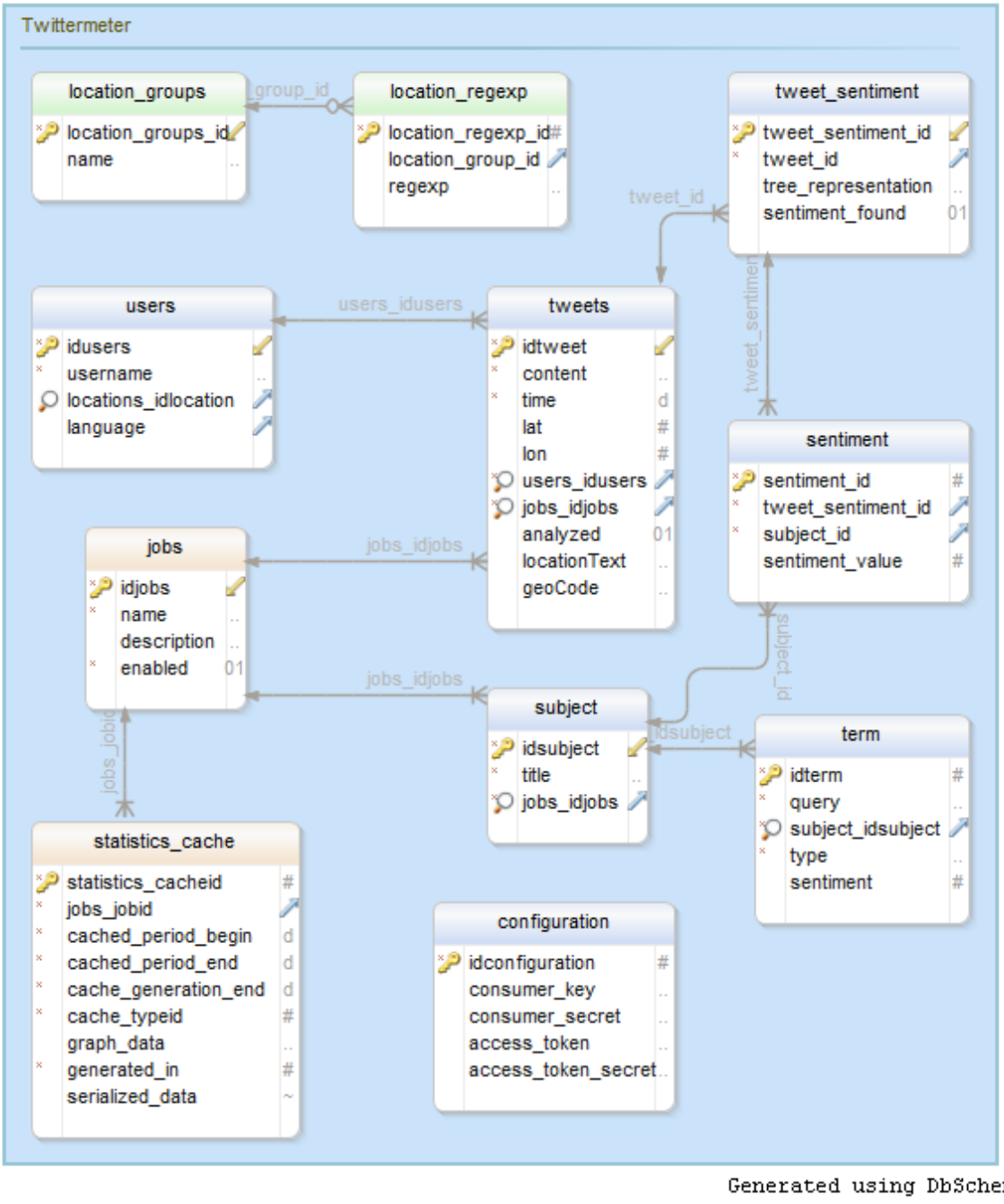

Figura 8: Modelo Relacional de la base de datos.

Capítulo 5

Análisis e interpretación de resultados obtenidos

5.1. Menciones de candidatos

Lo primero que se validó es la correctitud del funcionamiento del recolector de Tweets.

5.1.1. Validación de relevancia de Tweets

Se realizó una revisión manual de los Tweets recolectados desde el Stream de Twitter. Se revisó una muestra aleatoria de 1000 tweets recolectados por el sistema, comprobando que 967 mensajes efectivamente se refieren a los candidatos presidenciales, lo que representa al rededor de un 97 % de acierto. Los Tweets restantes cumplen con los criterios establecidos para la detección de relevancia, sin embargo, al comprender el mensaje, se observa que son referentes a temas de otros países u otras personas. Este es un error aceptable, ya que al existir una gran cantidad de Tweets, los mensajes erróneos se hacen estadísticamente irrelevantes.

5.1.2. Actividad diaria en Twitter

Un gráfico típico de actividad (menciones de candidatos) es el que se puede ver en la figura 9.

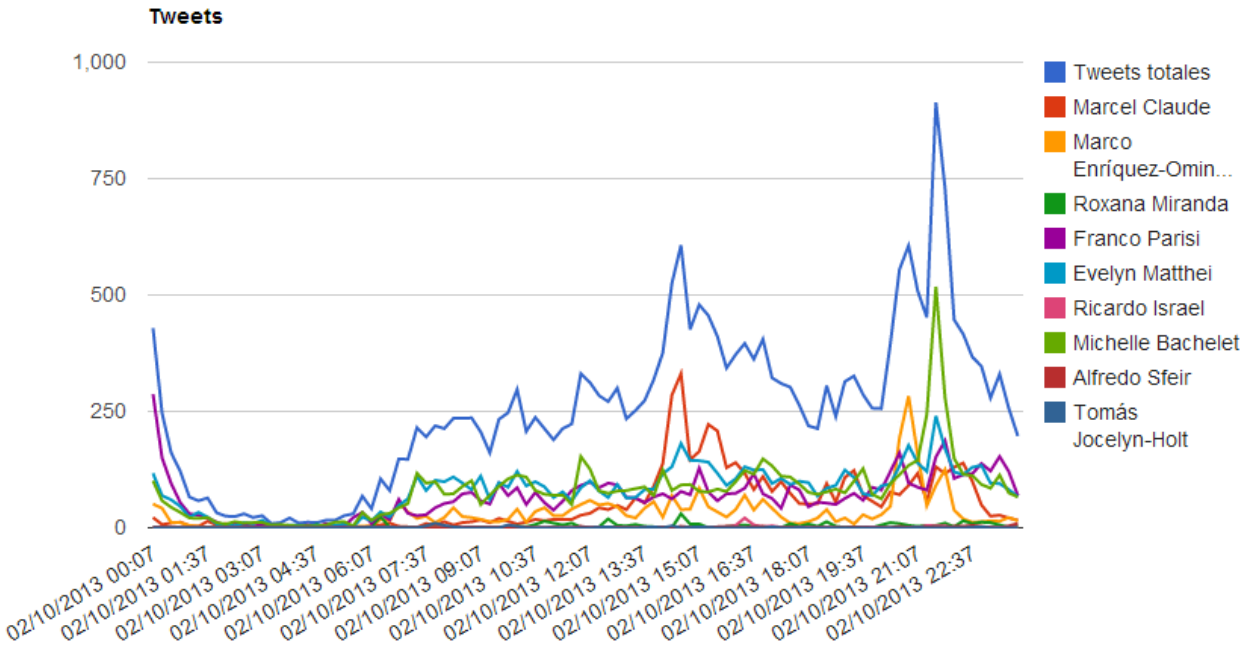

Figura 9: Actividad en Twitter sobre los candidatos el 2 de octubre del 2013.

Un comportamiento similar se observó durante la mayoría de los días.

En el se observa que la actividad decae en la madrugada, llegando a ser casi nula. Se puede observar un aumento de la actividad en la mañana, cuando las personas despiertan y van a sus trabajos y lugares de estudio. Durante la mañana, la actividad se mantiene casi constante hasta pasada la hora del almuerzo. En ese momento se observa un aumento en la actividad. En el horario típico de salida del trabajo la actividad disminuye para aumentar nuevamente durante los noticieros nocturnos y otros programas de debate.

Este comportamiento se puede relacionar con las actividades que realizan las personas durante el día. Duermen en la noche. Se conectan a Twitter cuando viajan hacia los trabajos. Las personas también utilizan parte de su tiempo de trabajo o estudios en. A la hora de almuerzo disminuye la actividad, y justo después del almuerzo, antes de trabajar, aumenta. Se observa una importante caída en la actividad en el horario de salida, lo que probablemente se debe a que las personas realizan otras actividades como comprar y hacer vida social. Al llegar a las casas y ver las noticias, los usuarios de Twitter comienzan a discutir lo ocurrido durante el día. Finalmente al acostarse a dormir la actividad vuelve a bajar hasta ser, prácticamente, nula.

Esta información se puede aprovechar para saber en que momento es más efectivo escribir mensajes con propósitos de modificar la opinión pública para que estos sean vistos por la mayor cantidad de usuarios. Este momento dependerá del tipo de mensaje que se desa publicar, en función de la actividad en Twitter y del horario, que defíne qué están haciendo las personas.

5.2. Segregación geográfica de los datos

Se analizó la correspondencia entre la localidad obtenida para un usuario a partir de la información en su perfil y la de su GPS. Se pueden tener dos enfoques al respecto. El primero busca saber qué porcentaje de los que dicen estar en una región, efectivamente están ubicados en ella. El segundo enfoque busca saber qué porcentaje de los que efectivamente están en una región dicen estar en ella, o su complemento, la cantidad de usuarios que, estando en una región, no dicen estar en ella.

La tabla del Apéndice C muestra la comparación entre los datos obtenidos desde el perfil de usuario y por GPS. Los mismos datos se pueden observar en la figura 10.

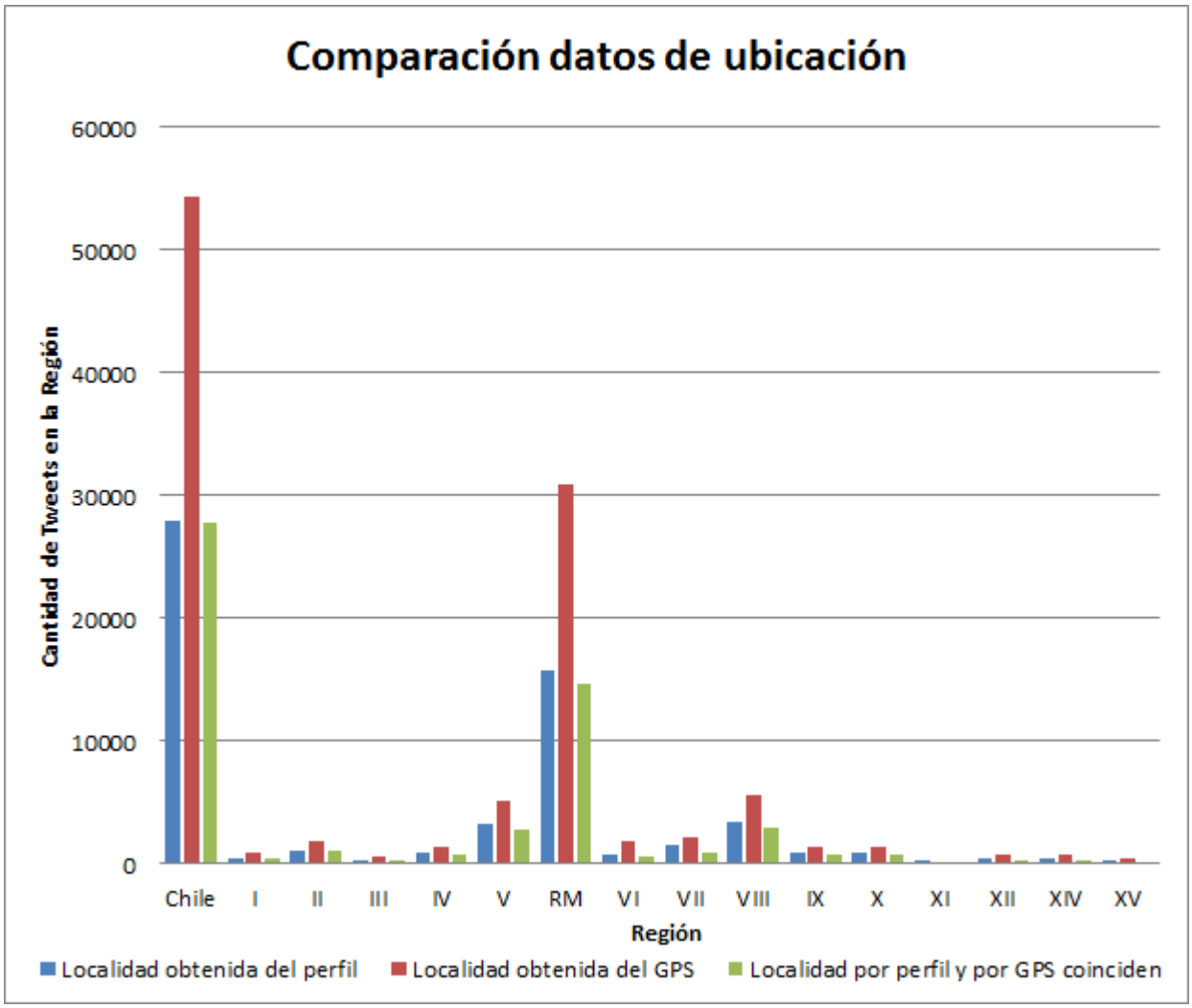

Figura 10: Gráfico de comparación de la ubicación de los usuarios obtenida desde el perfil y usando GPS.

Se puede apreciar que si se logra identificar una ubicación usando la información de localidad del perfil del usuario, existe un alto porcentaje, mayor al 90 % para la mayoría de las regiones, de obtener el dato correcto. Por otro lado, queda una cantidad cercana a la

mitad de todos los usuarios para los que no se puede obtener su localidad correctamente usando la técnica propuesta y probada. Esto significa que el dato se puede usar para obtener estadísticas de un gran volumen de datos geolocalizándolos, similarmente a lo que se realizó en este estudio, pero no sirve para, por ejemplo, obtener todos los Tweets de una localidad, pues quedará al rededor de la mitad de los mensajes fuera del conjunto obtenido.

Los datos obtenidos para este estudio consistieron en un universo de 56,946 Tweets con información de GPS recolectados entre el 15 de agosto y el 18 de diciembre del 2013.

La figura 11 muestra todos los Tweets de los que se tiene información GPS en el sistema, con los datos de Chile destacados en color verde.

Figura 11: Todos los Tweets con GPS recolectados.

No se han encontrado otras implementaciones de un sistema similar hasta el momento, por lo que esta característica es diferenciadora de otras empresas e instituciones que realizan estudios en las redes sociales.

5.3. Mensajes en apoyo y rechazo

Se hizo una revisión manual del acierto del identificador de sentimiento al procesar los Tweets. Se revisaron manualmente 3000 mensajes seleccionados aleatoriamente relacionados con la segunda vuelta de las elecciones, identificando si tenían una opinión, sobre qué candidata era esa opinión y si era positiva o negativa, y se comparó con lo identificado por la

aplicación desarrollada.

- Se identificó que un total de 1318 Tweets tenían una opinión marcada sobre alguna de las candidatas.
- El algoritmo detectó 630 opiniones.
- 385 de las opiniónes detectadas automáticamente coincidieron con la opinión realmente emitida.
- 97 opiniones de las que no coincidieron detectaron una opinión contraria a la real, es decir, se identificó una opinión positiva siendo negativa, o al revés.
- Las 148 opiniones restantes fueron identificadas como de preferencia o rechazo hacia una de las candidatas, sin embargo, eran mensajes neutros.

De esta manera, en caso de que el algoritmo haya detectado una opinión, existe un 61 % de probabilidades de que esta sea correcta, un 15 % de obtener un sentimiento inverso al real y un 24 % de obtener alguna opinión, siendo que el tweet es neutro.

Este estudio indica un alto porcentaje de falla del sistema automatizado para identificar la opinión. Se identificó también que la mayoría de las opiniones correctamente identificadas estaban marcadas con alguno de los hashatags en apoyo o rechazo hacia las candidatas. Estudiando más a fondo las razones de falla, se detectó que los árboles de dependencia en muchas ocasiones no se construyeron correctamente.

La detección del algoritmo es un poco mejor que una decisión tomada al azar, por lo que para grandes volúmenes de datos, se marca una tendencia estadística correcta. En el caso de la elección existió un gran volumen de Tweets marcados con algún hashtag que permitió identificar inequívocamente el apoyo o rechazo hacia un candidato. Gracias a esto, los resultados finales resultaron parecidos a la realidad.

Para futuros trabajos relacionados con el uso de una ontología para identificar opiniones, se recomienda estudiar otras librerías que permitan hacer procesamiento del texto de lenguje fluido. En septiembre del 2013 apareció una nueva versión de FreeLing [33], que supuestamente mejora varios aspectos de dependencia de otras librerías, uso de memoria y mejora de los soportes de algunos idiomas.

También se sugiere estudiar técnicas adicionales para preprocesar el texto antes de entregárselo a FreeLing.

Es importante mencionar que el módulo de identificación de sentimento, que usa la librería FreeLing, es más lento de lo que se quisiera, pues el tiempo que demora en procesar los datos de un día es mayor a ese día. Se observó que por cada 2 Tweets que se procesan, queda un Tweet cuyo procesamiento se atrasa. Este problema se puede solucionar paralelizando este proceso con otro PC. Los gráficos aqui mostrados incluyen el procesamiento de todos los Tweets del período que abarcan, pues se dejó funcionando el sistema por varios días más para procesar el trabajo postpuesto.

5.4. Usuarios que apoyan y rechazan

Para la segunda vuelta, se calculó una estimación de preferencia de voto por persona, obteniendo el resultado que se muestra en la imágen12. Lamentablemente, por razones de tiempo, esta información no se pudo obtener para la primera vuelta de elecciones, pero tener dos candidatas en vez de 9 simplifica la interpretación y análisis de los resultados.

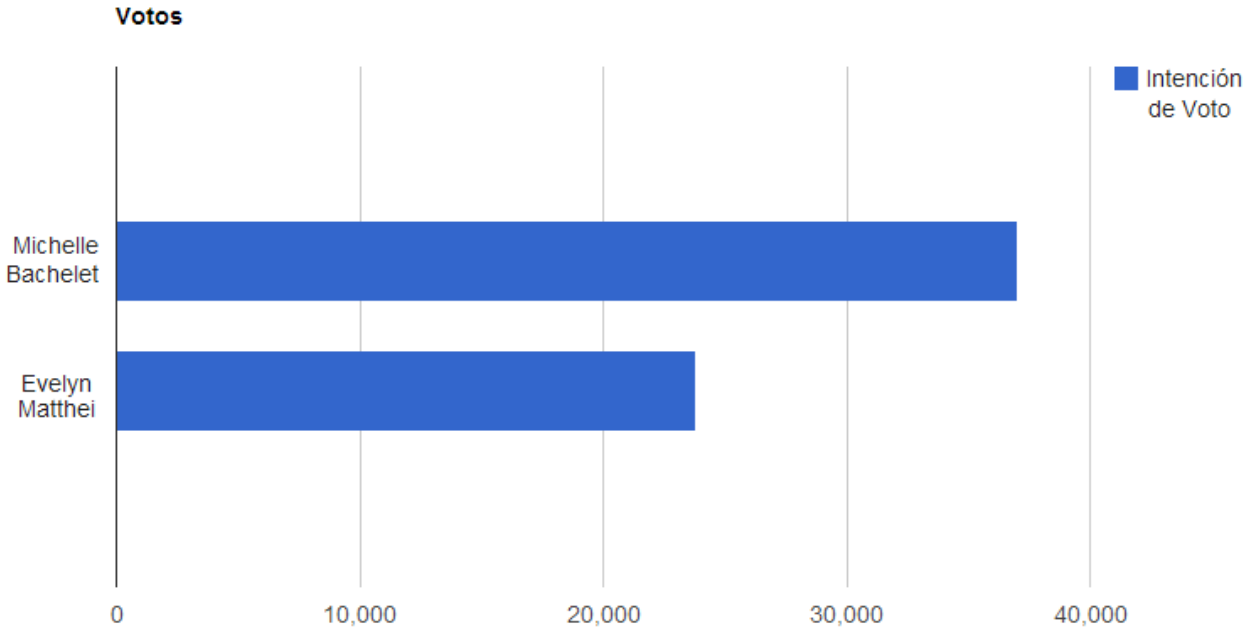

Figura 12: Estimación de intención de voto por las candidatas.

Al comparar el resultado estimado a partir de las opiniones en Twitter con la segunda vuelta de las elecciones [34], se observa un resultado muy parecido:

Candidata	Estimación de intención de voto	Votos obtenidos en segunda vuelta
Michelle Bachelet	60,88 %	62,16 %
Evelyn Matthei	39,12 %	37,83 %

La aplicación desarrollada pudo mostrar la tendencia en la opinión de todos los votantes a partir de los datos de Twitter.

Por falta de tiempo no se alcanzó a implementar una visualización que permitiera ver el apoyo y rechazo de los candidatos separado por región.

5.5. Comparación con Brandmetrics

Durante todo el período de la campaña presidencial, Brandmetrics mostró en su página índices de popularidad de los candidatos en Twitter. Durante todo ese período, se observó una variación de las curvas similar en Twittermeter y en Brandmetrics. Los saltos en la popularidad la mayoría de las veces se debió a alguna polémica que sucedía con los candidatos, observándose además, una similitud entre esta popularidad en Twitter con la popularidad en las encuestas tradicionales.

Brandmetrics también mostró[35] un gráfico con las fracciones de los Tweets que estaban marcados con los hashtags de la forma YoVotoPorXX, donde XX es el nombre de un candidato. Este gráfico se puede ver en la figura 13.

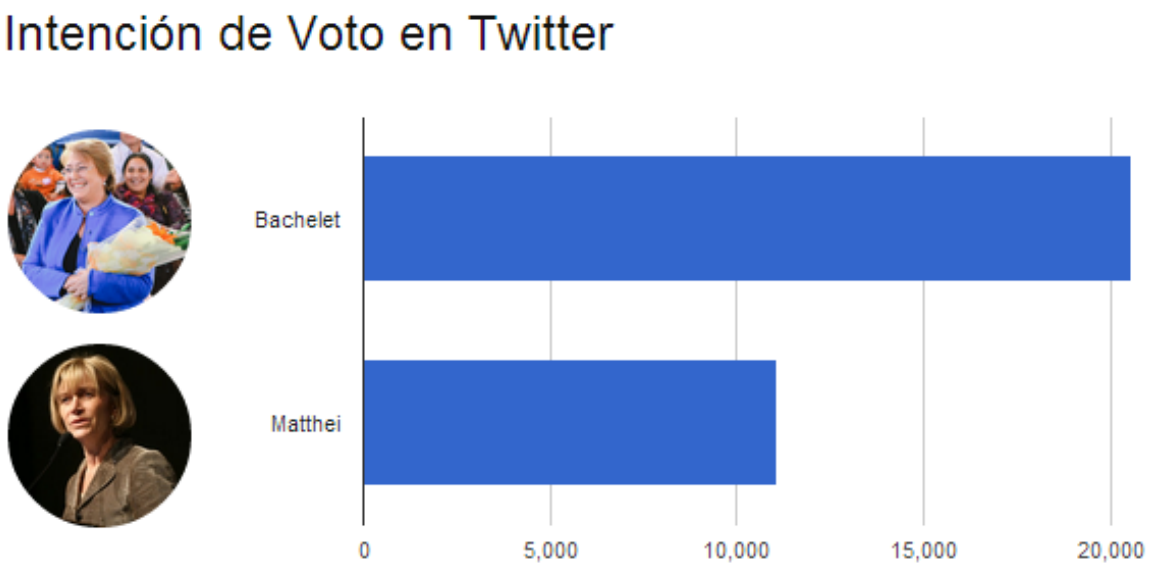

Figura 13: Tendencia observada por Brandmetrics.

Se observa un resultado similar al obtenido en la aplicación desarrollada en esta memoria. Esto se debe a que tanto Brandmetrics cómo Twittermeter usaron hashtags. Sin embargo, Twittermeter además de los hashtags usa las otras técnicas explicadas en la sección Construcción de la Ontología.

El resultado de preferencia según Brandmetrics es: 65 % a favor de la candidata Bachelet y 35 % a favor de la canidata Matthei. Numéricamente estos resultados son también muy parecidos a los resultados de las elecciones.

Brandmetrics también mostró un mapa con las ubicaciones de los Tweets sobre los candidatos, para esto se utilizaron los datos del GPS. Con esto, solamente el 2 % de los Tweets pudo ser puesto en el mapa. El método propuesto y desarrollado en este tabajo permite co-

locar cerca del 50 % de los Tweets en el mapa, lo que permite tener una imágen mucho más completa de las opiniones segregadas por región.

Capítulo 6

Conclusiones

En el desarrollo del trabajo se pudo dar cumplimiento a todos los objetivos descritos en 1.1.

Se estudió el estado del arte de identificar el sentimiento asociado a un trozo de texto y se eligió el método de una ontología, que permitió poner en marcha el sistema sin tener que entrenar algoritmos con millones de Tweets.

Se desarrolló un módulo de la aplicación para recolectar Tweets relevantes para las elecciones y guardarlos en una base de datos. En el período de alrededor de 4 meses se recolectaron más de 4,000,000 de mensajes de Twitter.

Como se explicó anteriormente, surgieron diferentes problemas que se fueron solucionando en mayor o menor medida. El problema más grande es la gran cantidad de errores de la librería FreeLing para entregar los resultados. A pesar de ello, la ontología construida mostró un resultado lo suficientemente correcto como aproximarse al de las elecciones en la segunda vuelta presidencial del año 2013 en Chile.

El método desarrollado para geolocalizar un Tweet a través del dato de ubicación del perfil del usuario, mostró un alto porcentaje de acierto en caso de poder identificar dicha ubicación. Por lo tanto, es una manera confiable para hacer un estudio segregado por región de los datos obtenidos.

El sistema desarrollado para crear un espejo de las páginas con las estadísticas resultó ser eficiente. Desde que se puso en marcha, la información en este espejo se mantuvo actualizada y siempre disponible para el público.

La recolección constante de los mensajes permitió observar la popularidad del uso de Twitter durante las 24 horas, identificando peaks de actividad en la noche, durante los noticieros y bajas de actividad a niveles casi nulos durante la madrugada.

Por razones de tiempo, solamente se alcanzó a implementar el sistema de identificación de preferencia de voto para la segunda vuelta de las elecciones. Los resultados obtenidos resultaron sorprendentemente cercanos a los resultados de la elección y algo diferentes, pero también muy cercanos, al sitio de Brandmetrics, que también realizó un análisis de los Tweets sobre las candidatas. La diferencia del sistema desarrollado de Brandmetrics es la implementación del identificador de la ubicación de los Tweets, que permite conocer la popularidad de los candidatos en cada una de las regiones de Chile. Brandmetrics solamente mostró esta información en base al GPS de los usuarios, que sólo corresponde al 2 % del universo de los Tweets que fueron relevantes. El sistema desarrollado en este trabajo puede geolocalizar cerca del 50 % de los Tweets. Lo que permite tener datos estadísticamente más relevantes.

El uso de Java y JBoss para implementar la aplicación resultó un éxito, con la salvedad de tener que implementar un módulo separado para detectar el sentimiento de los Tweets. La orientación a objetos del sistema y la Java Persistance API permitieron manejar fácilmente los datos almacenados en la base de datos, lo que permitió centrarse en la generación de datos, evitando el uso de tiempo en resolver problemas técnicos.

Cómo un análisis a posteriori, se considera que se podría haber implementado el módulo Identificador de sentimiento en C o C++. Si bien esto implicaría un acceso algo más complejo a la base de datos, se aprovecharía de mejor manera la eficiencia de este lenguaje. Además, FreeLing está implementado en C, por lo que el acceso a esta librería desde ese lenguaje de programación es nativo. Existen más y mejores ejemplos de implementaciones de programas que acceden a FreeLing usando C que cualquiér otro lenguaje, lo que habría acelerado el desarrollo de ese módulo.

Otra mejora posible es usar una base de datos orientada a grafos para la persistencia, para poder navegar fácilmente por el "Grafo social", relacionar distintos Tweets entre ellos, etc. Cosas que de todas formas son posibles de implementar en una base de datos relacional.

6.1. Posibilidades del uso de los datos recolectados

Los análisis presentados en el Capítulo 5 son sólo una pincelada superficial del potencial del software desarrollado. Los datos ya recolectados y los que se pueden recolectar abren un abanico de posibilidades para hacer minería de datos realizando cambios muy pequeños en el software del presente trabajo.

Estudios interesantes de realizar, en el contexto de las elecciones, incluyen conocer la variación de la preferencia de voto de las personas entre la primera y la segunda vuelta. Resulta útil para los candidatos y analístas conocer de que canidadato provienen los nuevos votantes para la segunda vuelta para poder enfocar sus campañas de mejor manera. Esto se puede lograr comparando la preferencia de voto de un mismo usuario antes de la primera y antes de la segunda vuelta.

Al conocer la ubicación desde donde fue emitida una gran fracción de los Tweets, se pueden calcular los porcentajes de preferencia de voto por localidad, para así, al igual que en el caso anterior, poder diseñar la campaña de manera más eficiente. Cabe destacar, que la información se puede obtener de manera casi inmediata (paralelizando el identificador de sentimiento), con lo que no es necesario esperar días o semanas para obtener los resultados de alguna encuesta.

En caso de configurar el software para que recolecte tweets sobre canales de televisión, marcas de productos de consumo y otros, se puede realizar un cruce de datos para conocer las preferencias de los productos de las personas, con los canales de televisión que ven y el horario en el que lo hacen correlacionando sus preferencias en los distintos ámbitos. Esto abre posibilidades a los equipos de marketing para apuntar sus campañas publicitarias al público objetivo deseado.

6.2. Trabajos futuros

El principal problema observado en la implementación de la versión actual del software es la poca confiabilidad en la librería FreeLing. Se recomienda probar con otras librerías de similares características. Un trabajo interesante es la evaluación y comparación cuantitativa de los distintos software de ingeniería de texto (análisis lexico del texto) existentes actualmente en el contexto del castellano Chileno tanto en texto formal (por ejemplo publicaciones en la prensa, etc) como en el lenguaje escrito del día a día, por ejemplo, en las redes sociales, foros, etc. La elección adecuada de dicha componente permitirá mejorar la fracción de aciertos del presente software.

Se recomienda, también, implementar un algoritmo estadístico, entrenado en forma semiautomática en base a textos de los que se conoce el sentimiento asociado, para comparar cuantitativamente dicho enfoque con el basado en una ontología. La complegidad en tal trabajo sería encontrar una forma viable de entrenar el algoritmo sin tener una gran cantidad de datos inicial. Una posibilidad es tomar datos desde Facebook y considerar como positivos aquellos mensajes que tienen una cantidad considerable de "Me gusta".

Aprovechando las capacidades de geolocalizar los tweets, es posible crear un sistema que obtenga información sobre desastres en algún área específica rápidamente. Para esto, sería necesario configurar el stream para recibir tweets con palabras típicamente asociadas a algún desastre y en caso de existir peaks de actividad en una zona geográfica, concluir que tal desastre ocurrió.

Capítulo 7

Bibliografía

[1] Global Web Index. Infografía twitter 2013. Internet. <http://globalwebindex.net/wp-content/uploads/downloads/2013/02/Twitter_GWI_2013.pdf>[Consulta: juunio 2013].

[2] Mr Jack. Estrategia online 2011. Internet. <http://mrjack.cl/uploads/mrjack_2011.pdf>[Consulta: abril 2013].

[3] Tweet archivist. Internet. <http://www.tweetarchivist.com/>[Consulta: mayo 2013].

[4] Twenty feet. Internet. <http://www.twentyfeet.com/>[Consulta: mayo 2013].

[5] Socialbakers. Internet. <http://www.socialbakers.com/>[Consulta: junio 2013].

[6] Brandmetric. Internet. <http://www.brandmetric.com/>[Consulta: agosto 2013].

[7] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *Processing*, pages 1–6, 2009.

[8] Oracle. Rightnow. Internet. <http://www.oracle.com/us/products/applications/rightnow/overview/index.html>[Consulta: mayo 2013].

[9] Oracle. Rightnow datasheet. Internet. <http://www.oracle.com/us/media1/rightnow-social-monitor-1585110.pdf>[Consulta: junio 2013].

[10] A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe. Predicting elections with twitter: What 140 characters reveal about political sentiment. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*, pages 178–185,

2010.

[11] Daniel Gayo-Avello. I wanted to predict elections with twitter and all i got was this lousy paper - a balanced survey on election prediction using twitter data. *CoRR*, abs/1204.6441, 2012.

[12] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, June 1993.

[13] Brandmetric - benchmark candidatos. Internet. <`http://eleccion.brandmetric.com/`>[Consulta: agosto 2013].

[14] Hong Yu and Vasileios Hatzivassiloglou. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP '03, pages 129–136, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[15] Janyce Wiebe. Learning subjective adjectives from corpora. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 735–740. AAAI Press, 2000.

[16] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86, 2002.

[17] Albert Bifet and Eibe Frank. Sentiment knowledge discovery in twitter streaming data. In *Proceedings of the 13th international conference on Discovery science*, DS'10, pages 1–15, Berlin, Heidelberg, 2010. Springer-Verlag.

[18] Peter D. Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 417–424, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[19] Dmitry Kan. Rule-based approach to sentiment analysis at romip 2011. 2011.

[20] Rudy Prabowo and Mike Thelwall. Sentiment analysis: A combined approach. *J. Informetrics*, 3(2):143–157, 2009.

[21] Khin Phyu Phyu Shein. Ontology based combined approach for sentiment classification. In *Proceedings of the 3rd International Conference on Communications and Information Technology*, CIT'09, pages 112–115, Stevens Point, Wisconsin, USA, 2009. World Scientific and Engineering Academy and Society (WSEAS).

[22] Wei Wei and Jon Atle Gulla. Sentiment learning on product reviews via sentiment ontology tree. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 404–413, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[23] Efstratios Kontopoulos, Christos Berberidis, Theologos Dergiades, and Nick Bassiliades. Ontology-based sentiment analysis of twitter posts. *Expert Systems with Applications*, 40(10):4065 – 4074, 2013.

[24] Varios autores. Freeling. Internet. <http://nlp.lsi.upc.edu/freeling/>[Consulta: mayo 2013].

[25] Taylor Singletary. Filtered stream - is it filtering out of 1 % sample stream? Internet. <https://dev.twitter.com/discussions/7140>[Consulta: junio 2013].

[26] The University of Sheffield. General architecture for text engineering. Internet. <http://gate.ac.uk/>[Consulta: mayo 2013].

[27] Google. Google geocode api. Internet. <https://developers.google.com/maps/documentation/geocoding/>[Consulta: noviembre 2013].

[28] Google. Google charts. Internet. <https://developers.google.com/chart/>[Consulta: julio 2013].

[29] Varios Autores. Bootstrap. Internet. <http://getbootstrap.com/css/>[Consulta: noviembre 2013].

[30] George Williams. Host a website on google drive. Internet. <http://chronicle.com/blogs/profhacker/host-a-website-on-google-drive/46737>[Consulta: noviembre 2013].

[31] Google. Google script. Internet. <https://script.google.com/>[Consulta: noviembre 2013].

[32] Google. Google maps api v3. Internet. <https://developers.google.com/maps/documentation/javascript/>[Consulta: octubre 2013].

[33] Varios autores. Freeling - downloads. Internet. <http://devel.cpl.upc.edu/freeling/downloads?order=time&desc=1>[Consulta: octubre 2013].

[34] Servicio Electoral de Chile. Resultados de la segunda votacion presidencial - 15 de diciembre. Internet. <http://www.eleccionesenchile.com/resultados-electorales-chile.php>[Consulta: diciembre 2013].

[35] Brandmetrics. Internet. <http://eleccion.brandmetric.com/segundavuelta2013/>[Consulta: diciembre 2013].

[36] Varios autores. 1 % of the firehose via streaming api even when only searching very specific keywords? Internet. <https://dev.twitter.com/discussions/12692>[Consulta: mayo 2013].

[37] Varios autores. IntroducciÓn a las etiquetas eagles (v. 2.0). Internet. <http://nlp.lsi.upc.edu/freeling/doc/tagsets/tagset-es.html>[Consulta: agosto 2013].

[38] Louis Rosenfeld and Peter Morville. *Information Architecture for the World Wide Web*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 3rd edition, 2006.

Apéndice A

API de FreeLing

FreeLing es la librería usada para hacer el análisis del texto. En este anexo se explica el procedimiento de su instalación y su uso desde Java.

A.1. Instalación de FreeLing

La librería FreeLing se puede descargar como un paquete precompilado para la mayoría de los sistemas operativos más populares. Al instalarla usando uno de estos paquetes, quedan disponibles las API para programar en C y C++.

Para poder usar la API desde java, es necesario compilar e instalar FreeLing, instalar librerías JNI headers y a continuación compilar los módulos de enlace con el jdk específico instalado y declarar algunas variables con los directorios donde quedó instalada la librería.

La instalación de la librería misma se reduce a configurar en un *Makefile* las rutas de algunas librerías de dependencia (en caso de ser necesario, éstas se deben instalar) y, simplemente, ejecutar el comando *make*.

En resumen, los comandos a ejecutar son:

```
sudo apt-get install build-essential automake autoconf
sudo apt-get install libboost-regex-dev libicu-dev
sudo apt-get install libboost-filesystem-dev libboost-program-options-dev

tar xzvf FreeLing-3.0.tar.gz
cd freeling-3.0
```

```
./configure
make
sudo make install

cd API/java
./configure
make
export FREELINGDIR=/usr/local
```

La librería viene con una aplicación en java de ejemplo "analyzer.java". Este programa se compila y ejecuta normalmente, asegurándose que el jar de freeling está en el classpath de java.

A.2. Procesamiento del texto

Para procesar una oración de texto en FreeLing, la librería debe ser inicializada. Este código es parte del programa de ejemplo en java que viene incluido con FreeLing:

```
String FREELINGDIR = "/usr/local";
String DATA = FREELINGDIR + "/share/freeling/";
String LANG = "es";

// Create options set for maco analyzer.
// Default values are Ok, except for data files.
MacoOptions op = new MacoOptions(LANG );
//op.setActiveModules(false, true, true, true, true, true, true,
                        true, true, true, false );
op.setActiveModules(false, true, true, true, true, true, true,
                        true, true, true, true );

op.setDataFiles(
                "",
                DATA + LANG + "/locucions.dat",
                DATA + LANG + "/quantities.dat",
                DATA + LANG + "/afixos.dat",
                DATA + LANG + "/probabilitats.dat",
                DATA + LANG + "/dicc.src",
                DATA + LANG + "/np.dat",
                DATA + "common/punct.dat",
                DATA + LANG + "/corrector/corrector.dat" );
```

```
// Create analyzers.
tk = new Tokenizer( DATA + LANG + "/tokenizer.dat" );
sp = new Splitter( DATA + LANG + "/splitter.dat" );
mf = new Maco( op );

tg = new HmmTagger( LANG, DATA + LANG + "/tagger.dat", true, 2 );
parser = new ChartParser(
                DATA + LANG + "/chunker/grammar-chunk.dat" );
dep = new DepTxala( DATA + LANG + "/dep/dependences.dat",
                parser.getStartSymbol() );
neclass = new Nec( DATA + LANG + "/nec/nec-ab.dat" );

dis = new UkbWrap( DATA + LANG + "/ukb.dat" );
```

Se configura la librería para funcionar en idioma castellano y se inicializan todos los módulos disponibles.

Para obtener el árbol de dependencia, se debe ejecutar un código similar a:

```
// Extract the tokens from the line of text.
ListWord l = tk.tokenize( line );

// Split the tokens into distinct sentences.
ListSentence ls = sp.split( l, false );

// Perform morphological analysis
mf.analyze( ls );

// Perform part-of-speech tagging.
tg.analyze( ls );

// Perform named entity (NE) classificiation.
neclass.analyze( ls );

// sen.analyze(ls);
dis.analyze( ls );

// Chunk parser
parser.analyze( ls );

// Dependency parser
dep.analyze( ls );
```

A.3. Estructura del árbol de dependencia de FreeLing

Después de ejecutar el código anterior, FreeLing deja toda la información obtenida del texto procesado en el objeto ls (ListSentence), que es similar a un *ArrayList* de java. Cada elemento de esta lista es de tipo *Sentence* y tiene la información sobre una sola oración.

El objeto *Sentence* tiene un método para obtener el árbol de dependencia:

```
TreeDepnode root = sentence.getDepTree();
```

Los nodos *TreeDepnode* tienen un contenedor con la información de las palabras que representan y un conjunto de hijos. Para obtener la información de una palabra, se deben usar los siguientes métodos:

```
Word w = root.getInfo().getWord(); /* Obtiene toda la información de
      una palabra*/
String wordForm = w.getForm(); /* La palabra tal, cual aparece en la
      oración  inicial
String wordLemma =w.getLemma(); /* La palabra en su forma básica
      (en infinitivo, genero masculino, singular, etc)*/
String wordTag =w.getTag(); /* Etiqueta de la palabra según la
      notación EAGLES (v. 2.0)*/
```

La etiqueta en notación EAGLES [37] permite identificar la morfología de la palabra. Se puede identificar si es un verbo, nombre, adverbio, pronombre, conjunción, adjetivo, etc. Además de información más específica como el genero, si la palabra esta en plural o singular, etc.

Apéndice B

Script que copia páginas del servidor a Google Drive

El siguiente código fue desarrollado para descargar dos páginas desde el servidor donde se ejecutaba la aplicación, que no estaba preparado para ser accedido masivamente, y guardarlas en Google Drive. Con esto, Google Drive actúa con un Proxy con Caché, al que se pueden dirigir los usuarios masivamente.

El código se ejecuta en los servidores de Google Script.

```
function downloadTwitterMeter() {
  var folderName = 'Twittermeter';
  var fileName = 'index.htm';
  var fileName2 = 'newIndex.htm';

  var fileNameSentiment = 'view_sentiments.htm';
  var fileNameSentiment2 = 'newview_sentiments.htm';

  var fileURL = 'http://cruchaga.bounceme.net:1080/TwitterMeterJSF
                            /faces/index.jsp';
  var fileURLSentiment = 'http://cruchaga.bounceme.net:1080/
                          TwitterMeterJSF/faces/view_last_sentiments.jsp';

  var folder = DocsList.getFolder(folderName);

  DocsList.createFile(fileName2, UrlFetchApp.fetch(fileURL)).
                         addToFolder(folder);
  while(countDocByName(folderName, fileName2) < 1){
    Utilities.sleep(10000);
    DocsList.createFile(fileName2, UrlFetchApp.fetch(fileURL)).
```

```
                        addToFolder(folder);
  }

  deleteDocByName(folderName, fileName);
  renameFile(folderName, fileName2, fileName);

  DocsList.createFile(fileNameSentiment2, UrlFetchApp.fetch(
                        fileURLSentiment)).addToFolder(folder);
  while(countDocByName(folderName, fileNameSentiment2) < 1){
    Utilities.sleep(10000);
    DocsList.createFile(fileNameSentiment2, UrlFetchApp.fetch(
                        fileURLSentiment)). addToFolder(folder);
  }

  deleteDocByName(folderName, fileNameSentiment);
  renameFile(folderName, fileNameSentiment2, fileNameSentiment);
}



function renameFile(FolderName, fileName, newFilename){
  var docs=DocsList.find(fileName)
  for(n=0;n<docs.length;++n){
    var folders = docs[n].getParents();
    for(f=0;f<folders.length;f++){
      if(folders[f].getName() == FolderName && docs[n].getName() ==
                        fileName){
        var ID = docs[n].getId()
        DocsList.getFileById(ID).rename(newFilename);
      }
    }
  }
}


function countDocByName(parentFolderName, fileName){
  var docs=DocsList.find(fileName)
  var count = 0;
  for(n=0;n<docs.length;++n){
    var folders = docs[n].getParents();
    for(f=0;f<folders.length;f++){
      if(folders[f].getName() == parentFolderName && docs[n].getName() ==
                        fileName){
        count++;
      }
    }
  }
```

```
  return count;
}



function deleteDocByName(parentFolderName, fileName){
  var docs=DocsList.find(fileName)
  for(n=0;n<docs.length;++n){
    var folders = docs[n].getParents();
    for(f=0;f<folders.length;f++){
      if(folders[f].getName() == parentFolderName && docs[n].getName() ==
                        fileName){
        var ID = docs[n].getId()
        DocsList.getFileById(ID).setTrashed(true)
      }
    }
  }
}
```

Apéndice C

Datos de validación del método propuesto para identificar la ubicación de los usuarios de Twitter

La siguiente tabla muestra la cantidad de Tweets para los que se pudo identificar una ubicación por GPS para cada región, la cantidad de estos que se pudieron identificar como pertenecientes a la región usando el dato de ubicación del perfil del usuario y los porcentajes de acierto del método.

Región	Cantidad total por perfil	Cantidad total por GPS	Intersección de ambos	Confiabilidad de dato obtenido	Fracción de localidades identificadas
Chile	27950	54315	27832	99,6 %	51,2 %
I	508	924	447	88,0 %	48,4 %
II	1101	1779	1039	94,4 %	58,4 %
III	323	568	296	91,6 %	52,1 %
IV	931	1456	834	89,6 %	57,3 %
V	3217	5206	2800	87,0 %	53,8 %
VI	828	1904	640	77,3 %	33,6 %
VII	1560	2097	996	63,8 %	47,5 %
VIII	3434	5636	2922	85,1 %	51,8 %
IX	942	1359	728	77,3 %	53,6 %
X	909	1343	819	90,1 %	61,0 %
XI	299	136	68	22,7 %	50,0 %
XII	380	776	317	83,4 %	40,9 %
XIV	389	835	343	88,2 %	41,1 %
XV	224	401	201	89,7 %	50,1 %
RM	15820	30956	14734	93,1 %	47,6 %