UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA CIVIL EN COMPUTACIÓN

SISTEMA DE DETECCIÓN DE TENDENCIAS EN LA WEB 2.0 BASADO EN ALGORITMOS DE CALIFICACIÓN DE RELEVANCIA Y SELECCIÓN AUTOMÁTICA DE FUENTES DE DOCUMENTOS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

RODRIGO ALFONSO DUEÑAS FERNÁNDEZ

PROFESOR GUÍA:
JUAN D. VELÁSQUEZ SILVA

MIEMBROS DE LA COMISIÓN:
JOSÉ MIGUEL PIQUER GARDNER
PABLO BARCELÓ BAEZA

SANTIAGO DE CHILE
NOVIEMBRE 2012

Resumen

El objetivo de esta memoria es solucionar las falencias de un sistema informático para la detección de tendencias en la Web de la consultora Duam S.A., a través del rediseño de su arquitectura de hardware y software, y la modificación de sus algoritmos de extracción de conocimiento.

Haciendo uso de este sistema informático, Duam S.A. ofrece el servicio de inteligencia de mercado a variadas empresas, el cual consiste en la realización de un informe donde se analiza el mercado en el cual están inmersas estas instituciones, señalando sus principales competidores, posibles amenazas, oportunidades de inversión, nuevas tecnologías, etc. Este estudio es realizado a través del minado de documentos desde blogs y sitios de noticias, junto con el análisis de las opiniones vertidas en la Web por parte de los usuarios de las redes sociales.

Debido a la alta cohesión del código utilizado por el sistema, replicar los procesos internos era una tarea ardua que involucraba altos costos para la empresa, por lo que no era factible dar abasto al minado de cientos de fuentes de documentos o al análisis de múltiples temáticas. Por otro lado, sus algoritmos de extracción de conocimiento no aprovechaban toda la información presente en los documentos recuperados. Además, el uso de una arquitectura física sin separación de capas, no permitía que el sistema escale eficientemente, por lo que la plataforma no era capaz de escalar acorde a la cantidad de fuentes. Por esta misma razón, el exponer la información recuperada para otras aplicaciones mermaba el rendimiento total de la aplicación.

Como solución a esta problemática, se desarrollaron nuevos algoritmos de extracción de información y conocimiento desde la Web 2.0 basándose en dos hipótesis: la primera es que es posible obtener fuentes presentes en web que aporten a los resultados obtenidos a partir del análisis de la información existente en los documentos minados por el sistema. La segunda hipótesis es que es posible hacer uso de los emoticones que se encuentran en los documentos opinados para obtener mejores resultados en la extracción de opiniones a partir de estos.

Para solucionar los problemas de escalabilidad del sistema se rediseñó la arquitectura considerando múltiples capas de procesamiento en donde todas las componentes están comunicadas y orientada a servicios.

Para remediar las falencias del sistema desde el punto de vista de software se modularizó cada una de sus componentes y se planteó una arquitectura para aplicaciones orientadas a terceros con el fin de que estas no mermen su rendimiento. Además, se implementó un algoritmo de calificación de relevancia y selección automática para fuentes de documentos para que la información recuperada retroalimente al sistema. Finalmente, se modificó el algoritmo de extracción de opiniones para que hiciera uso de los emoticones presentes en los documentos opinados a la hora de clasificar éstos según su polaridad.

El rediseño de la arquitectura del sistema resultó ser exitoso, reduciendo el uso de recursos, permitiendo la escalabilidad del sistema y además, la replicación de sus procesos internos para analizar múltiples temáticas a la vez.

En cuanto al modelo de minado de opiniones, este obtuvo mejores resultados que el original. Al detectar opiniones positivas, el recall aumento de un 0.59 a un 0.62, y la precision de 0.6 a 0.67; mientras que al detectar opiniones negativas, el recall y la precision aumentaron en 0.02 y 0.03 respectivamente. Para el algoritmo de calificación de relevancia, se obtuvo un recall de 0.62 y una precision de 0.35, que están dentro de los rangos esperados debido a la gran cantidad de ruido inducida por nombres de dominio que aparecen pocas veces entre los documentos minados.

Se concluye que realizar el rediseño de la plataforma fue beneficioso para la empresa, desde el punto de vista de negocio, debido a la reducción en costo humano que esta implica, y además, la mejora en los resultados entregados por el algoritmo de minado de opiniones permite una mejor apreciación del sistema por parte del usuario final. Así mismo, la posibilidad de incluir aplicaciones orientadas a terceros abre nuevas posibilidades de negocio para la empresa.

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Introducción

En el año 1991 nació la World Wide Web [5], también conocida como “La Web", con el fin de crear un repositorio del conocimiento humano para que todo usuario pudiese ser consumidor de información. Cumpliendo tal objetivo, si bien ya era una herramienta potente para la sociedad, al ser sólo utilizada para presentar información a los usuarios, aún se estaba muy lejos de alcanzar el verdadero potencial que su creador, Tim Berners Lee, pretendía que tuviera.

Con la aparición de las aplicaciones web que permiten la creación de contenido y la colaboración por parte de los usuarios, como lo son *wikis* y *blogs*, (las cuales darían el puntapié inicial a lo que ahora se conoce como la Web 2.0) la función de la WWW en la sociedad mundial se vio fuertemente potenciada, ya que dejó de ser tan sólo un repositorio de información, transformándose en un canal interactivo entre las todas las entidades que la componen, tanto usuarios como proveedores de información. Este cambio de paradigma permitió que todos ellos pudiesen contribuir activamente en la creación de contenido, provocando un explosivo aumento en la participación de sus usuarios en la Web, y por consiguiente, de la cantidad de información y conocimiento disponible en ella.

En los últimos años, el enfoque social de la Web fue impulsado aún más por el explosivo crecimiento de las llamadas *redes sociales* [38, 39], las que permitieron que sus usuarios compartieran opiniones e información, complementando los contenidos presentes en los sitios web con la expresión escrita de las opiniones de sus usuarios, los cuales dan a conocer sus sentimientos y percepciones respecto de una variedad de temas, algunos de estos analizados en los mismos sitios visitados y otros propuestos para su desarrollo a través de estas verdaderas “conversaciones virtuales".

Debido al crecimiento sin par de la población en el último siglo, y la manera en que el comercio se ha desarrollado en las pasadas décadas, se ha hecho sumamente necesario ser altamente competitivo si una empresa desea lograr alta rentabilidad cualquiera sea el nicho en el que esta se desenvuelva. En base a lo anterior han nacido múltiples iniciativas que buscan aprovechar toda la información que posee una empresa para aumentar su rendimiento y rentabilidad, sea disminuyendo costos o

aumentando la eficiencia con que se realizan sus procesos en base al análisis de la información interna de la empresa.

Aún cuando una empresa haga uso constante de este tipo de herramientas para optimizar sus procesos internos, ésta no sólo se ve afectada por variables endógenas a su funcionamiento, también se ve influenciada por una serie de factores exógenos, como lo son los nuevos competidores, fluctuaciones el mercado, posibles nuevos mercados que se pueden explotar y muchos otros. Las consecuencias que estos factores tienen en una empresa sólo pueden manipulados si se hace un análisis exhaustivo de la información presente en el exterior de la empresa.

A la larga, este análisis del entorno de una empresa se vuelve complejo y costoso de realizar de manera periódica y manual, razón por la cual la consultora nacional Duam S.A.[1] ofrece el servicio de inteligencia de mercado, donde utiliza a profesionales con experiencias en variadas áreas para realizar estas investigaciones con el fin de permitirle a las empresas que hagan uso de este servicio monitorear el panorama competitivo bajo el que se encuentran junto con explorar nuevas oportunidades de inversión y expansión de su negocio. Aún así, la gran cantidad de tiempo y las limitantes que conllevan la realización manual de un análisis de inteligencia de mercado hacen que ofrecer el servicio como tal se contempla en la actualidad no sea sostenible en el tiempo, lo que motivó el desarrollo de una herramienta que apoye la detección de tendencias en la Web.

1.1. Planteamiento del Problema y Motivación

Debido a las características del panorama competitivo en la actualidad, una empresa puede crecer en muchas direcciones y no sólo aumentando la cantidad de productos que produce o el número de personas a las que ofrece sus servicios. Así, a medida que una empresa va expandiendo sus negocios hacia nuevos mercados o comienza a ofrecer nuevos productos y servicios, si desea obtener rendimientos sobre el promedio la cantidad de información externa que debe abarcar para poder realizar un análisis acabado de su entorno se vuelve cada vez mayor, por lo que se requiere analizar un conjunto siempre creciente de fuentes para poder recuperar el conocimiento necesario para que este sea valioso para la empresa.

Por ello se vuelve sumamente necesario que un sistema que tenga como objetivo ayudar la toma de decisiones a través del análisis de las tendencias presentes en la Web, sea capaz de manejar grandes volúmenes de datos para gestionar de la mejor manera posible los recursos que se disponen, y al mismo tiempo reducir el nivel de dependencia que sus componentes poseen para que el sistema pueda escalar en capacidad de múltiples maneras o desde una variedad de ubicaciones sin que estas

[1] `http://www.duam.cl`

se vean atadas por restricciones de ubicación o de dependencia entre sus componentes.

Además, si bien un experto en un área puede ser capaz de saber a priori muchas fuentes que son útiles para la detección de tendencias, es muy probable que en la práctica se estén dejando un sinfín de fuentes sin minar sólo porque el experto que definió el conjunto de fuentes a utilizar no las consideró útiles bajo su criterio. Este tipo de problemas pueden provocar que los resultados de la plataforma estén sesgados a lo que conoce el o los expertos que realizaron el análisis del ecosistema desde el cual se está procesando la información.

A pesar de todo lo ya expuesto, la plataforma de detección de tendencias que posee Duam S.A. no fue diseñada con el objetivo de analizar grandes cantidades de información ni analizar múltiples mercados o nichos concurrentemente por lo que la motivación de esta memoria de título es mejorar dicho sistema para que sean capaces de sobrepasar estas limitaciones y además complementarlas, por un lado otorgándole al sistema la capacidad de aumentar la cantidad de fuentes siendo analizadas gracias a la detección automática de nuevas fuentes y por otro, permitiendo que los algoritmos de extracción de opiniones utilizados en ella consideren el uso de emoticones en las redes sociales como información valiosa a la hora de recuperar conocimiento a partir de ellas.

A lo largo de esta memoria de título se trabajarán sobre dos hipótesis, la primera de ellas consiste en que es posible, a partir de la información presente en los documentos siendo recuperados por la plataforma de detección de tendencias, encontrar nuevas fuentes que puedan aportar información al sistema; y la segunda hipótesis consiste en que el uso de los emoticones presentes en los documentos opinados recuperados por la plataforma mejorarán los resultados de los algoritmos de minado de opiniones.

1.2. Objetivos

1.2.1. Objetivo General

"Rediseñar y expandir plataforma de detección de tendencias una empresa de análisis de inteligencia de mercado para permitir su escalabilidad y mejorar el conocimiento recuperado."

1.2.2. Objetivos Específicos

- Establecer el estado del arte sobre los algoritmos de detección de comunidades y las metodologías utilizadas en desarrollo de sistemas escalables.
- Caracterizar la situación actual de la plataforma de detección de tendencias.
- Determinar los requisitos que la nueva plataforma debe cumplir en base a las falencias detectadas en el análisis de la situación actual.

- Plantear el rediseño de la arquitectura de la plataforma en base a los requisitos extraídos.
- Diseñar e implementar un algoritmo de calificación de relevancia de nuevas fuentes en base a los enlaces presentes en los documentos recuperados por la plataforma.
- Modificar el actual algoritmo de *Web Opinion Mining* de manera tal que considere el uso de emoticones a la hora de asignar un puntaje de opinión a cada documento recuperado.

1.3. Justificación del trabajo de título

Tal como se dió a conocer en los objetivos específicos de esta memoria de título, esta tiene una serie de desafíos en múltiples áreas como lo son el rediseño de procesos y de software, la implementación, puesta en marcha y evaluación de nuevos algoritmos y la escalabilidad de sistemas, debido principalmente al nivel de los flujos de datos que se busca que la plataforma sea capaz de analizar luego de su rediseño, ya que el software no debe apuntar a ser utilizado como una herramienta de análisis de información tal como lo es hoy en día, sino como un motor de procesamiento de datos que sustente distintas aplicaciones comerciales como también el análisis de múltiples temáticas o mercados según las necesidades de la empresa lo dicten.

Si bien la plataforma posee un conjunto de procesos de negocio bien definidos, es necesario rediseñar la plataforma para que cada uno de ellos pueda funcionar sin depender directamente de los otros a menos que sea realmente necesario (como es en el caso de los algoritmos de extracción de tópicos que se darán a conocer en el capítulo siguiente) y además, que sea posible el funcionamiento en paralelo de múltiples instancias de cada módulo sin que esto dañe los resultados obtenidos.

La primera dificultad que plantea este trabajo de título es hacer un análisis extenso de la situación actual de la plataforma tanto desde el punto de vista de procesos de negocio como del diseño de software. Debido a la alta complejidad del sistema y de la variedad de componentes que la conforman, es necesario ser muy cauteloso en separar cada componente y darlas a conocer de manera simple y concisa para así poder plantear un rediseño que se ajuste a las necesidades de la plataforma y que reutilice de una manera eficiente lo ya desarrollado.

Una vez ya definida la situación actual de la plataforma sale a la luz otra de las dificultades que se debe enfrentar al realizar un rediseño, la caracterización de los requisitos a satisfacer, el conjunto de los cuales debe ser definido de manera tal que el rediseño impacte de la mejor manera posible el flujo de trabajo actual que la empresa posee actualmente para el uso de la plataforma. Además, es necesario definir los requisitos que la plataforma ya satisface ya que no se posee una lista de estos de manera previa, lo que hace aún más crucial la definición de estos de una manera clara y con un

alto nivel de cobertura de las funcionalidades ya existentes.

Además, el modificar un software desarrollado previamente involucra una serie de desafíos ya que hay que asegurarse de que los resultados entregados por el software rediseñado sean consistentes con los que se recibían desde la versión antigua del mismo junto con asegurarse de seguir cumpliendo los requisitos impuestos previamente por el usuario y poder lograr de una manera correcta lo que se ha propuesto para mejorar la plataforma existente.

Debido a que será necesario agregar un nuevo algoritmo a la plataforma y modificar uno que la plataforma ya posee, será necesario definir pruebas capaces de demostrar que las soluciones propuestas funcionan de la manera que se esperaba y que además los cambios introducidos en la plataforma significan una mejora sobre los resultados del algoritmo original.

Las hipótesis de investigación que se desean verificar en esta trabajo son dos, la primera de ellas es que es posible obtener nuevas fuentes que aporten información a los resultados obtenidos por la plataforma a partir del análisis de los documentos presentes actualmente en ella. La segunda hipótesis es que es posible hacer uso de los emoticones presentes en los documentos de las redes sociales para obtener mejores resultados en la extracción de opiniones a partir de éstos.

Por último pero no menos importante, es el desafío profesional que propone el llevar a cabo un proyecto de esta naturaleza ya que no sólo se enfoca en el desarrollo de algoritmos o de componentes para un sistema, también involucra el analizar una plataforma previamente desarrollada desde el punto de vista de software y como una unidad de una empresa compuesta por una serie de procesos de negocio, lo cual se asemeja mucho a un proyecto que un ingeniero civil en computación debe ser capaz de llevar a cabo en su vida profesional.

1.4. Metodología y Plan de Trabajo

En esta sección se darán a conocer las metodologías que se utilizarán para el desarrollo de este trabajo de título y el plan de trabajo que se llevará a cabo durante la duración completa de este.

1.4.1. Metodología de trabajo

Desarrollo Ágil de Software

Para el diseño y el desarrollo del software se utilizará una metodología ágil de desarrollo de software. Este tipo de metodologías consisten en realizar un proceso de desarrollo altamente integrado con el cliente, realizando pequeñas iteraciones en las cuales se realiza un ciclo de desarrollo de software tradicional con pequeñas variaciones que permiten que el proceso sea más flexible [33].

En general el proceso de desarrollo ágil de software, tal como se muestra en la figura 1.1, es como

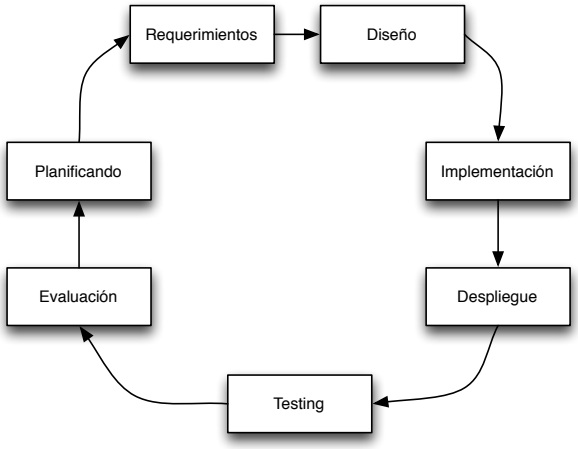

Figura 1.1: Desarrollo Ágil de Software.

sigue:

- De manera continua se planifican entregables a corto plazo en cada iteración, permitiendo una fuerte correlación entre el producto que se está desarrollando y lo que realmente necesita el cliente gracias a que permite poca desviación entre lo que percibe el equipo de desarrollo y lo que piensa el cliente.
- Se realizan reuniones diarias o con una frecuencia alta en las cuales se informa sobre el estado del proyecto, se sugieren mejoras, pequeños cambios en los requisitos y se expone cualquier otra información que se considere vital para el éxito del proyecto.
- En cada iteración se hace una revisión de lo entregado y se dan nuevos requisitos o se recalculan las prioridades de requisitos ya existentes

El gran valor de utilizar las metodologías ágiles para este trabajo de título es que este ciclo continuo de generación y evaluación de requisitos permite que a lo largo del proyecto se desperdicie poco tiempo en el desarrollo de requisitos que agregan poco valor al cliente e impide tiempos muertos, ya que siempre se tiene tareas por realizar que se sabe de antemano que son valoradas por el cliente. Además, cada paso importante que se realice podrá ser validado en el corto plazo, tanto por el cuerpo docente como por el cliente, permitiendo que cada objetivo y contribución que se considere agregar a este trabajo de título sea de real valor para ambos.

Metodología CRISP-DM

Para llevar a cabo todo el proceso de recuperación de información y minado de datos se utilizará la metodología CRISP-DM [55] que consiste en seis etapas que son: Comprensión del problema,

exploración de los datos y las fuentes de información, preparación de los datos, modelado de la información, evaluación del modelo y desarrollo. Las principales razones para la elección de esta metodología son que permite realizar un análisis de la problemática empresarial para luego transformar esto en un problema técnico, y que por otra parte es una metodología neutra respecto a la herramienta a utilizar para realizar el minado de datos.

Además, esta metodología se acopla de manera muy eficiente con las metodologías ágiles de desarrollo de software, ya que en cada iteración de desarrollo se puede realizar un ciclo completo de CRISP-DM y así lograr un mayor valor de los datos obtenidos en cada paso que se lleve a cabo.

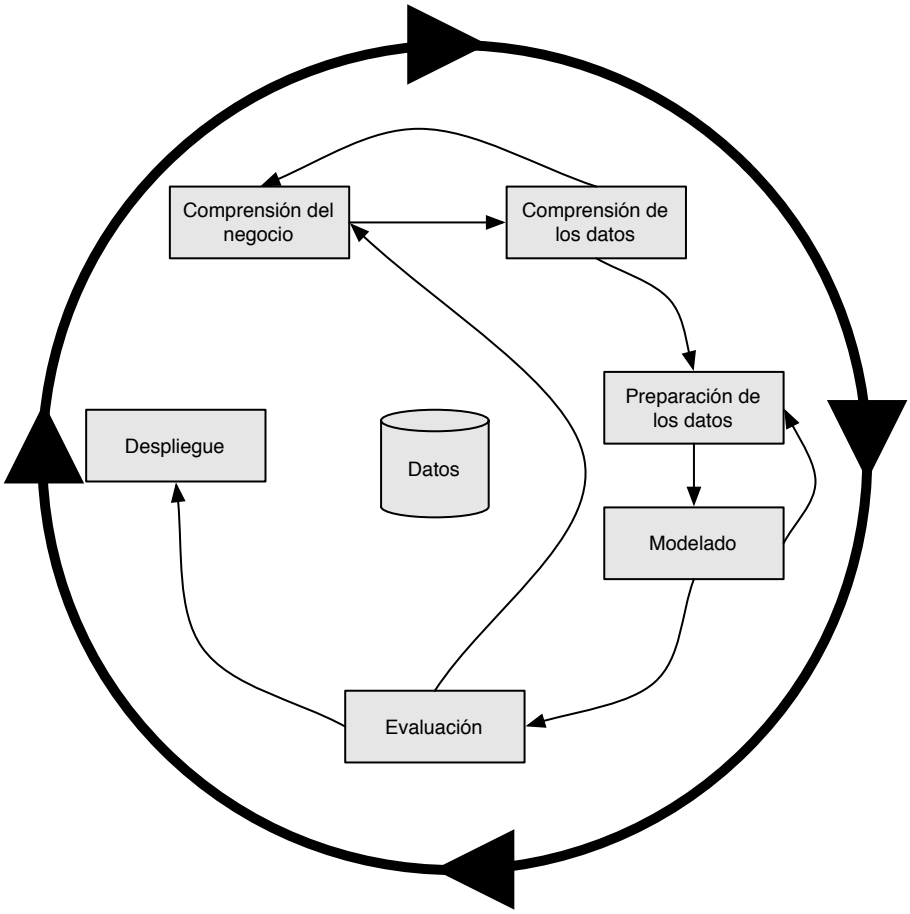

Figura 1.2: Metodología CRISP-DM

Debido a que la gran mayoría de los trabajos de título son proyectos de investigación y descubrimiento sobre áreas del conocimiento, los requerimientos para estos son siempre muy volátiles, lo que favorece el uso de metodologías como las ya presentadas. En el marco de este trabajo, cada iteración de trabajo se desarrollará de la siguiente manera:

- Estudiar las fuentes de información a analizar y el aporte que estas conllevan para el trabajo en su totalidad.

- Definir la manera en que se almacenarán estos datos, y todos los procesos de limpieza y extracción de *metadata* asociados a cada una de estas fuentes. Es importante señalar qué datos de cada documento se almacenarán y cuál es el objetivo de mantener tal información en los sistemas de almacenamiento de datos del proyecto. En este paso de la iteración se debe definir el modelo de datos y los procesos de limpieza, como por ejemplo remoción de *stop-words* y código html, que se realizarán sobre los datos obtenidos desde la web.

- Definir posibles cambios en la manera que se utilizarán los datos a lo largo del proyecto, para minimizar la cantidad de trabajo perdido sobre los datos, o diseñar los modelos de datos de manera tal que permitan otorgar funcionalidades extras de manera con un gasto en tiempo de desarrollo bajo.

- Definir qué modelos de los ya investigados se utilizarán para el correcto desarrollo de esta iteración, tomando una decisión fundamentada en los pasos anteriores y en una discusión integrada entre el cuerpo docente y el equipo a cargo del proyecto en el lado del cliente.

- Realizar el desarrollo

5. Realizar el proceso de evaluación, en este caso se espera calcular la eficacia de los modelos sin utilizar los datos extras de LDA y con LDA, analizando recall, precisión y F-measure.

Definidos los resultados anteriores se procederá a realizar la implementación de un software que permita la recepción de posts para blogs, la integración de los modelos clasificadores y utilizando estos resultados, una interfaz que permita publicar o no dicho contenido en una plataforma. La definición exacta de este proceso se espera abarcar en la segunda parte del trabajo de título.

1.4.2. Plan de Trabajo

El plan de trabajo que se utilizará para llevar a cabo el proyecto propuesto en esta memoria de título se divide en los siguientes pasos:

1. **Definición de la situación actual**

   Se analizará la plataforma actual desde dos puntos de vista: el de software, tanto la arquitectura que la sustenta y cómo está implementada, y el de procesos de negocio. En el análisis del software se buscará determinar qué tipo de arquitectura utiliza el sistema, tanto física como de software, y los puntos críticos que provocan que la plataforma no sea escalable. Además, se buscará definir todas las componentes que pueden ser rediseñadas con el objetivo de mejorar su eficiencia o reducir la cohesión entre cada una de las componentes del sistema. Desde el punto

de vista de los procesos de negocio que existen en la plataforma, se tratará de modelar cómo funciona actualmente y determinar si es posible realizar algún cambio en ellos que impacte de manera significativa el rendimiento de la plataforma.

2. **Investigación de trabajo relacionado**

   Se realizará un sondeo de lo que se necesita saber para poder realizar el rediseño de la plataforma y posteriormente se realizará un estudio de estos para poder establecer el estado del arte de las técnicas y las metodologías que se utilizarán para satisfacer los requisitos recopilados después del análisis de la situación actual de la plataforma.

3. **Definición de requisitos para el rediseño**

   Una vez se haya definido la situación actual de la plataforma y se haya realizado un estudio del trabajo relacionado a los problemas que se detectaron y las nuevas funcionalidades que se desean implementar se definirán los requisitos que la nueva plataforma debe satisfacer. Estos requisitos serán extraídos a partir de los que satisface inicialmente la plataforma y todos aquellos que puedan ser desprendidos a partir del análisis realizado de las falencias de ella.

4. **Propuesta de un rediseño para la plataforma**

   Se propondrá un rediseño para la plataforma en cada una de las siguientes áreas si el caso lo amerita: modelo de datos utilizado para almacenar los datos, arquitectura física que sustenta el sistema, arquitectura de software utilizada para el desarrollo de la plataforma y finalmente el desarrollo de cada una de las componentes que son utilizadas para la extracción y la recuperación de la información.

5. **Diseño e implementación del algoritmo de calificación de relevancia para nuevas fuentes**

   Se propondrá e implementará un algoritmo de calificación de nuevas fuentes en base a relevancia de enlaces con el fin de agregar nuevas fuentes de manera semi-supervisada para que sean minadas por la plataforma.

6. **Implementación de las modificaciones propuestas al modelo de extracción de opiniones**

   Se realizarán modificaciones al actual algoritmo de extracción de opiniones de manera tal que tome en cuenta los emoticones en los documentos recuperados para determinar el puntaje de opinión que se le asignará a cada uno de ellos.

7. **Evaluación de la plataforma y de los nuevos algoritmos incluidos en ésta**

Se evaluará cada uno de los algoritmos implementados o modificados para evaluar la calidad de la información recuperada por los nuevos algoritmos y si es que las hipótesis se cumplen o no. Además, se evaluará la plataforma en su totalidad para determinar cómo se comporta frente a distintos volúmenes de datos.

8. **Conclusiones**

   Finalmente, se darán a conocer las conclusiones extraídas durante el desarrollo del proyecto, las cuales dan a conocer lo aprendido, los resultados obtenidos y cómo estos logran cambiar el desempeño de la plataforma y la calidad de la información que se obtiene a través de ella. Además, se dará a conocer posibles ramas de extensión de este trabajo para dar cabida a nuevos proyectos a partir de esta memoria de título.

1.5. Alcances

El producto que se desprenderá de esta memoria de título es el software rediseñado junto con un conjunto de lineamientos de como debe ser el ambiente y la arquitectura física donde la nueva plataforma estará en producción. Sin embargo, se realizarán pequeños simulacros de puesta en marcha de la plataforma con el fin de probar las variaciones en el rendimiento que los cambios propuestos logran.

1.6. Contribuciones de la memoria

Una vez realizado el rediseño de la plataforma, el rendimiento de algunas componentes del sistema mejoró y la escalabilidad de la plataforma permite el aumento de la capacidad de procesamiento de información de esta cuánto sea necesario en todos los módulos del sistema a excepción del módulo de extracción de tópicos, el cual no representa un cuello de botella debido a la periodicidad con el cuál este es usado; la escalabilidad de la plataforma se puede lograr ya sea agregando nuevas máquinas a los módulos del sistema o aumentando el tamaño de las máquinas utilizadas.

Además, en cuanto al nuevo algoritmo para inclusión de fuentes, este es capaz de detectar nuevas fuentes relacionadas con las que se encuentran actualmente en uso y posteriormente evaluar si el agregar estas fuentes a la lista mejora la calidad de la información recuperada. Si bien el modelo es capaz de detectar automáticamente los puntos de acceso a los documentos de estas nuevas fuentes, esta es una limitación conocida del algoritmo y se utiliza para realizar una supervisión manual de las fuentes potenciales por parte del usuario para evitar la inclusión automática de fuentes indeseables o que no se tenga permiso de su uso y además, verificar que las fuentes extraídas por el algoritmo correspondan a lo que se busca.

Por último, tras las modificaciones realizadas al algoritmo de minado de opiniones, la cantidad de documentos identificados correctamente como positivos o negativos aumentaron al utilizar emoticones en conjunto con el algoritmo original que se usaba para clasificarlos.

1.7. Contexto del Trabajo

Esta memoria de título nace tras un proyecto inicialmente financiado por la consultora Duam - Innovación al Sur del Mundo, ubicada en Santiago de Chile, con el fin de desarrollar que se enfoca en apoyar la ejecución de proyectos pertenecientes al área de inteligencia de mercado y como base para la creación de una plataforma de generación de contenido colaborativo a través de la detección de tendencias en la web.

La herramienta a mejorar consiste en un sistema de detección de tendencias en la web, para ello, se basa en algoritmos de recuperación de la información, modelamiento de tópicos y de minado de opiniones. Así, la herramienta busca en blogs, sitios de noticias y otras fuentes de contenido, los tópicos que están siendo discutidos en la web, y de manera posterior busca qué opina la gente sobre éstos.

La herramienta en la actualidad sólo soporta un nivel bajo de flujo de datos, lo que conlleva una baja escalabilidad de la plataforma y a su vez, dificultades a la hora de implementar la herramienta como la base para una plataforma de uso masivo. Además, la recuperación de documentos es muy limitada ya que para el funcionamiento de la herramienta debe definirse una lista de sitios fuente que es fija a lo largo del tiempo a menos que el operador de la herramienta la modifique, lo que es una ardua tarea debido a los procesos que deben realizarse para la detección de fuentes. Finalmente, los algoritmos de extracción de opiniones, los que actúan sobre documentos extraídos a partir de redes sociales, no toman en consideración los emoticones lo que reduce el nivel de precisión en relación a la opinión detectada.

1.8. Estructura del trabajo de título

En el capítulo siguiente se dará a conocer el marco teórico que dará contexto a esta memoria de título y además, se realizará una revisión bibliográfica extensa de las técnicas y algoritmos que serán utilizados en el rediseño de esta plataforma.

En el capítulo 3, se realizará un análisis de la situación actual de la plataforma señalando los procesos de negocio existentes y la estructura actual del software y de la arquitectura que lo sustenta. Se dará una breve descripción de los algoritmos utilizados, el modelo de datos existente y cualquier otra información que se considere necesaria para la comprensión del estado actual del sistema.

Los requisitos que deben ser satisfechos por la plataforma rediseñada, tanto de usuario como funcionales, son dados a conocer en el capítulo 4. Es importante recordar que en este punto, que al utilizar una metodología de desarrollo ágil estos requisitos pueden cambiar a medida que el proyecto avance, por lo que sólo deben ser considerados como una guía inicial y no como un listado definitivo, sin embargo, se espera que la lista de requisitos no varíe de manera significativa.

El trabajo principal de esta memoria de título se presenta en el capítulo 5, donde se describe el rediseño propuesto, dando detalles sobre la arquitectura física que se propone para la puesta en marcha del sistema, la arquitectura de software que se utilizará en el rediseño del software, los cambios al modelo de datos sobre el cual se almacenarán los datos recuperados por la plataforma. Además, se presenta el algoritmo de calificación de fuentes y las modificaciones a realizar al algoritmo de minado de opiniones.

A continuación, en el capítulo 6 se describe la configuración experimental con los que se llevará acabo la evaluación del rediseño de la plataforma y de los modelos implementados. Además se dan a conocer los resultados de estos experimentos y se evalúa si los cambios realizados en la plataforma y los algoritmos introducidos tienen un impacto positivo en ella.

Para finalizar se dan a conocer las conclusiones de esta memoria de título, analizando los resultados de los nuevos algoritmos incorporados, las ventajas del rediseño de la plataforma, si se logró o no cada uno de los objetivos específicos y finalmente, si se comprobaron o no las hipótesis planteadas en esta.

Capítulo 2

Marco Conceptual

En este capítulo se busca describir el marco conceptual que da sustento al trabajo previamente desarrollado y a los algoritmos y modificaciones que se realizarán a la plataforma de detección de tendencias. Para ello, se dará una descripción general de cada una de las ramas del conocimiento que se involucran en el trabajo desarrollado, las cuales corresponden a minado de opiniones, modelamiento de tópicos y recuperación de conocimiento desde la web. Además, se detallarán las definiciones y algoritmos que permiten comprender todo lo discutido en este trabajo de memoria y que dan la base teórica sobre la cual se llevará a cabo el rediseño del sistema.

2.1. Recuperación de la información

Una de las ramas involucradas en el proyecto de detección de tendencias es la recuperación de la información, la cual tiene como objetivo proponer soluciones a la problemática de extraer conocimiento e información presente en uno o más documentos, colecciones de objetos o palabras, aún cuando se encuentre esta información de manera latente en los objetos que están siendo analizados [4]. En general, la recuperación de la información se enfoca en analizar documentos, los cuales se definen como una colección de datos discretos representables de manera digital, por ejemplo, un documento puede estar compuesto por letras, palabras, imágenes, etc..

Debido a que la plataforma de tendencias busca recuperar información desde sitios de noticias, blogs y redes sociales, se enfocará el marco teórico en el análisis de documentos provenientes de este tipo de fuentes.

2.1.1. Crawling

Para poder procesar documentos provenientes de la Web, y analizar colecciones de estos a través del uso de algoritmos de recuperación de información, es necesario poseer una copia local de estos. Así nace la necesidad de desarrollar algoritmos de recuperación de documentos desde la Web, los

cuales son agrupados bajo la categoría de Crawlers o Web Crawlers [24].

Dependiendo de la necesidad que se desea satisfacer a la hora de recuperar documentos existen distintos tipos de crawlers. Los tipos de crawlers más utilizados son:

- **Crawler de propósito general**: Se alimentan inicialmente de una lista de enlaces semilla y a partir de ahí van visitando todos los hipervínculos presentes en todos los documentos recuperados para visitar la mayor cantidad de sitios posibles. Es por esto que este tipo de crawlers es utilizado frecuentemente por buscadores web tales como Google Search, Yahoo! Search y Bing [15].
- **Crawler focalizado**: A diferencia de los crawlers de propósito general, estos buscan sólo recuperar documentos asociados a una temática o un conjunto de estas definidas previamente.

En general, el algoritmo utilizado por un crawler de propósito general es similar al presentado por Pant *et al.* en [48] que consiste de los siguientes pasos:

1. **Iniciar la frontera**: Se llama *frontera* al conjunto de URLs que debe visitar el crawler en cada iteración. En este paso se agregan a ésta todos los enlaces semilla.
2. **Recuperar documentos**: En el segundo paso, el crawler recupera todos los documentos presentes en cada una de las URLs de la frontera.
3. **Análisis de los documentos recuperados**: En este paso se llevan a cabo todos los algoritmos de limpieza de documentos y extracción de metadata a partir de estos. La limpieza de documentos consiste generalmente en remover stopwords y lematizar el documento (ambos procesos son descritos en la sección 2.1.2).
4. **Extracción de URLs de documentos para aumentar la frontera**: una vez recuperados y procesados los documentos, se procede a extraer a partir de estos nuevas URLs para agregar a la frontera. Dependiendo del tipo de crawler se hacen uso de distintas métricas de relevancia para determinar si una URL debe ser ingresada a la frontera o no.

2.1.2. Preprocesamiento de documentos

Para reducir la cantidad de ruido presente en los documentos y lograr que los documentos utilizados por los algoritmos de recuperación posean datos realmente significativos, es necesario preparar los documentos previamente a los realizar los procesos de minado de conocimiento.

Entre estos procesos se encuentran el de lematización (o stemming) y el de remoción de stopwords.

Lematización

Los algoritmos de lematización se enfocan en normalizar las palabras que componen un documento para que se reduzcan a su raíz semántica. Por esto, los algoritmos que realizan este proceso se dedican a remover sufijos de las palabras y reducirlas a su familia semántica.

El algoritmo de Porter [51] es ampliamente utilizado en aplicaciones de recuperación de la información y propone un acercamiento para el proceso de lematización basado en gramáticas y reglas de reemplazo.

Toda regla de gramática utilizada por el algoritmo de Porter es descrita por la ecuación 2.1:

$$(c)S_1 \rightarrow S_2 \tag{2.1}$$

Lo que denota que si la palabra en cuestión tiene el sufijo S_1, y la raíz que precede a este sufijo satisface la condición c dada, entonces el sufijo S_1 es reemplazado por S_2.

La condición utilizada en estas reglas puede estar compuesta por combinación de múltiples condiciones unidas por los operadores lógicos `AND` y `OR`. En su publicación original que se enfoca en documentos en inglés, Porter propone las siguientes condiciones:

- ***v***: La raíz contiene una vocal.
- ***S**: La última letra de la raíz es S.
- ***d**: La raíz termina en consonante doble.
- ***o**: La raíz termina la combinación consonante, vocal, consonante y la última consonante no es W, X o Y.

Además, Porter propone condiciones basadas en una métrica que describe el largo de una palabra desde el punto de vista semántico, la cual es llamada *medida*, donde si se descompone una palabra w en base a sus consonantes C y sus vocales V, la *medida* m viene dada por la expresión 2.2:

$$w = [C](VC)^m[V] \tag{2.2}$$

Remoción de stop-words

Una stop-word es una palabra w que sólo aporta ruido al documento debido a su alta frecuencia de uso o su nulo significado semántico en relación a lo que se está analizando. En general todos los conectores e ilativos son considerados como stop-words, como también números, preposiciones, etc.

El remover stop-words, al igual que la lematización, ayuda a reducir el ruido presente en los documentos y además busca reducir el sesgo estadístico hacía palabras que son utilizadas de sobremanera o que no aportan ningún valor a los algoritmos de recuperación de la información.

Un algoritmo de remoción de stop-words se puede definir en base al conjunto de stop-words $S\vec{W} = \{sw_i\}_{i \in \mathbb{N}}$, el carácter separador de palabras $\_$, la cadena vacía ϵ, el operador concatenador de palabras $\cdot$ y un documento d del cual se removerán las stop-words.

Algoritmo 2.1.1: Remoción de stop-words

Data: $S\vec{W}, d$

Result: d'

```
1 d' = ϵ;
2 for w ∈ d do
3     if w ∉ SW⃗ then
4         d' = d'·_·w
```

Se pueden plantear otros algoritmos parar remover stop-words dependiendo del lenguaje que estos sean implementados o cómo se busca en dónde están las stop-words a lo largo de un documento, sin embargo, todos funcionan bajo el mismo principio que el presentado en el algoritmo 2.1.1.

2.2. Modelos de Tópicos

Un modelo de tópicos tiene como objetivo identificar las relaciones latentes entre documentos pertenecientes a una colección con el fin de dar una descripción sucinta de esta sin perder información desde el punto de vista estadístico.

El precursor de los modelos de tópicos es David Blei, el cual en su publicación [8] describe de manera detallada los modelos de tópicos y las aplicaciones de estos y en ella se define un tópico como el conjunto de elementos que pueden representar una temática presente en una colección de documentos sin pérdida de información estadística. Por ejemplo, si existe una colección de documentos textuales que abarca múltiples temas, un tópico es un conjunto de palabras que logra describir estadísticamente uno de estos temas.

En esta memoria de título se hará uso de un modelo de tópicos sobre una colección de documentos textuales, el cual modela las relaciones latentes entre palabras, documentos y la colección propiamente tal a través de las distribuciones probabilísticas de relación entre cada una de ellas.

Entre los modelos de tópicos existente, tal como se mencionó los más utilizados son los desarrollados por Blei *et al.*. Entre ellos se encuentra el modelo LDA (Latent Dirichlet Allocation) [8], el

modelo CTM (Correlated Topic Model) [6] y el modelo dinámico DTM (Dynamic Topic Model) [7].

Todos estos modelos de tópicos se cimentan sobre las siguientes definiciones:

- Una *palabra* w es la unidad elemental de un documento textual y se define como un elemento de un vocabulario indexado V. Para efectos de estos modelos, para representar una palabra se hace uso de vectores unitarios en donde la n-ésima palabra de V se representa con un vector de largo $|V|$ en el cual sólo su componente n-ésima es igual a 1.

- Un *documento* es un arreglo de palabras descrito como $\mathbf{w} = (w_1, w_2, ..., w_N)$, donde w_n es la n-ésima palabra de este.

- Un *corpus* es una colección de documentos descrita como $D = \{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_M\}$.

- Un *tópico* es una distribución de probabilidad sobre un vocabulario V fijo. Por ejemplo, el tópico *política* está descrito por palabras como *partido, diputado, senado, ley* de manera frecuente y palabras como *guerra, marcador, gol* con probabilidad casi nula.

A continuación se da a conocer una descripción de cada uno de los modelos mencionados anteriormente, dando a conocer las diferencias entre estos y las principales características de cada uno de ellos.

2.2.1. Latent Dirichlet Allocation

El modelo llamado *Latent Dirichlet Allocation* [8] es considerado el más sencillo de los modelos de tópicos presentes hoy en día, y por ello es utilizado frecuentemente en aplicaciones que requieran el uso de modelos de tópicos para obtener información sobre colecciones de documentos de manera rápida y eficiente.

El modelo LDA trabaja bajo el supuesto de que los tópicos presentes en la colección de documentos que se está analizando no necesariamente están relacionados y por consiguiente no dependen entre ellos.

Para extraer la estructura de tópicos presente en una colección, este modelo hace uso de un modelo estadístico de generación de documentos, tópicos y palabras a lo largo del tiempo que abarque esta. El siguiente proceso se realiza para cada documento presente en una colección:

1. Definir una distribución aleatoria para la presencia de los tópicos en la colección y una distribución para la presencia de las palabras para cada tópico que se desea encontrar.

2. Luego, por cada palabra presente en el documento bajo análisis se debe:

a) Escoger un tópico aleatoriamente haciendo uso de la distribución generada en el paso 1.

b) Escoger una palabra del documento aleatoriamente a partir de la distribución del vocabulario en relación al tópico escogido.

Desde el punto de vista estadístico, es posible representar el modelo LDA como en la figura 2.1. El proceso de elección de un palabras y tópicos es representado por una caja interior cada uno y el proceso de generación de documentos desde el corpus es representado por el cuadrado exterior.

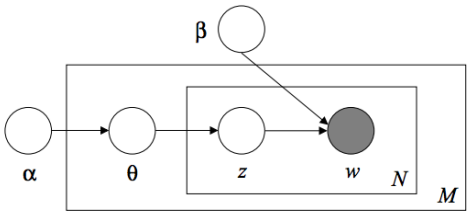

Figura 2.1: Representación gráfica del modelo LDA

Formalmente, para determinar toda la estructura de tópicos existente luego del proceso de generación, es necesario calcular las distribuciones condicionales entre los tópicos y sus documentos, el cual es un problema NP completo debido a que la cantidad de estructuras que pueden representar una colección de documentos crece exponencialmente en relación a la cantidad de documentos y palabras presente en esta. Este proceso es descrito formalmente como sigue:

1. Escoger $N \sim Poisson(\xi)$
2. Escoger $\theta \sim Dirichlet(\alpha)$
3. Para cada palabra w_n en $\mathbf{w}$

   a) Escoger un tópico $z_d \sim Multinomial(\theta)$

   b) Escoger una palabra w_d a partir de $p(w_n|z_n, \beta)$, la distribución multinomial de probabilidades condicionada sobre el tópico z_n.

Donde cada variable del proceso corresponde a:

- β es la matriz de probabilística de que el documento contenga la palabra w^j dado que discute el tópico z^i, con $B_{ij} = p(w^j = 1|z^i = 1)$.
- θ_d es la distribución de tópicos para el documento d, es decir, el conjunto de probabilidades $\theta_{d,k}$ donde esta corresponde a la probabilidad de que el documento d trate del tópico k.

- z_d son las asociaciones de tópicos para el documento d con $z_{d,n}$ es el tópico asociado a la palabra n-ésima del documento d
- w_d es el conjunto de palabras presentes en el documento d.
- $w_{d,n}$ es la palabra n-ésima del documento d.

A partir de esto, es posible definir el proceso generativo de documentos a través de la distribución conjunta de variables observables y no observables como se define en la ecuación 2.3, el cual es posible obtener a través de algoritmos de inferencia estadística como el algoritmo *Sampleo de Gibbs* [57,62], los que además de estimar la estructura de tópicos de una colección, permiten inferir la estructura de tópicos presente en otros *corpus* que estén compuestos de documentos que hablen de temas similares a los utilizados para entrenar el modelo.

$$p\left(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}\right) = \prod_{i=1}^{K} p(\beta_i) \prod_{d=1}^{D} p(\theta_d) \left(\prod_{n=1}^{N} p(z_{d,n}|\theta_d) p(w_{d,n|\beta_{1:K}, z_{d,n}}) \right) \tag{2.3}$$

2.2.2. Correlated Topic Model

El modelo Correlated Topic Model [6] se diferencia del LDA ya que hace uso de un proceso distinto para determinar la distribución de tópicos a utilizar, ya que el modelo CTM permite la existencia de relación entre las distintas componentes del modelo (y por consiguiente la covarianza entre ellas). Este nuevo grado de libertad busca permitir que el modelo de tópicos describa colecciones en donde la presencia de un elemento (sea este una palabra, documento o un tópico) puede estar relacionada con la presencia de otro de su misma categoría dentro del *corpus* en cuestión. En el diagrama 2.2 se puede observar la representación gráfica del modelo de generación estadístico que utiliza el CTM.

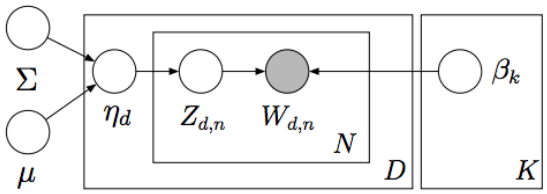

Figura 2.2: Representación gráfica del modelo CTM

2.2.3. Dynamic Topic Model

Finalmente, el modelo DTM [7] remueve una restricción que simplifica considerablemente el proceso de resolución del problema de generación estadística, esta es que cada palabra y documento

presente en el *corpus* son intercambiables en las distribuciones de probabilidad utilizadas para representar la estructura de tópicos de este último. En cualquier colección de documentos que busque demostrar cómo la información y el conocimiento evoluciona a lo largo del tiempo, como lo son los noticiarios, los blogs de tecnología, etc., el permitir que cada unidad sea intercambiable impide enlazar un tópico en un tiempo t con su equivalente en un tiempo $t' > t$ de manera inherente al modelo. Así, el modelo DTM permite modelar la evolución de tópicos a lo largo del tiempo en un corpus ordenado de manera cronológica con intervalos de tiempo discreto. La representación gráfica de este modelo se puede observar en la figura 2.3.

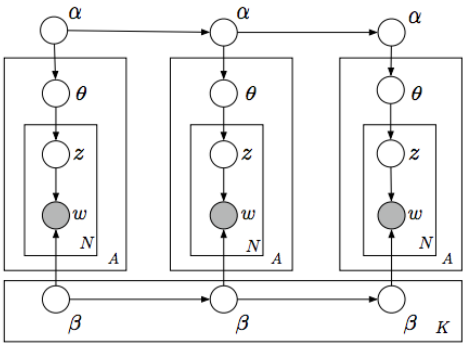

Figura 2.3: Representación gráfica del modelo DTM

2.3. Modelos de extracción de opiniones

Con el nacimiento de las redes sociales y la llegada de la Web 2.0, los usuarios comenzaron ser capaces de generar nuevo contenido en la web y también de dar a conocer sus opiniones sobre variados hechos, productos, servicios y cualquier otro tema que sea susceptible de generar un sentimiento o una opinión en un usuario.

Para aprovechar este nuevo conocimiento que está siendo generado en la web se han cosechado una serie de metodologías, algoritmos y técnicas para recuperar información desde documentos opinados que los usuarios consigan en las redes sociales. Esta nueva rama de la recuperación de la información es llamada *Web Opinion Mining*, la cual tiene como objetivo principal extraer información a partir de las opiniones que se encuentran en los documentos opinados publicados en la web [30].

Los modelos de opinión son utilizados frecuentemente en donde es necesario hacer uso de las opiniones de los usuarios para evaluar u obtener información sobre productos y servicios. En [20, 46] se menciona que los algoritmos de minado de opiniones son utilizados frecuentemente en detección de spam en review de productos, creación y mejoramiento de sistemas de recomendación de productos y servicios o de avisaje online, evaluación de nuevos productos en la web, evaluar el impacto que

tienen las reviews en las utilidades de un negocio o un producto, etc.

Para efectos de esta memoria de título, una opinión se define como una creencia subjetiva por parte de un sujeto sobre algún objeto, tema o situación en particular, que nace de una interpretación emocional por parte de éste del objeto bajo análisis o una característica de este [13]. Por consiguiente, una opinión es una creencia subjetiva de un *emisor* sobre el *receptor* o una característica de este, y posee una polaridad que señala el tipo de emoción (positiva o negativa) que da paso a la opinión propiamente tal.

Los modelos de extracción o minado de opiniones trabajan sobre *documentos opinados*, los cuales son definidos en [30] como todo documento que contenga una o más oraciones que expresan una opinión. Por lo tanto, se puede decir que los modelos de extracción de opiniones buscan determinar qué tipo de emoción motiva la emisión de una opinión en un documento [13].

Para dar a conocer un modelo de opiniones es necesario presentar una serie de definiciones que dan sustento a la gran mayoría de los modelos utilizados en la actualidad. Estas definiciones son las que siguen:

- **Objeto**: Un *objeto* o es una entidad que puede ser un producto, un servicio, un individuo, una organización, un evento, etc. descrito por la dupla (T, A) donde T es la jerarquía que describe cada una de las componentes del objeto y A es el conjunto de atributos de este. A su vez, cada componente posee su propio conjunto de sub-componentes y atributos.

- **Opinión**: Una *opinión* sobre una característica f objeto o es una evaluación emocional que realiza un *emisor* sobre este o una característica de él.

- **Emisor**: El *emisor* de una opinión es aquella persona u organización que la expresa.

- **Polaridad**: La *polaridad* de una opinión indica si la opinión es *positiva*, *negativa* u *objetiva*.

Además, en el modelo de análisis de opiniones basado en características [23], un objeto o se describe como un conjunto de características $F = f_1, f_2, ..., f_n$ donde también se incluye el objeto en cuestión como una característica particular. En este caso, cada característica f_i puede ser descrita por el conjunto de palabras o frases $W_i = w_{i1}, w_{i2}, ..., w_{im}$, donde cada w_{ij} es un sinónimo de la característica f_i; además, f_i también puede ser expresada a través del conjunto de indicadores de característica $I_i = i_{i1}, i_{i2}, ..., i_{iq}$.

Bajo este modelo, un documento que contiene opiniones d es descrito como aquel que contiene opiniones sobre un conjunto de objetos $o_1, o_2, ..., o_q$ emitidas por un conjunto de emisores $h_1, h_2, ..., h_p$. En este caso, cada opinión o_j se enfocan en un subconjunto F_j de características del objeto en cuestión y puede ser clasificada en uno de los siguientes tipos:

- **Opinión directa**: Es la quintupla $(o_j, f_{jk}, oo_{ijkl}, h_i, t_l)$, donde o_j es el objeto sobre el cuál consiste la opinión, f_{jk} es la característica del objeto o_j que está siendo analizada, oo_{ijkl} es la polaridad de la opinión sobre la característica f_{jk}, h_i es el emisor de la opinión y finalmente, t_l es el momento en el cuál h_i expresó la opinión.

- **Opinión comparativa**: Expresa la relación, sea esta de similitud o de diferencia entre dos o más objetos y las preferencias del emisor de la opinión sobre un conjunto común de características entre los objetos.

Toda opinión se basa en las emociones que guían al emisor a emitirla en el momento que este acto sucede. De acuerdo a lo expresado en [30], las emociones son ***sentimientos y pensamientos subjetivos***, y estas se dividen en 6 tipos primarios: *amor, alegría, sorpresa, rabia, tristeza* y *temor* [49].

Si bien todas las opiniones nacen de una emoción, la manera en que estas son expresadas por el emisor de ellas permite clasificarlas en dos tipos: las opiniones *explícitas*, aquellas en que el emisor expresa claramente la opinión a través de una frase subjetiva; y las *implícitas*, donde la opinión en cuestión es expresada a través del uso de una frase objetiva. Un ejemplo de opinión explícita es *"me encanta el sabor de este helado"* y de opinión implícita es *"la linterna explotó a la semana de haberla comprado"*.

2.3.1. Aplicaciones de los algoritmos de minado de opiniones

Entre las aplicaciones que tienen los algoritmos de minado de opiniones podemos encontrar:

1. **Análisis de reviews de productos**: En [3, 23, 26, 50] se discuten distintas aplicaciones de estos algoritmos en el análisis de reviews de productos, entre ellas se destacan el resumen de opiniones, detectar reviews falsos o spam y la evaluación monetaria de las características de un producto.

2. **Sistemas de recomendación**: En [14, 58, 59] se estudia mejorar sistemas de recomendación de productos a través del uso de las opiniones emitidas por usuarios de estos mismos sistemas.

3. **Inteligencia de negocios**: Es posible predecir el impacto sobre las ventas de una empresa que tendrá un producto a través de analizar las opiniones presentes en la web [32, 37].

4. **Política**: En [42, 53, 56] se muestran distintos enfoques para analizar campañas políticas y la percepción de la gente sobre leyes y candidatos políticos.

5. **Marketing online**: En [16,25] se muestra cómo medir el impacto de campañas virales y cómo mejorar un sistema de avisaje online haciendo uso de opiniones.

6. **Análisis financiero a través de opiniones**: Múltiples intentos se han realizado con el objetivo de ver la correlación entre las opiniones presentes en las redes sociales y los precios de las acciones [21,54].

2.3.2. Tipos de algoritmos de minado de opiniones

Extracción de resúmenes de opiniones basados en características

En [31] se trata el tema de algoritmos de extracción de resúmenes de opiniones, los cuales se enfocan en, a partir de un documento d que contiene múltiples opiniones sobre un objeto o, se obtiene un resumen de las opiniones que fueron expresadas para toda característica f_i mencionada en el documento. En [60] se trata el uso de este tipo de algoritmos para la detección de tendencias al agregar una dimensión temporal a la información recopilada para ver el cambio de percepción de los usuarios a lo largo del tiempo.

Detección de opiniones en documentos

Para comenzar a realizar el minado de opiniones en un documento muchas veces es útil saber de antemano si este contiene o no una opinión. Así mismo también es útil conocer de antemano qué partes del documento en cuestión son opinadas y cuales no. Sin embargo, solucionar este problema es considerado complejo [11,36], por lo que no existen soluciones robustas en esta área actualmente.

Comparación entre objetos o cualidades

Las opiniones pueden ser utilizadas para comparar objetos o cualidades de estos a través de determinar qué es lo que motiva al *emisor* de la opinión a elegir un elemento por sobre otro. En general, este tipo de modelos son utilizados para determinar el valor monetario de una característica [3] o para mejorar sistemas de recomendación de productos o servicios ya que permiten determinar, en base a las opiniones emitidas por el usuario o usuarios con perfiles similares, los avisos y recomendaciones que tienen una mayor probabilidad de ser percibidos por el usuario de manera positiva.

Detección de polaridad de opiniones en documentos

Otro tipo de algoritmo perteneciente a la rama de minado de opiniones es aquella que busca determinar la naturaleza de una opinión, o en otras palabras, qué tipo de emoción es lo que motiva al *emisor* a emitirla y además, qué polaridad posee esta (negativa o positiva). Generalmente este tipo de modelos buscan conocer el nivel de connotación negativa, positiva y objetiva tiene una

opinión o simplemente cual es la polaridad predominante en la opinión. Comúnmente estos algoritmos son conocidos como de *clasificación de subjetividad* y la clasificación de cada una de las oraciones presentes en el documento como algoritmos de *clasificación de sentimientos*.

Una aplicación frecuente de este tipo de algoritmos es mejorar la información encontrada en sistemas de *Business Intelligence* [18] a través del cruce entre la información contable y la apreciación que tienen los clientes y usuarios de la empresa y que es expresada en la web. Por otro lado, también son utilizados para saber la apreciación sobre un nuevo producto que ha sido lanzado al mercado, sobre una compañia que cotiza en la bolsa o cualquier objeto o tema que puede ser discutido por parte de los usuarios en la web.

2.3.3. Algoritmos para extracción de polaridad de opiniones

En la plataforma de detección de tendencias se hace uso de algoritmos de detección de polaridad para determinar qué es lo que se opina en la web sobre los tópicos que son extraídos desde los documentos recuperados, por lo que a continuación se dará a conocer las dos afluentes más utilizadas de algoritmos de detección de polaridad en opiniones.

Algoritmos de clasificación a través de aprendizaje supervisado

La mayoría de los algoritmos de aprendizaje supervisado existentes (Naive-Bayes, Supporting Vector machines, etc.) pueden ser aplicados a la clasificación de polaridad de documentos tal como se muestra en [40, 47].

El algoritmo más utilizado debido a su simplicidad es un clasificador Naive-Bayes, el cual buscar obtener las probabilidades de que un documento d posea la polaridad p $\Pr(p \mid d)$. Este tipo de clasificador obtiene estas probabilidades al resolver el siguiente problema de maximización: $arg\,\max_{p \in P}\{\Pr(p \mid d)\}$.

Los clasificadores de Naive-Bayes hacen uso de la regla de Bayes para poder simplificar el problema de maximización que deben resolver:

$$p_d = arg\,\max_{p \in P}\left\{\frac{\Pr(d \mid p) \cdot \Pr(p)}{\Pr(d)}\right\} \tag{2.4}$$

Debido a que sólo se busca conocer la probabilidad de que un documento tenga una polaridad y no obtener un puntaje específico para el nivel de polaridad que posee, el denominador de la ecuación 2.4 puede ser eliminado. Esto junto con el hecho de que uno de los supuestos del clasificador de Naive-Bayes es la existencia de independencia condicional entre todas las polaridades, se puede decir que:

$$\Pr(d \mid p) = \prod_{i=1}^{m} \Pr(w_i \mid p) = \prod_{i=1}^{m} \frac{\#(w_i, p)}{\#(w_i)} \tag{2.5}$$

Con $\#(w_i, p)$ el número de veces que la palabra w_i se ha encontrado en documentos de polaridad p en el conjunto de entrenamiento y $\#(w_i)$ el número de veces que la palabra w_i aparece en este último. Para evitar que existan probabilidades 0, se realiza un proceso llamado "suavización de Laplace"que consiste en lo siguiente:

$$\Pr(d \mid p) = \prod_{i=1}^{m} \frac{\#(w_i, p) + 1}{\#(w_i) + m} \tag{2.6}$$

Con estas ecuaciones basta resolver el problema de maximización planteado para obtener las probabilidades de que cada documento posea una polaridad en particular.

En general, las características utilizadas por los algoritmos de aprendizaje supervisado se dividen en las siguientes categorías:

- *Frecuencia y presencia de términos*: Si bien el uso de frecuencia de aparición de términos, por ejemplo a través del modelo *tf-idf*, en la recuperación de la información siempre ha sido de mucha utilidad, en [47] se muestra que en el caso de la extracción de opiniones desde documentos la *presencia* de un término es más importante que la frecuencia con que este aparece.

- *Partes del discurso*: Los adjetivos han sido utilizados con frecuencia [40] en el uso de algoritmos de aprendizaje supervisado, y tal como se muestra en [22] existe una alta correlación entre la presencia de adjetivos en una oración y la subjetividad de esta.

- *Sintáxis*: Múltiples investigaciones [35,41] han tratado de utilizar la relación entre las palabras como características en algoritmos de aprendizaje supervisado.

Algoritmos de clasificación a través de aprendizaje no supervisado

En [61] se propone un algoritmo de aprendizaje no supervisado para la clasificación de polaridad de documentos que se compone de tres etapas:

1. Se extraen todas las frases con verbos o adjetivos, ya que tal como se menciona en [22,40], estas partes del discurso se han mostrado muy útiles a la hora de detectar opiniones en documentos. Sin embargo, a pesar de que un adjetivo por si solo puede demostrar subjetividad, puede que no exista la información suficiente para determinar la polaridad de la opinión y por consiguiente este algoritmo trabaja con pares de palabras, una de ellas siendo un adjetivo y la otra una

palabra contextual que facilitar la determinación de la polaridad de la oración en cuestión. Estos pares de palabras son extraídos si, considerando las dos palabras y la que les sigue, corresponden a alguno de los patrones presentados en la tabla 2.1.

Primera Palabra	Segunda Palabra	Tercera palabra
Adjetivo	Sustantivo Plural o Singular	Palabra
Adverbio	Adjetivo	No Sustantivo Plural ni Singular
Adjetivo	Adjetivo	No Sustantivo Plural ni Singular
Adjetivo	Adjetivo	No Sustantivo Plural ni Singular
Sustantivo Plural o Singular	Adjetivo	No Sustantivo Plural ni Singular
Adverbio	Verbo	Palabra

Cuadro 2.1: Patrones de partes del discurso

2. Se estima la polaridad de las frases extraídas en base a los patrones de la tabla 2.1 haciendo uso de la métrica de dependencia estadística entre términos llamada *pointwise mutual information* (PMI) que se presenta en la ecuación 2.7

$$PMI(w_1, w_2) = log_2 \left(\frac{\Pr(w_1 \wedge w_2)}{\Pr(w_1) \Pr(w_2)} \right) \tag{2.7}$$

Luego, la polaridad de una frase puede ser calculada basándose en el nivel de asociación entre ella y las palabras de referencia *pobre* y *excelente* a través de la ecuación 2.8

$$oo(frase) = PMI(frase, "excelente") - PMI(frase, "pobre") \tag{2.8}$$

3. Finalmente, el algoritmo calcula la polaridad *oo* promedio de todas las frases en el documento y lo clasifica dependiendo de si el promedio es positivo o negativo.

Algoritmos basados en lexicones de opinión

Los algoritmos basados en lexicones de opinión son los algoritmos más sencillos y a su vez los que buscan ser de uso más general debido a que la información utilizada para determinar la polaridad de una opinión no está restringida a ningún dominio en particular. Un lexicón es un conjunto de palabras rotuladas con polaridad de sentimientos, es decir, cada palabra perteneciente al lexicón tiene asociado un puntaje de polaridad.

Estos algoritmos trabajan bajo la hipótesis de que una palabra es considerada la unidad elemental de una opinión y por lo tanto la polaridad de una opinión puede reconstruirse a partir de la polaridad de cada una de las palabras que la componen. Ejemplos de algoritmos que hacen uso de lexicones para determinar la polaridad de una opinión se pueden encontrar en [9, 27, 43, 47]. En relación al

minado de opiniones desde documentos de microblogging, Kouloumpis *et al.* dan a conocer en [28] que los algoritmos de basados en lexicones pueden dar buenos resultados.

El lexicón utilizado por la plataforma de detección de tendencias es *SentiWordNet* [17] el cual está disponible públicamente para ser usado en este tipo de aplicaciones de minado de opiniones. En este trabajo de memoria se trabajará con la versión 3.0 de *SentiWordNet*.

Cada palabra presente en un lexicón tiene asociado un puntaje por cada polaridad positiva, negativa y objetiva que representan el aporte de esta palabra para la polaridad de una opinión. En el caso de *SentiWordNet* [43] se tiene que cada palabra tiene asociado sólo los puntajes de polaridad positiva y negativa, el puntaje de objetividad viene de siguiente la ecuación:

$$w^p + w^n + w^o = 1 \tag{2.9}$$

Donde w^p es el puntaje de polaridad positiva de la palabra w, w^n el puntaje de polaridad negativa y w^o el puntaje de objetividad de ésta.

Los algoritmos basados en lexicones de opinión hacen uso de las siguientes metodologías para reconstruir la polaridad de la opinión contenida en un documento a partir de sus palabras:

- **Conteo de palabras**: los puntajes de polaridad de un documento se obtiene a través de la fracción de palabras cuya que posee una polaridad predominante p. En este caso, una palabra será considerada de una polaridad p si su mayor puntaje es el de aquella polaridad.
- **Promedio de palabras**: En un algoritmo de promedio de palabras, el puntaje asociado a una polaridad p es el promedio de los valores de polaridad p de todas las palabras presentes en el documento.

A partir de estas metodologías básicas se pueden realizar diversas variaciones tales como las presentadas en [45], donde se sugiere: modificar los puntajes de cada palabra en base al conjunto de palabras que la rodean en el documento, hacer uso de las negaciones y la capitalización, y finalmente incorporar al puntaje la existencia de intensificadores y disminuidores de adjetivos.

2.4. Diseño de software

A continuación se darán a conocer las definiciones necesarias para poder comprender un rediseño de software, dando a conocer lo que se conoce como arquitectura física, arquitectura de software y escalabilidad de aplicaciones.

2.4.1. Arquitecturas Físicas

La arquitectura física de un software corresponde a la topología física de este, es decir, la manera en que las diferentes componentes de hardware del sistema están relacionadas; y además, cómo el software está distribuido a lo largo de cada una de estas componentes, sean estas múltiples máquinas, servidores, data centers, etc.

Por ejemplo, la arquitectura física de una aplicación web está compuesta por el servidor web, el servidor de aplicaciones y el servidor de bases de datos. Además, se debe explicar qué parte de la aplicación corresponde a cada nodo, continuando con el ejemplo, en el servidor web corre el software encargado de manejar las peticiones web, por ejemplo *Apache*; en el servidor de aplicaciones se encuentra toda la lógica del sistema, por ejemplo en el caso de una aplicación en *Tomcat* esta se encontraría en el servidor de aplicaciones; y finalmente, en el servidor de bases de datos se encuentra el motor de base de datos que se utiliza en la aplicación.

En general, se hace uso diagramaciones de arquitecturas físicas cuando se desea disponer de una guía de como debe ser puesto en marcha el software, considerando cuál es la manera óptima de desplegar la aplicación a lo largo de los recursos disponibles para sacar provecho de estos.

2.4.2. Arquitecturas de Software

A diferencia de la arquitectura física de una aplicación, la arquitectura de software se enfoca en detallar las distintas componentes de la aplicación que está siendo descrita [19], detallando sus componentes de software y como estas se comunican entre sí, junto con dar estructura a como deberían ser desarrollados los módulos la plataforma propiamente tal. Se puede pensar en una arquitectura de software como aquel conjunto de reglas y patrones que permiten razonar sobre el sistema en su totalidad de manera sencilla, y que a su vez permiten reducir la cantidad de anti-patrones utilizados con el fin de lograr el desarrollo de un software de calidad.

Entre los pasos iniciales del desarrollo de una plataforma o una aplicación siempre se encuentra el diseñar la arquitectura de software, la que permite reducir la complejidad del sistema a través de la abstracción de funcionalidades y la separación de responsabilidades entre las distintas componentes que se desean incluir en este. Además, el diseño previo de ésta permite incluir en el desarrollo patrones de diseño de software, mejores prácticas en base a la aplicación que se quiera desarrollar y la tecnología que se utilizará y finalmente considerar todas las necesidades del software, tales como escalabilidad, tolerancia a los fallos, mantenibilidad, etc.

Hay múltiples patrones de diseño para arquitectura de software, a continuación se dan a conocer los más utilizados en el desarrollo de este tipo de aplicaciones:

- **Cliente - Servidor**: En este tipo de patrón de diseño se busca desarrollar una aplicación que distribuye el trabajo realizado entre el proveedor del recurso que se desea utilizar (*servidor*) y el consumidor de éste (*cliente*). En cuanto a la ubicación de ambos desde el punto de vista físico no hay limitantes, por lo que ambos pueden vivir en el mismo nodo o estar distribuidos a lo largo de una red. Ejemplos de aplicaciones que hacen uso de este patrón son las aplicaciones web y los protocolos `ftp` y `http` entre otras. Este patrón es descrito en la figura 2.4.

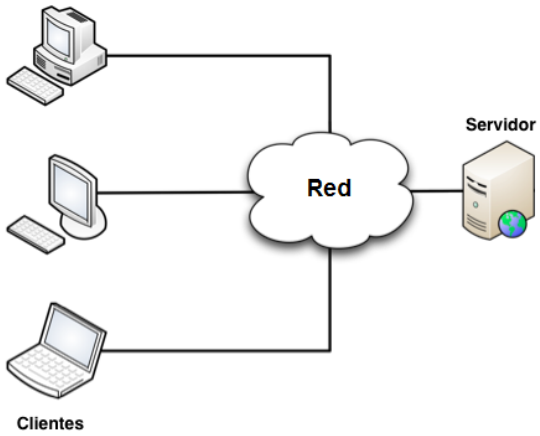

Figura 2.4: Patrón Cliente - Servidor

- **Peer - to - Peer**: A diferencia de la arquitectura cliente - servidor, la arquitectura P2P (figura 2.5) considera que todos los nodos de la red pueden actuar tanto como servidor y como cliente. El objetivo detrás de esta arquitectura es permitir que cada uno de los nodos que componen el sistema tengan acceso a todos los recursos existentes en la red sin la necesidad de tener un servidor central que disponga de estos. Este tipo de arquitectura es utilizada frecuentemente en redes para compartir archivos (Napster, eDonkey, BitTorrent), streaming de video distribuido, redes de intercambio monetario como BitCoin o redes de comunicación como Skype.

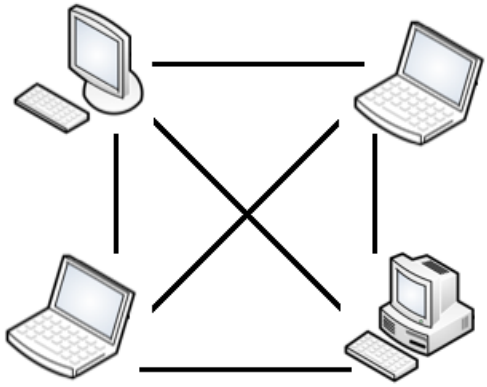

Figura 2.5: Patrón Peer - to - Peer

- **Basada en Eventos**: Esta arquitectura se basa en la noción de que un evento corresponde a un cambio de estado de algún elemento del sistema, y que provoca una reacción por parte de otras componentes de este. Por lo tanto, se puede describir una arquitectura basada en eventos en base a la producción, detección y reacción de eventos. Son utilizados frecuentemente en aplicaciones asíncronas o en las cuales sus componentes están débilmente ligadas. Es común hacer uso de este patrón junto a una arquitectura orientada a servicios debido a que comparten múltiples nociones y la detección de eventos puede permitir disminuir el tiempo de respuesta de una aplicación que haga uso de servicios. Bajo este tipo de arquitectura son diseñados todos los sistemas de notificación de las plataformas móviles actuales junto con los sistemas de mensajería `pubsub`.

- **Aplicación Monolítica**: Una aplicación diseñada haciendo uso de este patrón debe ser capaz de realizar todas las tareas pertinentes al problema que busca solucionar. En general, se refiere a una aplicación monolítica si esta no está diseñada modularmente o si no está diseñada para que cada una de sus partes pueda actuar de manera independiente (aún cuando la aplicación sea desarrollada de manera modular siguiendo principios de orientación a objetos). Ejemplos de este tipo de arquitectura son la mayoría de las aplicaciones de escritorio que no tienen algún tipo de conexión con internet, como por ejemplo los procesadores de texto, las calculadoras, los reproductores de música, etc.

- **Orientada a Servicios**: La arquitectura orientada a servicios o SOA por sus siglas en inglés[1] busca desarrollar aplicaciones basadas en un conjunto de servicios que interactúan entre si. Cada servicio debe basarse en una unidad de negocio bien definida las cuales pueden ser utilizadas posteriormente en otro tipo de aplicaciones. En general SOA hace referencia a una serie de principios que definen cómo distintas aplicaciones web o servicios presentes en esta deben comunicarse e interactuar a lo largo de múltiples plataformas.

2.4.3. Escalabilidad

La escalabilidad es la cualidad de un algoritmo, protocol, software, o cualquier otro objeto capaz de funcionar de manera eficiente y práctica a medida que la cantidad de elementos con los que trabaja aumenta considerablemente. En otras palabras, la escalabilidad es la habilidad de un sistema de crecer en capacidad de procesamiento de manera similar a como crece la demanda por este.

Es importante considerar que la escalabilidad no sólo considera la cantidad de usuarios que hacen uso del sistema, también considera todas las variables externas que afectan a este como por ejemplo

[1] SOA: Service Oriented Architecture

fuentes externas de información, objetos que deba indexar, operaciones de procesamiento de datos que involucren todas las componentes del sistema, etc.

Cuando se habla de escalabilidad en el ámbito del diseño de software, se hace referencia principalmente a la habilidad de un sistema distribuido para aumentar o disminuir sus recursos a medida que la demanda cambie. Existen dos tipos de escalabilidad en este ámbito y ambos son descritos a continuación.

- **Escalabilidad horizontal**: la escalabilidad horizontal hace referencia a la capacidad del sistema de crecer al agregar nuevos nodos a la plataforma, sean estos en algún componente del sistema (como en el caso de los sistemas diseñados bajo SOA) o a la plataforma en su totalidad.
- **Escalabilidad vertical**: que un sistema posea escalabilidad vertical significa que su rendimiento crece a medida que se agregan recursos a los nodos que el sistema posee, como por ejemplo agregando procesadores, memoria RAM, etc.

La ley de Amdahl [2] da una cota superior a la mejora de rendimiento en programas paralelizables. Si se define P como la fracción del software que es paralelizable, y N la cantidad de procesadores en uso se tiene que la mejora en rendimiento $S(N)$ viene dada por la ecuación 2.10:

$$S(N) = \frac{1}{(1-P) + \frac{P}{N}} \tag{2.10}$$

2.5. Soluciones existentes para detección de tendencias

En el ámbito académico, múltiples investigaciones [1,12,34] han abordado la detección de tendencias la web, principalmente en las redes sociales, destacándose entre ellas dos tipos distintos, aquellas que tienen como objetivo detectar de manera temprana aquellos tópicos que serán tendencia en el corto plazo, y las que buscan detectar aquellos tópicos que están siendo tendencia y su presencia va en aumento a lo largo del tieimpo.

En aplicaciones comerciales, la plataforma web *NewsWhip*[2] ofrece prestaciones similares a las presentes en la plataforma de detección de tendencias que posee Duam S.A., sin embargo, su enfoque es lograr ser un agregador de noticias con características sociales, como la medición de menciones en las redes sociales de una noticia en particular o el análisis de noticias de una empresa en particular en la web. Además, *NewsWhip* ofrece la herramienta *Spike*, que permite a los generadores de contenido analizar cómo sus noticias se esparcen por la web.

[2] `http://www.newswhip.com/`

La empresa *Sysomos*[3] se enfoca en monitorear las redes sociales en búsqueda de información relevante para una empresa en particular, sin embargo, no hacen uso de la información presente en las noticias y no tienen como objetivo hacer un análisis extenso de las tendencias en la web, si no monitorear las conversaciones que se están realizando en las redes sociales.

Otra iniciativa que busca detectar tendencias en la Web es Google Trends, la cual toma un enfoque distinto a los ya mencionados al analizar el comportamiento de búsqueda de los usuarios de su motor de búsqueda, sin embargo, no hacen uso de los datos presentes en su red social Google+ para complementar las tendencias obtenidas con información sobre las opiniones de la gente sobre ellas.

A pesar de buscar herramientas de código abierto o propietario que permitieran el tipo de análisis deseado por Duam S.A., no fue posible encontrar alternativas disponibles, principalmente debido a que toda empresa que ofrece servicios similares poseen patentes sobre su tecnología y no tienen a disposición su plataforma para que sea utilizada por terceros para ofrecer servicios similares.

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

Capítulo 3

Situación actual de la plataforma

Tal como se mencionó anteriormente, el foco principal de este trabajo de título es el rediseño y mejoramiento de una plataforma de software ya existente que busca ser capaz de servir como una herramienta de detección de tendencias en la Web, a través de la aplicación de modelos conocidos de recuperación de la información y gestión del conocimiento sobre tópicos presentes en la Web y la opinión que los usuarios consignas sobre éstos en las redes sociales.

De acuerdo a lo planteado en el capítulo 2, el primer paso de todo rediseño de una plataforma o de software es analizar la situación actual de ésta. En el caso de un software, la situación actual está compuesta por los requisitos que satisface y por cómo está implementada la solución, tanto la arquitectura sobre la cual esta se encuentra como el diseño de software en el cual está basada.

En este capítulo se dará a conocer la situación actual del sistema que se desea mejorar, la cual se detalla a continuación de la misma manera en que se describirá la solución propuesta, presentando primero la arquitectura física sobre la que se encuentra el software, posteriormente el diseño arquitectural de la plataforma y una descripción detallada de las funcionalidades de cada módulo existente para finalizar con el modelo de datos utilizado.

3.1. Arquitectura física de la plataforma

La aplicación actualmente sólo presenta un esbozo de lo que puede ser considerado como una arquitectura física de 3 capas dado que no existe una división clara entre estas, ni tampoco considera la separación entre cada uno de los módulos que componen el sistema de detección de tendencias.

Así, la arquitectura física actual puede ser descrita por la figura 3.1. En ella sólo hay dos capas: la primera está encargada de todo el procesamiento de datos y despliegue de estos (considerada una tercera capa virtual), y la segunda, la capa de persistencia, en donde son almacenados los datos recuperados y procesados.

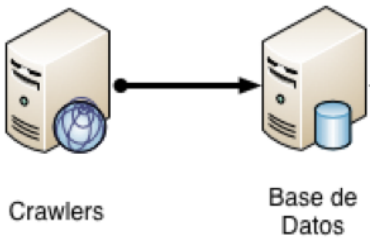

Figura 3.1: Arquitectura física actual para el procesamiento de datos

3.2. Arquitectura Lógica

La descripción del diseño arquitectural de la plataforma se realizará describiendo la arquitectura lógica de cada una de las capas mencionadas en la sección anterior, dando a conocer los módulos que componen cada capa y como estos interactúan entre si.

3.2.1. Capa de procesamieto

En la capa de procesamiento, se encuentran dos aplicaciones monolíticas en un mismo servidor que trabajan en conjunto para obtener toda la información que se desea a partir de los datos recopilados por esta.

Software de detección de Tendencias

El software de detección de tendencias se compone de dos aplicaciones que son ejecutadas periódicamente y que están encargadas de leer y escribir información hacia la capa de persistencia.

Esta herramienta trabaja bajo dos conceptos fundamentales, uno de ellos son los llamados "*periodos de trabajo*", que son intervalos de tiempo predeterminados en los cuales se dedica sólo a recuperar, desde las fuentes escogidas, documentos del periodo actual y opiniones correspondientes al periodo anterior; y las iteraciones de trabajo, las cuales se suceden múltiples veces por periodo y es donde se realizan las tareas de recuperación de tópicos a partir de los documentos extraídos, recuperación de opiniones desde documentos opinados y clasificación de estas últimas con sus respectivos tópicos.

El primero de los módulos que conviven bajo el software de detección de tendencias consiste en un crawler de documentos y documentos opinados el cual, en cada iteración escoge una fuente perteneciente a una lista de fuentes fijada previamente y descarga uno por uno los nuevos documentos que se encontraron en recurso especificado y luego, para un tópico del periodo anterior que aún no ha sido procesado, se buscan documentos en las redes sociales que contengan las palabras claves que describen el tópico en cuestión.

Además, el módulo de crawling se encarga de procesar los documentos opinados para asignarles

un puntaje de opinión tal como se discutió en el capítulo 2, es decir se clasifica cada uno de ellos con el fin de detectar si es una opinión positiva, negativa o neutral, y qué tan fuerte es el sentimiento en el que se basa la opinión.

La segunda de estas componentes es ejecutada al final de cada periodo de trabajo, y es la encargada de crear una nueva estructura de tópicos con los documentos recuperados durante el periodo en cuestión para que sea utilizada por el módulo anterior en el periodo siguiente.

En la figura 3.2 se observa un diagrama de las componentes existentes en este módulo de la capa de procesamiento, donde se deja en claro que cada una de ellas apunta a la capa de persistencia que fue mencionada en la sección anterior.

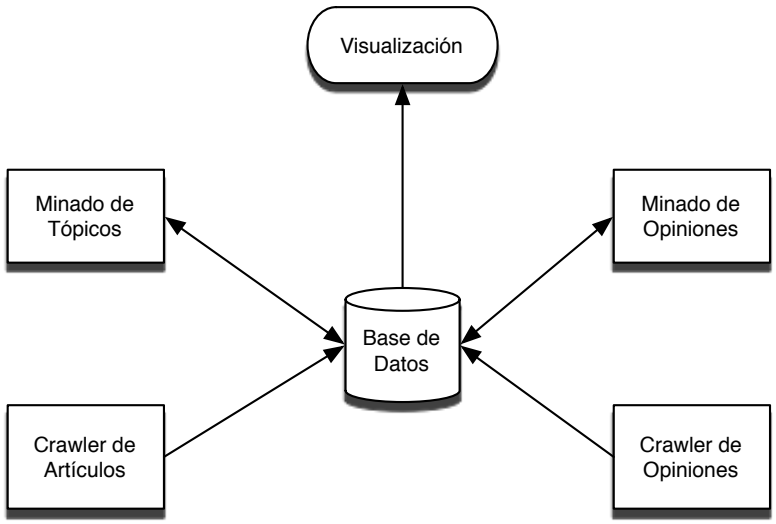

Figura 3.2: Arquitectura física actual para el procesamiento de datos

Aplicación de presentación de datos

Esta aplicación se encarga de presentar gráficamente la información obtenida por el software de detección de tendencias. Consiste de una aplicación escrita en PHP encargada de leer la información desde la base de datos y desplegarla gráficamente. No posee una API ni ninguna manera de extraer la información presentada en el sitio para permitir el uso de esta plataforma como base para la creación de aplicaciones orientadas a usuarios específicos.

Como se puede apreciar de la descripción realizada, la cohesión entre cada uno de las componentes de la plataforma es alta, el flujo de trabajo en algunas de estas sólo permite trabajar de manera secuencial y monolítica, y finalmente, no considera la creación posterior de nuevas aplicaciones con la información recuperada. Además, no se consideró una arquitectura de software orientada a servicios a la hora de implementar cada una de las aplicaciones, la cual permitiría desacoplar

muchas componentes del sistema otorgándole a este la flexibilidad necesaria para que sea un servicio escalable.

3.3. Diseño de software

Para continuar con el análisis de la situación actual, una vez que ya ha sido descrita la arquitectura lógica de la plataforma, es necesario dar a conocer cómo funciona cada componente perteneciente a un módulo o una capa del software, por lo que se dará una breve descripción de los algoritmos en uso junto con las interacciones entre cada uno de los módulos que trabajan en conjunto en el sistema.

3.3.1. Módulo de recuperación de documentos

Para describir la solución que se utilizará para la recuperación de documentos, primero es necesario dar a conocer las definiciones con las que se trabajará en ella para posteriormente describir el algoritmo de recuperación de documentos.

Se considera que una fuente de documentos presente en la web es un *feed* si cada elemento que esta contenga es desplegado de manera cronológica y pertenecen todos una misma temática. Si una fuente de documentos dispone de un punto de acceso donde se puedan recuperar cada uno de los documentos existentes en ella se dice que es un *feed sindicable*, un ejemplo de esto son todos aquellos sitios web que tienen la opción de suscribirse a su contenido a través de RSS.

Una limitante a considerar a la hora de trabajar con feeds sindicables es que el conjunto de documentos presentes cuando se accede a esta depende del momento en que se realice esta acción, por lo que el conjunto de documentos $\{d_i^F\}_{i\in\mathbb{N}}$ que se obtienen al solicitar todos los documentos desde la fuente F se ve limitada por el momento t en el cual se realice esta petición, por lo tanto, se define $\{d_i^{F^t}\}_{i\in\mathbb{N}}$ como el conjunto de documentos recuperados desde una fuente F en un instante de tiempo t. Además, se define $\{F_i\}_{i\in\mathbb{N}}$ como el conjunto de *feeds* que recorrerá el módulo de recuperación de documentos a través de su *crawler* para alimentar el módulo de extracción de tópicos.

Para este proyecto, sólo se trabajará con feeds sindicables, por lo que, en base a lo anterior, es posible definir un algoritmo de recuperación de documentos a partir de una lista de fuentes $\{F_i\}_{i\in\mathbb{N}}$ sindicables (sean estas *RSS* o *Atom*) como se describe en el algoritmo 3.3.1:

Algoritmo 3.3.1: Recuperación de documentos

Data: $\{F_i\}_{i\in\mathbb{N}}$, t

Result: $\bigcup_i \{d_j^{F_i^t}\}_{j\in\mathbb{N}}$

```
for i ← 1 to ‖{F_i}_{i∈ℕ}‖ do
    XML ← retrieveXML(F_i);
    documents := [];
    for d in parseXML(XML) do
        documents ← documents ⋃ d;
return documents;
```

En donde cada documento d recuperado por el crawler contiene la siguiente información:

- Feed F desde donde el crawler recuperó el documento.
- Tiempo t cuando el documento fue creado en la feed F.
- Contenido c que corresponde a todo el contenido textual del documento.
- URL h en donde se encuentra el documento publicado en el servidor desde donde se proceso la feed F.
- Un conjunto M con todos aquellos datos relevantes al documento que no son considerados primordiales a la hora de almacenar un documento en la base de datos. Este conjunto es llamado conjunto de metadatos, y ejemplo de datos que se encuentran en ellos son la fuente desde donde el feed F extrajo la información o las categorías a las cuales el documento d está asociado en el feed F.

Para que la plataforma tenga los documentos necesarios para la detección de tendencias, es necesario ejecutar el algoritmo 3.3.1 de manera periódica, para así capturar la totalidad del universo de documentos presente en todas las fuentes. Debido a su naturaleza secuencial este algoritmo posee variadas limitaciones para la recuperación de documentos, ya que si el conjunto de feeds a recuperar es muy grande o cada feed posee una cantidad considerable de documentos nuevos cada vez que el algoritmo recorre un ciclo completo, es posible que documentos nuevos no alcancen a ser observados por este módulo provocando así problemas de pérdida de información.

3.3.2. Módulo de extracción de tópicos

Este módulo se encarga de recuperar los tópicos tratados en los documentos recuperados por el módulo previamente descrito. La extracción de tópicos a partir de una serie de documentos ordenados cronológicamente se realiza de la siguiente manera:

El modelo LDA permite, dada una colección de documentos $\{d_i\}_{i=1...\mathbb{N}}$, obtener un conjunto de tópicos t asociados a documentos, los cuales están descrito por la probabilidad $P(topic = t|document = d)$ de que un documento d pertenezca al tópico t y además, para cada tupla (w, t) la probabilidad $P(topic = t|word = w)$ de que una palabra w describa al tópico t. Así, es posible obtener los tópicos que se tratan a lo largo del tiempo en los feeds que se están minando y las palabras que los describen para luego utilizar esta información con el fin de recuperar documentos opinados desde las redes sociales.

Para cada periodo t_i, se toman todos los documentos de los dos periodos anteriores t_{i-1}, t_{i-2} y se entrena un nuevo modelo LDA con estos. Luego, para los documentos del periodo t se realiza inferencia con el modelo LDA sobre estos para descubrir el modelo de tópicos subyacente en estos.

Una vez que se tengan los documentos de los periodos t_i, t_{i-1}, t_{i-2}, es posible enlazar dos tópicos T y T', con vectores de probabilidades de palabras $\vec{w}_T$ y $\vec{w}_{T'}$ través de una función de distancia de tópicos que se define como sigue:

$$d(T, T') = \sum_{w \in \vec{w}_T} \sum_{\vec{w}_{T'}} w_i - w_j \tag{3.1}$$

Y luego, dado toda dupla T y T' de tópicos, se enlazan sí y sólo si el resultado la función $d(T, T')$ está bajo un umbral ϕ que se define a la hora de comenzar el análisis.

3.3.3. Módulo de extracción de opiniones desde la Web

Para la extracción de opiniones desde documentos opinados recuperados desde redes sociales, se utilizará un algoritmo basado en lexicones, recurso de información rotulada que asocia palabras con polaridad de sentimientos, en otras palabras, a cada palabra le asocia un valor en el continuo de polaridad de opinión. El uso de los lexicones en algunos modelos de opinión mining se basan en la hipótesis de que una palabra puede ser considerada como una unidad fundamental de información sobre opinión, y por lo tanto puede dar indicios sobre la polaridad de un documento en su totalidad. Algoritmos de *opinion mining* que se basan en el uso de lexicones pueden ser encontrados en [9, 27, 43, 47], los cuales de acuerdo a lo mostrado por Kouloumpis *et al.* [28] pueden dar buenos resultados

en el contexto de minado de opiniones desde documentos de *microblogging*.

Cada palabra existente en el lexicón tiene asociado un puntaje. En el caso de esta plataforma se utilizará *SentiWordNet*[1] [43], en el cual se tiene que cada palabra tiene asociado los siguientes puntajes:

$$\vec{w} = \langle w, w^p, w^o \rangle \tag{3.2}$$

Con $\vec{w}$ el vector rotulado de la palabra w, w^p el puntaje positivo de la palabra, w^n el puntaje negativo y w^o el puntaje neutral de esta. Además, cada palabra rotulada presente en *SentiWordNet* posee la siguiente restricción sobre sus componentes:

$$w^p + w^n + w^o = 1 \tag{3.3}$$

Así, dado el vector $\vec{w}_d$ de largo k de palabras presentes en el documento opinado d, es posible asociar a cada documento opinado tales puntajes de la siguiente forma:

$$d^p = \frac{\sum_{i=1}^{k} w^p}{\|\vec{w}_d\|}, d^n = \frac{\sum_{i=1}^{k} w^n}{\|\vec{w}_d\|}, d^o = \frac{\sum_{i=1}^{k} w^o}{\|\vec{w}_d\|} \tag{3.4}$$

Luego, considerando la función `polaridad(documento)` que dado un documento opinado d retorna el vector (d^p, d^n, d^o) y un conjunto de $\{d_i\}_{i=1...\mathbb{N}}$ de documentos opinados, se realiza el siguiente procedimiento para asociar un puntaje de opinión a cada uno de ellos:

Algoritmo 3.3.2: Clasificación de documentos opinados

Data: $\{d_i\}_{i=1...\mathbb{N}}$

Result: $\{\vec{d_i}\}_{i\in\mathbb{N}}$

```
documents := [];
for i ← 1 to ‖{d_i}_{i=1...N}‖ do
    d_i ← polaridad(d_i);
    documents.append(d_i);
return documents;
```

Para definir qué documentos serán recuperados desde redes sociales, se tiene el algoritmo 3.3.3, que dado un tópico T obtiene todos los n-gramas de largo n que caracterizan a ese tópico en particular en el periodo t. El parámetro N consiste en la cantidad de palabras que deben ser utilizadas para

[1] SentiWordNet es el lexicón más utilizado en la actualidad en aplicaciones de web opinion mining.

la obtención de los unigramas que describen al tópico, además, el método $T.\texttt{words}(t, N)$ obtiene las N palabras más relevantes del tópico T en el periodo t.

Algoritmo 3.3.3: Método `generateQueries`

Data: T, t, n, N

Result: $\{query_i\}_{i \in \mathbb{N}}$

```
1 queries := [];
2 words = T.words(t, N);
3 forall p ∈ permutaciones(words, n) do
4     queries.append(p);
5 return queries;
```

Luego, para cada tópico T, se obtienen todas las queries que le correspondan, y se realizan búsquedas en las redes sociales que se determinen utilizando sus APIs. En el caso particular de este proyecto, sólo se trabajará con la red social de microblogging Twitter.

3.3.4. Módulo de detección de tendencias

El módulo de detección de tendencias es el núcleo de la plataforma, la cual consiste de una herramienta de visualización de tópicos a lo largo del tiempo. Para ello, fue desarrollada una aplicación web en PHP que se encarga de leer los datos y desplegarlos en un gráfico por tópico (figura 3.3). Tal como se mencionó en la sección de arquitectura física, este módulo no expone ningún tipo de API para que usuarios externos utilicen la aplicación.

Actualmente este módulo sólo consiste de scripts que presentan la información pero no tiene ningún tipo de estructura que facilite la inclusión de nuevas funcionalidades o la creación de una API que haga uso de la información para hacerla accesible por terceros. Además, no existe una separación entre las capas de presentación, de lógica de negocios y de acceso a la capa de persistencia, por lo que reutilizar lo desarrollado con este fin es poco viable.

3.4. Modelo de datos

En esta sección se presenta el modelo de datos que se utiliza para almacenar los patrones recopilados a lo largo de todo el proceso de detección de tendencias. Para facilitar una mejor lectura de los modelos, estos serán presentados de manera separada dependiendo de la etapa del proceso que correspondan, y señalando, en caso de ser necesario, los vínculos existentes entre los modelos pertenecientes a cada fase.

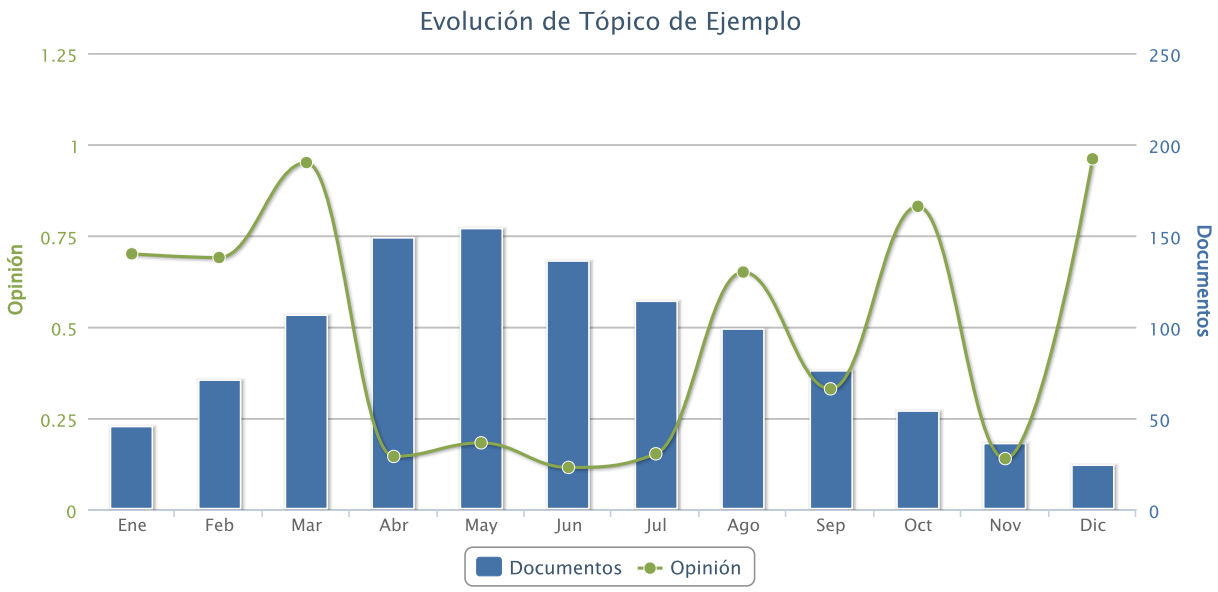

Figura 3.3: Ejemplo de gráfico por tópico

3.4.1. Recuperación de documentos y extracción de tópicos

Para cada documento se deben almacenar: la fecha de publicación, la fuente, la *url*, el título, el contenido sin procesar, la fecha en la cual fue añadido a la base de datos y toda la metadata asociada a él. En este caso, la metadata se almacenará serializada para permitir la inclusión posterior de información, a medida que nuevas fuentes de documentos traigan nueva información que no se tenía considerada antes.

Un ejemplo de documento a almacenar es una noticia publicada en un sitio web, ejemplos de cada campo se presentan a continuación:

- **Fecha de publicación**: `19/10/2012`
- **Fuente**: `http://www.latercera.com`
- **URL**: `http://diario.latercera.com/2012/10/19/01/contenido/mundo/8-120852-9...`
- **Contenido sin procesar (extracto)**: Tras meses de expectación y retrasos de último minuto, finalmente ayer el gobierno colombiano y las Fuerzas Armadas Revolucionarias de Colombia (Farc) dieron por inaugurada la segunda etapa del proceso de paz, cuya mesa de negociación se instalará el 15 de noviembre próximo en La Habana, Cuba.[2]
- **Título**: Farc desafían a Presidente Santos y dicen que no habrá "paz express"

[2]Fuente: La Tercera

- **Fecha de creación en la base de datos**: `19/10/2012`

Por otro lado, para los tópicos es necesario que en la base de datos se encuentre información sobre: fecha de creación, fecha de actualización, palabras que identifican al tópico en un periodo en particular, y la probabilidad de que el tópico contenga la palabra mencionada en el periodo en el cual se realiza la asociación.

Por ejemplo, para un tópico relacionado al fútbol, se deberá almacenar lo siguiente:

- **Fecha de creación**: `20/10/2012`
- **Palabras que identifican al tópico**: `balón (0.52), portería (0.45), delantero (0.32), gol (0.81), tarjeta (0.33), suspensión (0.17)`

Además, es necesario tener la relación entre los tópicos y los documentos desde los cuales se extraen. En esta relación se debe considerar la fecha de asociación y el puntaje que el modelo de extracción de tópicos le da a la pertenencia de un documento en relación al tópico correspondiente.

Finalmente, existe una tabla extra en caso de que se quiera incluir posteriormente el uso de jerarquías para la clasificación de los documentos recuperados. Esto es útil en caso de querer no sólo clasificar los documentos por tópicos, si no dar una clasificación especial de los documentos recuperados que se sepa de antemano que facilita la comprensión de la información extraída.

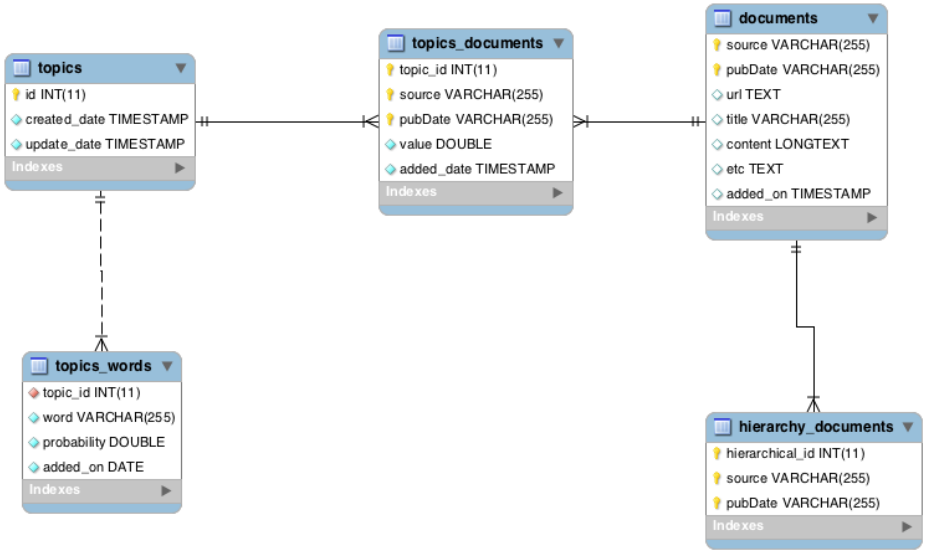

Figura 3.4: Modelo de datos para la recuperación de documentos y la extracción de tópicos

3.4.2. Recuperación de documentos opinados y extracción de opiniones

El modelo de datos que almacenará los documentos opinados enlazados con los tópicos ya extraídos se presenta en la figura 3.5.

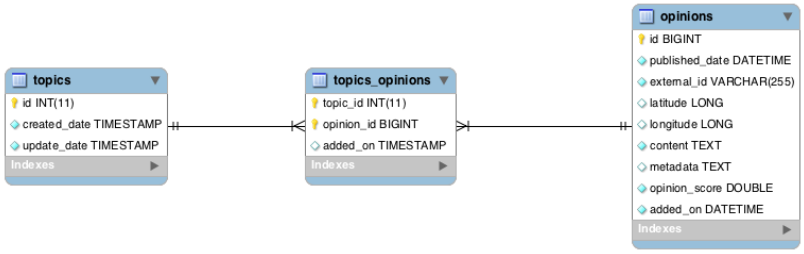

Figura 3.5: Modelo de datos para documentos opinados y extracción de opiniones

En el caso particular de la plataforma que se está rediseñando, la red social que se procesa es Twitter y debido a la gran cantidad de información que otorga esta red social hoy en día, es necesario que para cada documento extraído de esta fuente se almacenen: latitud y longitud desde donde y la fecha cuando se emitió del *tweet* en *Twitter*, el id del *tweet*, fecha en la cual la opinión fue enlazada con el tópico y finalmente el puntaje de opinión que tiene asociado ese documento opinado. Un ejemplo de *tweet* se presenta a continuación:

- **Latitud y longitud**: `-30.23123, 15.3929`
- **Fecha de emisión**: `19/10/2012`
- **Id del *tweet***: `a182b918cd3gef`
- **Contenido**: ‘‘`Que película más espectacular! :D`’’
- **Fecha de enlace con el tópico**: `21/10/2012`
- **Puntaje de opinión**: `0.3`

3.5. Proceso de negocio actual

Los procesos de negocio involucrados en la plataforma actual se describen en la figura 3.6, estos están descritos en base a cada una de las componentes del sistema y además el usuario que hace uso de la plataforma.

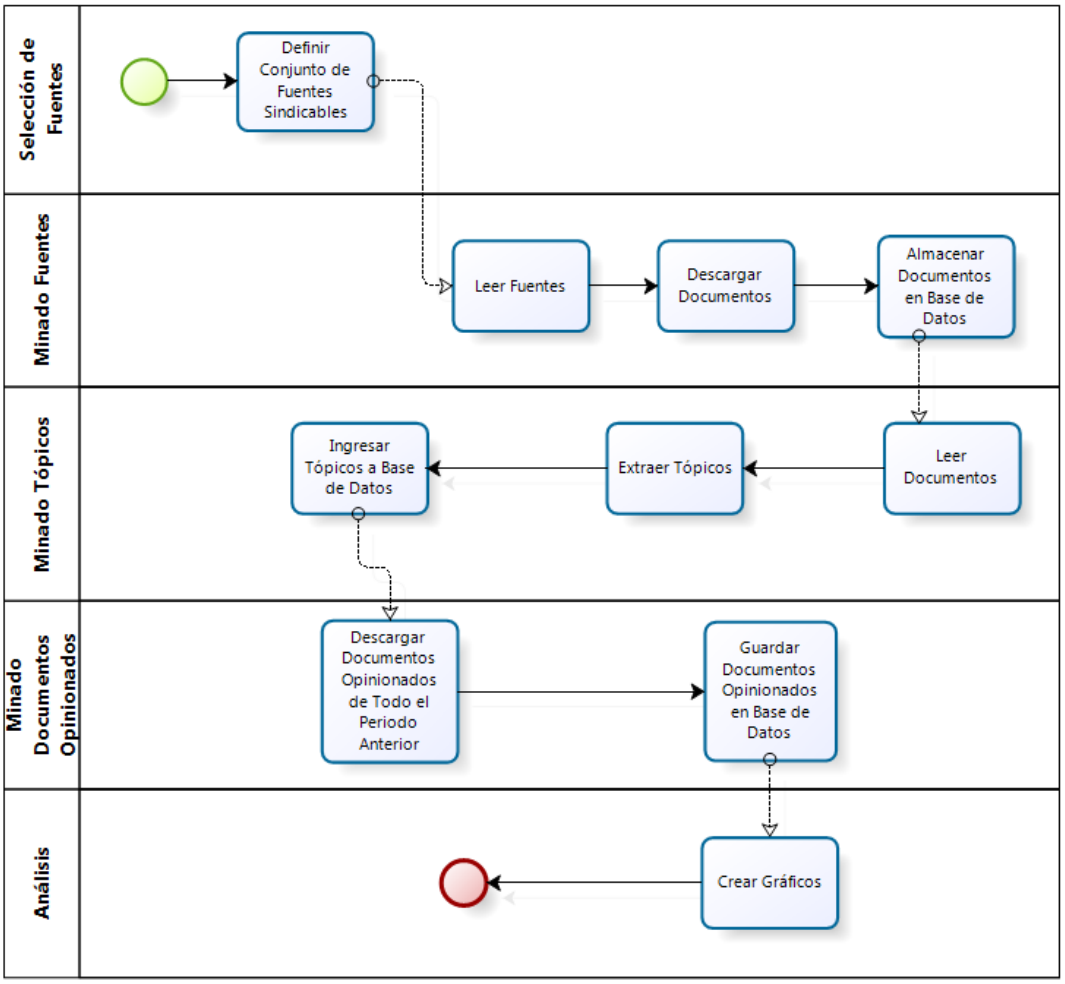

Figura 3.6: Procesos de Negocio en la plataforma actual

Capítulo 4

Requisitos de la solución

Una vez analizada y definida la situación actual del sistema bajo inspección, el segundo paso en el rediseño es dar a conocer los requisitos que debe cumplir la nueva plataforma, los cuales no sólo deben describir qué es lo que debe cumplir el software desarrollado, también deben ser capaces de dar a conocer cómo debe ser utilizada la plataforma por el usuario y finalmente qué requisitos debe cumplir la arquitectura que soporta el sistema que se desea construir. Por lo tanto, en este capítulo no sólo se enunciarán los requisitos que satisface la plataforma actualmente, también deben mencionarse aquellos que es deseable que esta cumpla.

Para definir los requisitos que se deben considerar en este rediseño, se analizaron las necesidades existentes en la plataforma actual, es decir las falencias detectadas a través del análisis de la situación actual, y las mejoras que se desean incluir para mejorar los resultados del sistema, las cuales se extraen de los objetivos específicos de esta memoria de título mencionados en el capítulo 1. De esta manera se logró determinar cuáles son los requisitos de usuario, de software y aquellos pertinentes a la plataforma que debe satisfacer la plataforma rediseñada para cumplir con todos los objetivos planteados.

4.1. Descripción del Software

Tal como se dio a conocer en el capítulo 1 en esta memoria de título se desea llevar a cabo el rediseño de la plataforma de detección de tendencias presentada en el capítulo anterior. Esta plataforma de detección de tendencias tiene como objetivo apoyar los estudios de inteligencia de mercado de Duam S.A. a través del minado de tópicos de documentos recuperados desde la Web, y además, la extracción de información sobre el sentimiento de la gente en las redes sociales.

4.2. Actores del Sistema

- **Usuario**: el usuario interactúa con el sistema ingresando fuentes al sistema y además observando los resultados de la plataforma a través del módulo de visualización de resultados.
- **Fuente de documentos**: Es cualquier sitio web con contenido sindicable que es minado por el módulo de recuperación de documentos.
- **Red social**: Es aquel sitio web donde la gente da a conocer sus opiniones e interactúan con otros usuarios del mismo.

4.3. Funciones del sistema

Los requisitos funcionales de la plataforma tras el rediseño son:

1. **Ingreso de fuentes**:

   El sistema debe permitir al usuario incorporar a él las fuentes que serán utilizadas para el minado de tópicos.

2. **Recuperación de fuentes RSS**:

   El sistema debe proveer un mecanismo para recuperar documentos a partir de fuentes en formato RSS.

3. **Definición de la duración de un periodo de trabajo**:

   El sistema debe permitir que el usuario defina la cantidad de tiempo tras la cual se realizará una iteración del algoritmo para detectar tópicos entre las noticias ya recuperadas. En otras palabras, el usuario debe ser capaz de definir la duración de un periodo de análisis de la plataforma.

4. **Definición de la cantidad de tópicos**:

   El sistema debe dar al usuario la capacidad de definir la cantidad tópicos que se espera obtener en una iteración de este, ya que dependiendo del nicho sobre el cual se estén buscando tópicos, es esperable que la cantidad de estos varíe.

5. **Extracción de tópicos**:

   El sistema debe permitir la extracción de tópicos desde un conjunto de documentos previamente recuperado durante un periodo de tiempo.

6. **Visualización de tópicos**:

   El sistema debe disponer de una visualización para los tópicos extraídos en cada periodo de trabajo a través de las palabras que los describen.

7. **Recuperación de documentos desde las redes sociales**:

   El sistema debe ser capaz de recuperar documentos opinados desde una red social en particular. En el caso particular de esta implementación, esta red social es *Twitter*.

8. **Extracción de opiniones desde documentos opinados**:

   El sistema debe proveer un mecanismo para detectar la polaridad de los documentos opinados recuperados.

9. **Extracción de fuentes potenciales**:

   El sistema debe proveer un conjunto de fuentes potenciales a partir de los documentos recuperados para la extracción de tópicos.

10. **Selección de nuevas fuentes**:

    El sistema debe proveer una interfaz que permita agregar las URLs RSS de las fuentes potenciales detectadas que se desean evaluar.

De estos requisitos, los que se se decidió agregar durante el rediseño son aquellos donde se señala el usuario debe ser capaz de obtener una lista de fuentes potenciales para su inclusión y posterior evaluación en el algoritmo de recuperación de documentos. Es decir, si se detecta que un sitio web puede ser una fuente potencialmente útil para la detección de tendencias, el usuario puede decidir incluir esta fuente para su evaluación en el módulo de recuperación de documentos, y si esta pasa ciertos criterios de evaluación que sea automáticamente agregada a la lista de fuentes que son utilizadas por el módulo de extracción de tópicos.

4.4. Requisitos de software

La definición de requisitos de software busca detallar qué es lo que debe lograr el sistema de detección de tendencias y cuáles son las mejoras que deben realizar a la plataforma ya existente todo visto desde el punto de vista del desarrollo de software.

1. **El software debe minar continuamente las fuentes de noticias seleccionadas**:

   De manera continua el software debe conectarse a fuentes ya definidas, recuperar los documentos nuevos y almacenarlos en una base de datos. La periodicidad del algoritmo de recuperación de documentos debe ser configurable por el administrador del software. Una restricción a considerar es que las fuentes de noticias deben encontrarse en formato *RSS* o *Atom*, decisión basada en el hecho de que la mayor parte de las fuentes de noticias en la actualidad proveen sus artículos a disposición en alguno de estos dos formatos y existen múltiples recursos que permiten el procesamiento de este tipo de fuentes de manera eficiente.

2. **El software deber ser capaz de detectar, sugerir y evaluar nuevas fuentes de noticias de manera autónoma**:

   Si bien el sistema debe otorgar la capacidad al usuario para modificar las fuentes de manera manual, se espera que el software sea capaz de sugerir nuevas fuentes de información. Para esto, debe considerar las fuentes presentes en el sistema y los documentos recuperados a partir de estas para sugerir fuentes que pueden ser un aporte potencial para la extracción de tendencias, y en caso de que sean aceptadas por el usuario deben ser monitoreadas y evaluadas por un algoritmo de calificación que decida si una fuente tiene relación alguna con el nicho que se está explorando y por lo tanto debe ser incluida en el análisis de tópicos.

3. **El software debe ser capaz de extraer documentos de distintas fuentes de manera paralela**:

   Para mejorar la eficiencia del sistema, la recuperación de documentos desde las fuentes que se estén minando debe realizarse de manera concurrente, permitiendo así que múltiples documentos de distintas fuentes sean descargados al mismo tiempo y por consiguiente aumentando la cantidad de fuentes que el sistema puede minar en un período determinado de tiempo, disminuyendo la pérdida de información debido a largos tiempos entre visitas a la fuente por parte del algoritmo de recuperación de documentos.

4. **El software debe extraer los tópicos a partir de las noticias relacionadas**:

   Periódicamente el software de extracción de tópicos debe correr el algoritmo que, dado todos los documentos recuperados entre la última ejecución de este y el tiempo actual, extraiga los tópicos relevantes. Además, el algoritmo debe ser capaz de relacionar las noticias almacenadas con los tópicos extraídos y enlazar estos últimos con tópicos ya existentes en semanas anteriores con un límite de tiempo definido de antemano.

5. **El software debe minar documentos opinados para cada tópico**:

   Para cada tópico extraído en un periodo dado, deben extraerse todos los documentos opinados desde Twitter generados en el mismo periodo que tengan relación con este y luego deben ser almacenados en la base de datos correspondiente.

6. **Extracción de opiniones sobre los tópicos de un periodo**:

   A partir de los documentos opinados asociados a un tópico en un periodo en particular, el software debe calcular el puntaje de opinión correspondiente para este. Debido a que este algoritmo ya está implementado, se espera realizar cambios en su implementación para mejorar su eficiencia, y además, incluir el procesamiento de emoticones para mejorar la precisión de este ya que estos son utilizados frecuentemente en las redes de microblogging para dar a conocer opiniones o sentimientos sobre el tema que se discute en el documento.

7. **Presentación de los resultados de manera gráfica**:

   Basándose en toda la información recaudada por los distintos módulos de la plataforma, el software debe presentar de manera visual el cruce entre información opinada y noticias para cada tópico a lo largo del tiempo.

8. **El sistema debe ser capaz de dar cabida a aplicaciones creadas por terceros con la información presente en ella**:

   Se espera el desarrollar la capa de presentación de resultados de manera tal que sea posible la creación a posteriori de una API que permita el uso de la información extraída por nuevas aplicaciones o por usuarios que deseen tener acceso a ella.

4.5. Requisitos de la plataforma

Debido a que la plataforma no es capaz de soportar grandes volúmenes de datos, será necesario rediseñar la plataforma con el objetivo de lograr modularidad y escalabilidad tanto horizontal, que apunta a mejorar el rendimiento de esta aumentando las máquinas utilizadas, como vertical, que se enfoca en cómo aumentar la capacidad de procesamiento a través de mejoras en las máquinas que están siendo utilizadas. A continuación se presentan los requisitos que deben ser satisfechos para lograr que el sistema sea escalable en ambas dimensiones:

1. **Diseño de software modular**:

   Cada componente de la plataforma deberá actuar de manera independiente de los otros módulos que coexiste con ella. Este requisito es la base de lograr una plataforma escalable capaz de manejar grandes volúmenes de datos.

2. **Paralelización de los procesos**:

   Cada módulo de la plataforma debe ser capaz de funcionar de manera concurrente con otras instancias de si mismo.

3. **Creación de puntos de acceso para aplicaciones orientadas a usuarios externos**:

   Además de reestructurar la plataforma para que esta sea modular, es necesario considerar la inclusión a posterior de aplicaciones que expongan esta información, ya sea procesada o no, a usuarios externos de la empresa, ya que es deseable en el corto plazo poder dar servicios de agregación de contenidos a externos u otro tipo de información que sea valiosa para clientes potenciales.

4. **Extensibilidad del sistema**:

   El rediseño de la plataforma debe permitir que el sistema sea extensible, ya sea agregando nuevas funcionalidades a sus módulos o agregando nuevos módulos al sistema que trabajen sobre los datos.

En el capítulo siguiente se dará a conocer el diseño de la solución, dando a conocer las componentes ya existentes en el sistema, cuál es la situación inicial de estas desde un punto de vista arquitectural y de diseño de software y además, denotando todos los cambios que se proponen a cada una de ellas, junto con las nuevas funcionalidades que serán implementadas en base a los requisitos definidos en este capítulo.

4.6. Requisitos de Rendimiento

1. **El sistema debe ser capaz de procesar la información en menos de un periodo de trabajo**:

   El sistema debe ser capaz de minar los documentos del periodo de trabajo y extraer tópicos a partir de estos, junto con recuperar documentos opinados del periodo de trabajo anterior y extraer la polaridad de cada uno de ellos, sin que la cantidad de tiempo necesaria para ello supere el tiempo de duración de un periodo de trabajo definido por el usuario.

2. **Bajo tiempo de respuesta por parte de aplicaciones orientadas a usuarios externos**:

   El sistema debe ser capaz de responder en un tiempo prudente a las solicitudes de información realizadas por aplicaciones orientadas a usuarios externos, esto quiere decir que cualquier tipo de petición que se realice para obtener datos no debe ver su rendimiento afectado por los distintos procesos que ocurren en la plataforma.

4.7. Atributos no funcionales

En esta sección se dan a conocer los requerimientos no funcionales que debe satisfacer la plataforma:

1. **Rendimiento**:

   El sistema debe ser capaz de realizar las partes críticas del análisis de documentos en un tiempo acorde al periodo de trabajo definido y además, dar acceso a terceros, si se desea, a la información recuperada en un tiempo prudente para ellos.

2. **Escalabilidad**:

   El sistema debe ser capaz de escalar tanto vertical como horizontalmente, de manera de permitir la inclusión de múltiples fuentes y usuarios externos sin que esto provoque una merma en el rendimiento.

3. **Orientación a Objetos**:

   El sistema debe ser modelado considerando patrones de diseño conocidos, haciendo especial hincapié en la modelación de cada entidad del sistema como un objeto.

4. **Extensibilidad**:

   El sistema debe ser fácilmente extensible a recuperar otro tipo de documentos objetivos o documentos opinados desde otras redes sociales.

5. **Asincronía**:

   El sistema debe permitir la lectura de la información extraída en cada módulo sin interferir con el proceso de detección de tendencias.

4.8. Casos de Uso

A continuación se presentan los casos de uso correspondientes al usuario y a la plataforma.

1. **Caso de uso**: Ingresar fuente

   - **Actores**: Usuario
   - **Propósito**: Dar a conocer al sistema desde que fuentes se recuperarán documentos objetivos para la extracción de tópicos.
   - **Resumen**: El usuario ingresa en un formulario la URL del punto de entrada RSS desde el cual se desea recuperar los documentos.
   - **Tipo**: Primario y esencial

2. **Caso de uso**: Visualizar tópicos

   - **Actores**: Usuario
   - **Propósito**: Dar a conocer al usuario los tópicos que se han extraído en los distintos periodos de trabajo junto con las palabras que los representan a lo largo del tiempo.
   - **Resumen**: El usuario hace una solicitud para obtener la información de los tópicos presentes en el sistema. Luego, la plataforma recolecta toda la información necesaria y la despliega de vuelva al usuario. En caso de ser necesario, el usuario puede escoger un periodo de tiempo en particular para visualizar los tópicos.
   - **Tipo**: Primario y esencial

3. **Caso de uso**: Visualizar tendencias

   - **Actores**: Usuario
   - **Propósito**: Dar a conocer al usuario el comportamiento de cada tópico recuperado a lo largo del tiempo en cuanto documentos y opiniones concierne.
   - **Resumen**: El usuario hace una solicitud a la plataforma para visualizar de manera gráfica la evolución de los tópicos en el tiempo.

- **Tipo**: Primario y esencial

4. **Caso de uso**: Ingresar URL de RSS para fuentes potenciales

   - **Actores**: Usuario
   - **Propósito**: Permitir que el usuario ingrese fuentes en formato RSS para que la plataforma pueda recuperar documentos desde fuentes potenciales para su posterior análisis.
   - **Resumen**: El usuario ingresa en un formulario (uno para cada fuente potencial) la URL del punto de entrada RSS desde el cual se desea recuperar los documentos.
   - **Tipo**: Primario y esencial

5. **Caso de uso**: Incluir en la lista de fuentes del módulo de recuperación de documentos, fuentes potenciales aprobadas

   - **Actores**: Usuario
   - **Propósito**: Permitir al usuario la inclusión en la lista de fuentes de la plataforma aquellas que han sido aprobadas por el algoritmo de calificación de relevancia.
   - **Resumen**: Se le presenta al usuario una lista de fuentes potenciales aprobadas, en la cual puede escoger cuales deben ser incluidas en el sistema para que sean minadas desde el próximo periodo de trabajo.
   - **Tipo**: Primario y esencial

El diagrama de casos de uso de la plataforma es presentado en la figura 4.1.

4.9. Requisitos desde el punto de vista de negocio

Los cambios que se realicen a la plataforma deben apuntar a disminuir los costos operacionales de esta, tanto en recursos utilizados para la recuperación de documentos y el posterior procesamiento de estos, como en la cantidad de tiempo que se requiere para incluir nuevos clientes y aplicaciones que permitan nuevos modelos de negocio con la plataforma. Esto se traduce en los siguientes requisitos:

- **Aplicaciones orientadas a terceros**:

  Con el fin de proveer al sistema de posibilidades de expansión en cuanto a su modelo de negocio, es necesario que sea sencillo desarrollar aplicaciones orientadas a terceros que utilicen la información presente en el sistema.

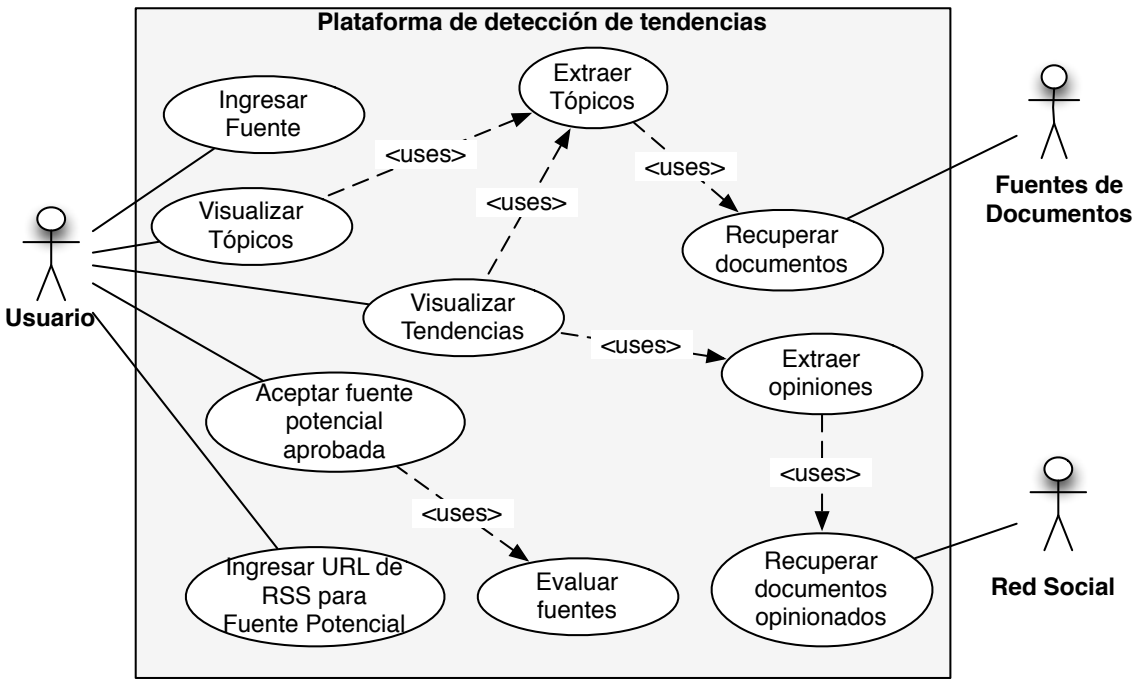

Figura 4.1: Diagrama de casos de uso

- **Tiempo de inclusión de nuevos clientes**:

  El sistema debe proveeer una manera sencilla de agregar nuevos clientes, a través de la inclusión de nuevas fuentes al sistema.

- **Reducción de uso de recursos**:

  La cantidad de recursos utilizados por cada fuente extraída debe disminuir tras el rediseño, y debe permitir la escalabilidad a bajo costo marginal de la plataforma, considerando costo marginal como la cantidad marginal de recursos necesaria por una nueva fuente incluida en el sistema.

Capítulo 5

Descripción de la solución

El tercer paso en todo rediseño de procesos es el definir qué es lo que se realizará para satisfacer los requisitos previamente definidos, por lo tanto en este capítulo se describirán en forma esquemática los cambios propuestos a la plataforma de detección de tendencias de la misma manera que se realizó el análisis de su situación actual, partiendo por la arquitectura física, seguida por la arquitectura de software y finalmente las mejoras que se implementarán a la plataforma original en cuanto a implementación de algoritmos se refiere, para otorgar flexibilidad y escalabilidad a medida que el uso de las herramientas desarrolladas vaya aumentando.

5.1. Arquitectura Física

A lo largo de esta sección se dará a conocer la nueva arquitectura física propuesta en este rediseño, descrita por un lado por la infraestructura necesaria para llevar a cabo la recuperación y el procesamiento de datos de manera modular; y por el otro, en la arquitectura utilizada para crear servicios web que puedan ser ofrecidos a usuarios externos. Esta separación existe para poder aislar de manera segura la información de uso interno que crean los algoritmos de recuperación y procesamiento de datos, y además, poder proveer escalabilidad a los servicios web que puedan ser ofrecidos junto con la plataforma para permitir el acceso a la información por parte de terceros.

5.1.1. Recuperación y procesamiento de datos

Para comenzar se dará a conocer la manera en que las distintas componentes del módulo de recuperación y procesamiento de datos interactuarán para recuperar, procesar y analizar los datos y documentos presentes en la web con el fin de extraer la información y el conocimiento necesario sobre las tendencias presentes en la web.

La arquitectura física de esta parte de la solución (figura 5.1) se basa en tener tres nodos independientes, cada uno encargado de uno de tres pilares que son considerados fundamentales en toda plataforma del gestión del conocimiento:

- **Recuperación de la información**: En este nodo se encontrarán todas las aplicaciones encargadas de recuperar información de la web. Específicamente, se deben ubicar los crawlers de documentos y de opiniones. Más adelante en este capítulo se ahondará en cómo están diseñados estos y de qué manera operarán, junto con los cambios realizados para lograr un aumento en la eficiencia de estos.

- **Capa de persistencia**: Es la capa física donde se almacenan todos los datos, sean estos sin procesar, o información extraída por parte del software de extracción de conocimiento. En esta capa sólo vive el servidor de base de datos.

- **Extracción de conocimiento**: Capa donde se realiza todo el procesamiento y la lógica sobre los datos recuperados para extraer el conocimiento necesario para alimentar la plataforma de detección de tendencias. Entre los procedimientos que se realizan en esta capa, se encuentran: limpieza de documentos, extracción de tópicos y extracción de opiniones a partir de documentos opinados entre otros.

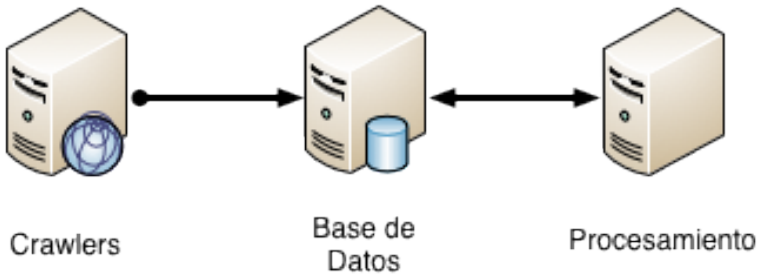

Figura 5.1: Arquitectura física para el procesamiento de datos

La separación de capas que propone esta nueva arquitectura, a diferencia de la original, permite manejar muchas más fuentes de información y una mayor cantidad de documentos, ya que la reducción de cohesión entre las componentes del software permite paralelizar la mayoría de los procesos de recuperación de documentos que se realizan en la plataforma, ya sea dentro de una misma máquina paralelizando las fuentes que se están minando, o a través de varios servidores lo que significa que el sistema es escalable horizontalmente y verticalmente.

A continuación se listan algunas de las medidas que pueden ser tomadas para permitir que la plataforma escale esta nueva arquitectura en caso de ser necesario:

- Creación de clústers de bases de datos que permitan la recolección y el manejo de una gran cantidad de datos.
- Creación de réplicas de sólo lectura para permitir que los modelos existentes en la capa de extracción de conocimiento accedan a la información de la base de datos sin mermar el rendimiento general del sistema.
- Inclusión de balanceadores de carga entre los usuarios externos y el servidor que recibirá todas las consultas de estos últimos.
- Uso de distintas IPs públicas para aumentar el límite de *tweets* que es posible recopilar por hora ya que la API de Twitter hace uso de este recurso para limitar la cantidad de peticiones que pueden ser realizadas en una unidad de tiempo.

Además de permitir que el sistema escale a medida que la cantidad de información recuperada y procesada en este aumente, tener esta arquitectura abre la posibilidad de agregar servicios adicionales a la plataforma, como por ejemplo un servicio de publicación de datos con el fin de permitir la creación de aplicaciones de usuario como se explica a continuación.

5.1.2. Aplicaciones de usuario

Uno de los requisitos que surgió a la hora de diagnosticar los problemas de la herramienta, fue el de extender esta plataforma para que fuese capaz de albergar aplicaciones orientadas a usuarios externos sin conocimiento de la herramienta, como por ejemplo la creación de reportes automáticos, portales web con información sobre tendencias, monitoreo de redes sociales, etc.

Para satisfacer este nuevo requisito, es decir, la creación de aplicaciones que se alimenten de esta plataforma y estén orientadas hacia usuarios externos, es necesario considerar además de la arquitectura física mencionada para el procesamiento de datos, una arquitectura de tres capas (figura 5.2) comúnmente utilizada en el desarrollo de aplicaciones que exponen servicios. Las capas que componen la arquitectura física para esta parte del sistema son:

- **Capa de clientes**: En la capa del cliente, se encuentran todos los usuarios de los servicios que se pretendan ofrecer. Esto quiere decir, que cualquier dispositivo con conexión a Internet y un navegador web, puede acceder a los sitios web que se desarrollen posteriormente siempre y cuando tengan la autorización necesaria. Además, en esta capa se encuentran toda aplicación externa que haga uso de las APIs creadas en la plataforma de detección de tendencias.

Figura 5.2: Arquitectura física de aplicaciones para usuarios externos

- **Capa de aplicaciones**: En la capa de aplicaciones se encuentra además de los módulos que sustentan la plataforma, los cuales fueron descritos en la subsección anterior, todas las aplicaciones desarrolladas para esta plataforma que están encargadas de moldear los datos ya procesados según se requiera para cada aplicación.

- **Capa de datos**: Esta capa es similar a la capa de datos mencionada anteriormente en la sección referente a la recuperación y el procesamiento de datos, con una importante diferencia: sólo permite el acceso de lectura desde el servidor presente en la capa de aplicaciones, previniendo así cualquier corrupción de los datos que pudiese provenir de usuarios externos.

De estas tres capas, sólo la de datos y la de aplicaciones se ubican dentro del sistema. Si bien en la capa de aplicaciones se considera el desarrollo de software que pueda ser accedido por usuarios externos, el objetivo principal de ella es albergar APIs que den acceso a la información para que plataformas externas hagan uso de ella.

5.2. Arquitectura de Software

Una vez propuesta la arquitectura física que soportará el rediseño de esta plataforma, en esta sección se da a conocer cómo se estructurará el software de manera que sea capaz de aprovechar las ventajas de ésta.

5.2.1. SOA

Debido a que cada componente de la aplicación debe ser capaz de funcionar por si sola y no depender de manera sincrónica de los otros módulos desarrollados, se optó por utilizar una arquitectura de tipo SOA[1], la cual hace referencia a que cada componente de la plataforma debe poseer la capacidad de interoperar con otros servicios que se encuentren en esta. Cada uno de estos servicios

[1]Service Oriente Architecture

puede ser identificado como una unidad de negocio bien definida, la cual, desde el punto de vista de software, puede ser reutilizado para distintos propósitos.

El uso de una arquitectura tipo SOA permite que cada una de las componentes del software sea desarrollada de manera independiente, sin tener que esperar a tener desarrolladas todas las componentes del software para realizar pruebas completas, si no que todas las pruebas se van realizando de manera aislada a medida que se van construyendo los módulos necesarios; y además, permite la escalabilidad de los sistemas debido a la abstracción presente en la comunicación entre cada una de las capas y la baja cohesión que existe entre los módulos que componen la plataforma en si.

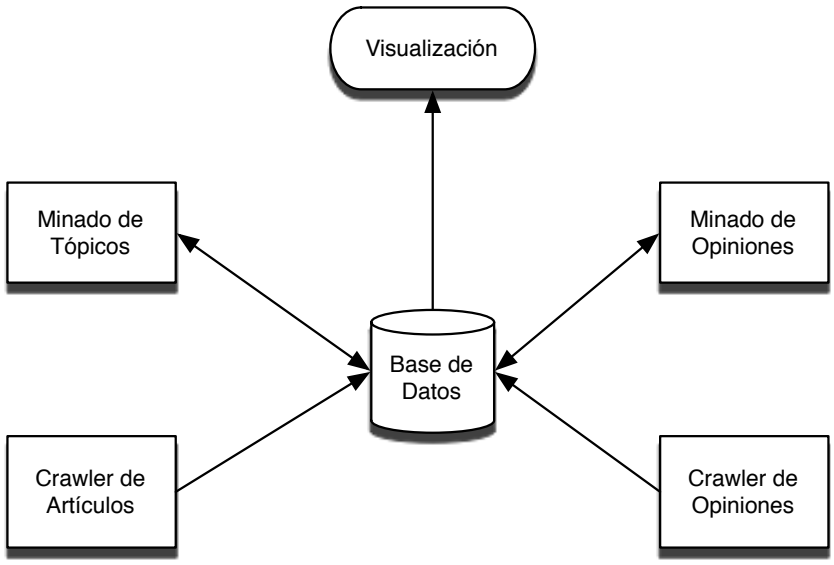

Figura 5.3: Arquitectura orientada a Servicios de la plataforma

En la figura 5.3 se da a conocer un diagrama de interacción de cada una de las componentes de este sistema y la manera en que se relacionan entre ellas de manera abstracta. Cabe destacar que inicialmente todos los módulos del sistema estarán conectados a una misma base de datos, lo que puede cambiar dependiendo de las medidas que se tomen para aumentar la estabilidad del sistema a medida de que la cantidad de datos a procesar vaya aumentando. Cada una de las componentes de este ecosistema de aplicaciones SOA se detallará más adelante en la sección 5.3, detallando su implementación y el rol que juegan dentro de toda la plataforma.

5.2.2. Arquitectura de Software propuesta para aplicaciones externas

Para la creación de aplicaciones orientadas a entidades externas, sean estos usuarios o aplicaciones, se sugiere desde un punto de vista de software el uso de una arquitectura de tres capas como es usual en el desarrollo de aplicaciones web el cual se presenta en la figura 5.4. Entre las aplicaciones

orientadas a externos se encuentran inicialmente la herramienta de visualización y cualquier API que se desee tener pública para exponer los datos obtenidos por la plataforma de detección de tendencias.

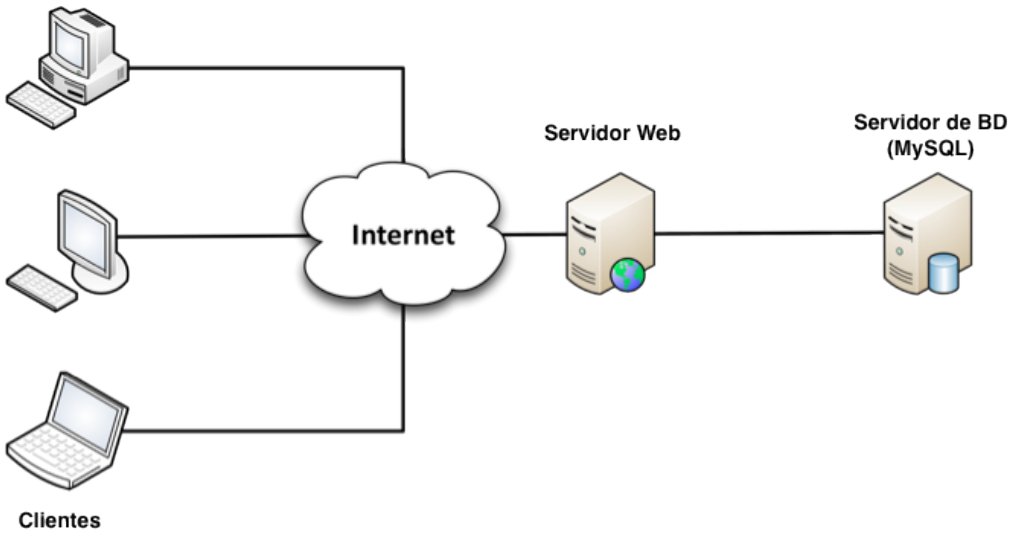

Figura 5.4: Arquitectura de software propuesta para aplicaciones orientadas a usuarios externos

- **Capa de clientes**: Están todos los usuarios de los servicios que se pretendan ofrecer. Esto quiere decir, que cualquier dispositivo con conexión a Internet y un navegador web, puede acceder a los sitios web que se desarrollen posteriormente.
- **Capa de aplicaciones**: Se encuentra el servidor web que responde las peticiones de los clientes. Este se preocupa de enviar la petición a la aplicación que corresponda a través de distintas interfaces, sea esta *CGI*, *fastCGI* o *WSGI* entre otros. En caso de que la aplicación web desarrollada requiera de un servidor de aplicaciones, es posible configurar este último para que actúe como un servidor web capaz de recibir y responder las peticiones de los clientes.
- **Capa de datos**: Esta capa es similar a la capa de datos mencionada anteriormente en la sección 5.1.1 referente a la recuperación y el procesamiento de datos, con una importante diferencia: sólo permite el acceso de lectura desde el servidor presente en la capa de aplicaciones, previniendo así cualquier corrupción de los datos que pudiese provenir por parte de los usuarios.

Cada una de estas capas se relaciona con una componente del software a desarrollar, la capa de clientes se relaciona con las *vistas*, las cuales se encargan de desplegar el contenido según el usuario lo solicite; los *controladores*, que son aquellos módulos encargados de toda la lógica de negocios que sea necesaria para mostrar datos al usuario; y finalmente los *modelos*, cuya única función es manejar la comunicación entre la capa de persistencia (bases de datos, archivos, etc.) y los controladores.

Este tipo de diseño arquitectural para desarrollar software es conocido como el modelo **MVC** debido a las siglas de cada una de sus componentes: **M**odelo, **V**ista y **C**ontrolador.

5.3. Diseño de software

A continuación se describen las mejoras en implementación realizadas a cada una de las componentes del sistema, dando a conocer los cambios hechos a los algoritmos utilizados y los detalles sobre la puesta en marcha de estos en la plataforma. Además, en las secciones 5.4 y 5.5 se describirán dos nuevos algoritmos que serán utilizados por la plataforma para mejorar los resultados que esta provee y los cuales son la principal contribución de esta memoria de título.

5.3.1. Módulo de recuperación de documentos

Según se describió en el capítulo 3, el principal problema este módulo es el procedimiento secuencial que utiliza para recorrer las fuentes, lo que no permite que la capacidad de procesamiento de información de esta componente escale acorde al aumento de la cantidad de fuentes utilizadas.

Por otro lado, si una de las fuentes presenta problemas, como por ejemplo contenido corrupto o problemas de acceso, puede provocar que la ejecución del software se vea interrumpida o que tome un tiempo excesivo de ejecución causando la pérdida de documentos para las fuentes que se deben procesar a continuación, lo que puede conllevar resultados sesgados o incorrectos debido a que se estaría utilizando un subconjunto de la información realmente disponible.

Para solucionar este problema se propone recuperar cada fuente de manera paralela, es decir, crear un *thread* por cada una de ellas que se desee minar. El uso de *threads* para realizar el procesamiento paralelo de fuentes tiene una serie de ventajas, entre ellas el reducir la cohesión entre el minado de una fuente con respecto a las otras, permitiendo así la escalabilidad horizontal del sistema; la reducción de errores en el sistema debido a la separación entre el proceso de recuperación de cada una de las fuentes; y finalmente, minimizar la cantidad de documentos perdidos debido a errores que se puedan presentar en la conexión con cada una de las fuentes sindicables.

Incorporando estas mejoras, el nuevo algoritmo de recuperación de documentos sería el que sigue:

Algoritmo 5.3.1: Recuperación de documentos paralelizado

Data: $\{F_i\}_{i \in \mathbb{N}}$, t

```
1 for i ← 1 to ‖{F_i}_{i∈ℕ}‖ do
2     createFeedThread(F_i);
3     sleep(T);
```

Algoritmo 5.3.2: `createFeedThread`(F_i, t)

Data: F, t

```
1 while True do
2     XML ← retrieveXML(F_i, t);
3     documents := [];
4     for d in parseXML(XML) do
5         documents ← documents ⋃ d;
6     saveToDatabase(documents);
7     sleep(T);
```

Donde cada documento d posee la misma estructura anterior al rediseño, es decir, una tupla con los siguientes datos:

- Fuente F: sitio desde donde se obtuvo este documento.
- Tiempo t: donde la publicación fue creada en la fuente F.
- Contenido c: Todo el contenido textual del documento.
- URL h: identificador único que identifica la ubicación de este documento en el servidor desde donde se recuperaron los documentos.
- Metadata M: El conjunto M involucra toda aquella información relevante al documento que no fue considerada en el modelamiento previo de un documento. Ejemplos de información que cae en este conjunto son los *tags*, *categorías*, *fuente*, etc.

Por cada iteración del algoritmo de recuperación de documentos para una fuente en particular, un conjunto consistente de todos los documentos recuperados es enviado directamente en la capa de persistencia, haciendo uso del método `saveToDatabase`, reduciendo así la cantidad de consultas realizadas a la base de datos y disminuyendo significativamente el impacto que tiene la recolección de documentos de documentos en el rendimiento de la plataforma.

Además, entre cada iteración del algoritmo que mina una fuente en particular se debe esperar un tiempo fijo para no saturar el servidor que se está consultando con peticiones y además se mejora el rendimiento general del módulo de recuperación de documentos al reducir la cantidad de peticiones realizadas para recuperar nuevos documentos.

5.3.2. Módulo de minado de tópicos

Tal como se describió en el capítulo 3 el módulo de minado de tópicos no necesita de ninguna mejora ya que el algoritmo que toma la mayor parte del tiempo es el de Latent Dirichlet Allocation y está lo suficientemente optimizado para que cualquier cambio que se realice sobre él atraiga una mejora significativa al rendimiento general de la plataforma. Por otro lado, los algoritmos de limpieza de documentos no consumen mucho tiempo de procesamiento y son simples de programar, por lo que realizar optimizaciones en esta área tampoco es de utilidad alguna si se busca mejorar el rendimiento de este módulo.

Como nota final, debido a que los algoritmos de este módulo sólo son ejecutados periódicamente no es necesario que estén altamente optimizados si esto implica una disminución en la calidad de los resultados y además, como estos están enfocados en unir todas las componentes de la plataforma la cantidad de trabajo paralelizable es mínima, reduciendo la cantidad de mejoras posibles para permitir la escalabilidad horizontal de este.

5.3.3. Módulo de recuperación de opiniones

Al igual que el módulo de recuperación de documentos, hay una serie de mejoras que se pueden realizar para que el minado de opiniones sea más eficiente y pueda ser distribuido de mejor manera en múltiples servidores, facilitando así la escalabilidad horizontal junto con permitir la recuperación de una mayor cantidad de documentos por unidad de tiempo si es que se considera el hecho de tener diversas máquinas en distintas ubicaciones para así poder minar Twitter de manera más eficiente y rápida aún respetando los límites de uso que estos imponen.

Por lo tanto, el algoritmo presentado en la sección 3.3.3 puede ser modificado de manera tal que múltiples conjuntos de tópicos sean minados de manera independiente a lo largo del periodo de recolección de opiniones. Esta modificación considera el límite de $\mathcal{Q}$ consultas por hora que impone Twitter, sólo que lo paraleliza logrando que cada uno de los N servidores sea capaz de realizar aquella cantidad de consultas por hora.

Para simplificar el trabajo de cada servidor y prescindir de una entidad que controle qué tópicos están siendo minados por cada servidor, se considerará clases de equivalencia de los tópicos en función del índice que estos ocupan en la base de datos. Así, el primer servidor se encargará de todos aquellos cuales se cumpla la condición $index \,\%N = 0$, es decir, que el módulo del índice que le corresponda, con la cantidad total de servidores que se posea sea 0. Así, se puede generalizar esta condición de la siguiente manera:

$$\tau_i \in Server_j \text{ si y sólo si } i\,\%N = (j-1)\ \forall j = 1 \ldots N \tag{5.1}$$

En base a esto, se puede realizar el rediseño de este módulo considerando que se está trabajando con N de servidores y un modelo de tópicos $\mathcal{T}$ con una cantidad T de tópicos a minar. Las modificaciones a realizar deben enfocarse en paralelizar la recuperación de documentos opinados de manera tal que en cada servidor se procesen sólo aquellos tópicos que cumplan con la ecuación 5.1. El algoritmo modificado que se presenta a continuación es aquel que será ejecutado en una máquina en particular y recibe como parámetros la estructura de tópicos $\mathcal{T}$, la cantidad S de servidores disponibles y el índice de la máquina s:

Algoritmo 5.3.3: Minado de opiniones paralelizado

Data: $\mathcal{T}, t, n, N, s, S, \mathcal{Q}$

```
1  hourlyQueries := 0;
2  forall τ_i ∈ T | i%S = s − 1 do
3      documents_i := [];
4      queries_i = generateQueries(T, t, n, N);
5      forall query ∈ queries_i do
6          retrievedDocuments = TwitterAPI.search(query);
7          forall retrievedDocument ∈ retrievedDocuments do
8              d_i ← polaridad(retrievedDocument);
9              documents_i.append(d_i);
10         hourlyQueries++;
11         if hourlyQueries == Q then
12             hourlyQueries := 0;
13             sleepUntilNextHour();
14     documents_i.saveToDatabase();
```

En esta nueva metodología de recuperación de documentos opinados cada servidor realizará $\mathcal{Q}$ consultas por hora en relación al conjunto de tópicos que le corresponda. Al usar una distribución uniforme de tópicos para las máquinas disponibles se busca que la cantidad de consultas realizadas por cada servidor sea similar permitiendo que exista un balance natural de carga entre todos los crawlers de documentos opinados.

5.3.4. Módulo de detección de tendencias

Sólo es necesario realizar los cambios necesarios para utilizar una arquitectura de software que siga el modelo **MVC** mencionado con anterioridad. Así, la extracción de información se realiza ahora desde la capa de persistencia, y se reescribió ligeramente el código de este módulo con el fin de construir gráficos, y la manera en que estos se despliegan al usuario estarán separadas, reduciendo la cohesión de código, facilitando la extensión del módulo agregando nuevas funcionalidades y finalmente disminuyendo la cantidad de trabajo a realizar en caso de que se quieran agregar nuevas métricas o maneras de representar los datos obtenidos.

5.4. Adición de nuevas fuentes

En base al análisis realizado a la situación actual de la plataforma, se descubrió que una limitación del modelo de recuperación de documentos que era utilizado por la esta es el hecho de que el minado de documentos está restringido a un conjunto de fuentes preexistente que no es modificada a lo largo del tiempo a menos que el usuario se dé el trabajo de hacerlo.

La mayoría de los documentos que son descargados y almacenados por el módulo poseen enlaces a otras fuentes los cuales son removidos junto con todo el `html` que se encuentra en los documentos originales para poder utilizar estos como entrada para los algoritmos de limpieza de documentos que se llevan a cabo en el módulo de extracción de tendencias, tales como remover stopwords o el algoritmo de stemming. Debido a esta pérdida de contenido, la plataforma no es capaz de aprovechar en su totalidad los documentos recuperados y en particular, no hace uso de los enlaces presentes en los documentos originales.

Los enlaces presentes en los documentos generalmente hacen referencia a documentos relacionadas al tema que está en discusión en este, sean estos fuentes, anexos, artículos con otros puntos de vista, etc. y si se hiciera uso de estos enlaces, sería posible obtener nuevas fuentes a incorporar en la plataforma para así obtener una mayor cantidad de información sobre el subconjunto de la web que se está analizando. Por ejemplo, si en los documentos recuperados desde el conjunto de fuentes actual se determina que un nuevo sitio es referenciado frecuentemente, es posible determinar que este es una afluente potencial de nuevos documentos relacionados al ecosistema que se está analizando.

En base a lo anterior, para evitar la pérdida de la información mencionada, se propone incorporar al módulo de recuperación de documentos un algoritmo que permita seleccionar de manera automática fuentes que puedan ser potenciales aportes al conjunto ya existente de estas y posteriormente, un algoritmo capaz de evaluarlas para determinar si pueden aportar información a la plataforma.

En [10, 29] se discuten algoritmos de calificación de relevancia que permiten la evaluación au-

tomática de nuevas fuentes con el objetivo de descubrir comunidades en redes de blogs. Debido a la naturaleza del conjunto de fuentes utilizadas como entrada para el módulo de recuperación de documentos, se espera que cada elemento de esta pertenezca a una comunidad, aún cuando el nivel de especificidad de esta pueda variar, por lo que la inclusión de un algoritmo de esta naturaleza permitirá evaluar fuentes potenciales las cuales serán escogidas en base a la frecuencia que estas tengan en los documentos recuperados.

La metodología que se propone para tratar este problema consiste de dos pasos: primero se seleccionan las fuentes potenciales en base a su frecuencia de aparición en los documentos y luego están son evaluadas para determinar si pertenecen al nicho que se está analizando y por lo tanto pueden aportar información al algoritmo de extracción de tópicos y a si mismo a la plataforma en su totalidad.

A continuación se da a conocer el algoritmo a utilizar para extraer fuentes potenciales a partir de la información textual recuperada por el módulo de recuperación de documentos. Se espera que éste sea ejecutado de manera periódica y no cada vez que se recupera un documento, ya que para determinar si una fuente tiene una frecuencia de aparición relevante se necesita una colección de documentos provenientes de todas las fuentes que se estén analizando.

La función `extractFeedURLs` se encarga de extraer todos las urls de las fuentes presentes en los enlaces de un documento, con ciertas salvedades: al hacer uso de documentos provenientes de la Web, muchos de estos enlaces corresponden a enlaces de avisos publicitarios, por lo que deberán ser ignoras para evitar que incorporen ruido al algoritmo; además, debe obtenerse sólo el dominio de la fuente para evitar considerar múltiples veces la misma fuente. Si bien en algunos casos no debe usarse la URL completa como es el caso de las URLs que apuntan a un servicio que haga uso de un agregador de noticias como FeedBurner, estos enlaces sólo apuntan a la fuente misma, por lo que pueden ser ignoradas. Para obtener la URL de una fuente inicialmente se definieron las siguientes reglas:

- Si la URL posee una componente de *query*, es decir, la parte de la URL que contiene datos que son utilizados por la aplicación que publica el sitio web, esta debe ser removida.
- Si la URL apunta directamente a un archivo, por ejemplo un archivo `html` o `php`, se utilizará sólo el dominio.

Algoritmo 5.4.1: Detección de fuentes potenciales

Data: $\{d_i\}_{i \in \mathbb{N}}$

```
1  feasibleFeeds = [];
2  forall document ∈ {d_i}_{i∈ℕ} do
3      feeds = extractFeedURLs(document);
4      forall feed ∈ feeds do
5          if database.updateFeedCount(feed) then
6              feasibleFeeds.append(feed);
7  average = database.getFeedCountAverage(feasibleFeeds);
8  forall feed ∈ database.getFeedData(feasibleFeeds) do
9      if feed.count > average then
10         createPossibleNewFeed(feed);
```

La cantidad de veces que aparece una url en un documento no es considerada ya que sólo importa saber cuántos documentos apuntan a un dominio. Si se considerara todas las veces que un dominio aparece en un documento, se estaría sesgando el análisis en base al estilo de escritura de una fuente en particular, ya que algunas citan múltiples veces a una fuente dentro de un documento y por el contrario, algunas ni siquiera realizan citaciones de las fuentes de las que consiguen la información, sólo lo muestran como fuente del artículo.

La función `database.updateFeedCount(feed)` aumenta en uno la cantidad de apariciones de la fuente entregada como parámetro, y devuelve `true` si la fuente aún no ha sido migrada a la lista de fuentes a evaluar y `false` en caso contrario.

Además, se hace uso de las funciones `getFeedCountAverage(feeds)` que se encarga de obtener el promedio entre todas las apariciones de las urls que se entregan como parámetro; y la función `createPossibleNewFeed(feed)` que crea una nueva entrada en la lista de fuentes candidatas que deben ser evaluadas en el segundo paso de esta metodología y marca como ya procesada la nueva fuente, por lo que será ignorada por futuras iteraciones de este paso de la metodología. Así, sólo se considerará en la lista de fuentes potenciales todas las urls que tengan una frecuencia de aparición mayor al promedio.

El algoritmo de evaluación que se utilizará en el segundo paso de esta metodología está basado en la metodología de detección de comunidades descrito por Bulters *et al.* en [10].

Para hacer uso de esta metodología, se implementó una interfaz de usuario (figura 5.5) en la cual el usuario puede promover una fuente potencial a una lista de fuentes a analizar al ingresar

Figura 5.5: Interfaz para promover fuentes potenciales

la url a la fuente RSS que le corresponde. Una vez que una fuente ha sido promovida hacia esta segunda lista, se empezará a recuperar documentos desde ella sin que estos sean considerados por el módulo de extracción de tópicos. Luego, se calculará de manera periódica la *linkStrength* (paso 2 de la metodología presentada en [10]) presentada en la ecuación 5.2 entre esta fuente potencial y cada una de las fuentes que son utilizadas para extraer tópicos y si la cantidad de fuentes que tienen una *linkStrength* mayor a σ (considerando como punto de origen a la fuente candidata) es mayor a un porcentaje ρ, se agregará esta fuente al conjunto de fuentes utilizadas para la extracción de tópicos a ser aprobadas por el usuario.

$$linkStrength(f, f') = w_{relev} \cdot relev + w_{reciprocity} \cdot recip + w_{cocitation} \cdot cocit \tag{5.2}$$

Con f la fuente bajo análisis y f' una fuente actualmente siendo utilizada, además cada variable de esta ecuación se define como sigue:

- $linkIntersection = f.linkSet \cap f'.linkSet$
- $recip =$ `if` $f'.linkSet$ posee un enlace a f `then` 0.5 `else` 0.0
- $cocit = \dfrac{linkIntersection.size}{f.linkSet.size}$
- $relev$: Dados los tópicos $\{t_i\}_{i \in \mathbb{N}}$ a los que pertenecen los documentos de f', la relevancia viene dada por la fracción de documentos que pueden ser asociados a uno o más de estos tópicos.

En el caso de los conjuntos de enlaces *linkSet* para una fuente se basan en todos los enlaces presentes en los documentos recuperados que no apunten a un sitio de avisaje.

Además, los pesos a utilizar en la ecuación 5.2 serán los presentados por [10]:

$$w_{relev} = 0.5, w_{co-citation} = 0.3, w_{reciprocity} = 0.2$$

Esta metodología se ve descrita en 5.4.2, la cual recibe como parámetro la fuente potencial F_p a evaluar y el valor umbral ρ que será utilizado para decidir si F_p debe ser incluida en el conjunto de

fuentes analizadas.

Algoritmo 5.4.2: Evaluación de fuentes potenciales

Data: F_p, ρ

```
1 relatedFeeds = 0;
2 actualFeeds = database.getFeeds();
3 forall feed ∈ actualFeeds do
4     if linkStrength(F_p, feed) > σ then
5         relatedFeeds++;
6     if relatedFeeds / actualFeeds.length > ρ then
7         database.addFeed(F_p);
```

Finalmente, una vez que el algoritmo califica de relevantes a las fuentes que se están analizando, el usuario debe aprobar su inclusión en el proceso de minado de documentos para la extracción de tópicos. Este proceso se realiza a través de la interfaz de usuario presente en la figura 5.6.

Figura 5.6: Interfaz para aprobar fuentes

5.5. Uso de emoticones para clasificar opiniones

Además de las mejoras al módulo de recuperación de documentos, se espera agregar cambios a la plataforma de detección de tendencias a través de la inclusión de funcionalidades extras en la metodología utilizada actualmente para la extracción de opiniones desde los documentos recuperados desde las redes sociales en base a los tópicos previamente recuperados.

Aún cuando el uso lexicones para la extracción de opiniones desde documentos opinados provee buenos resultados en la mayoría de los casos, una de sus falencias es el hecho de que estos no consideran realmente el contexto en el que la palabra en cuestión es utilizada, citando el ejemplo dado en [52], en el caso de un comentario sobre una película, *impredecible* puede ser una palabra positiva si se trata de la trama de ésta, pero si se trata de un comentario sobre la dirección de un

automóvil, el hecho de que ésta sea impredecible es un comentario negativo.

Por lo tanto, para mejorar el algoritmo de extracción de opiniones, se propone hacer uso de los emoticones presentes en los documentos de manera similiar a lo realizado en [52], sin embargo, se hará uso de estos de una manera más simple como lo presentado en [44], es decir, si se encuentra un emoticón positivo *ep* (`:-), :), =), :D`) se marcará el documento como si este tuviera polaridad positiva, y si se encuentra un emoticón negativo *en* (`:-(, :(, =(, ;(`) se determinará que el documento posee polaridad negativa. Y en cuanto a aquellos documentos que no contengan ningún emoticón se seguirá utilizando la metodología descrita en el capítulo 3.

El hecho de que un documento contenga un emoticón y sea marcado bajo una polaridad en particular sirve como medida para que el algoritmo basado en lexicones tenga indicios de qué tipo de polaridad debería tener el documento. Esto quiere decir que si un documento contiene un emoticón asociado a una polaridad p entonces se espera que el algoritmo que hace uso de lexicones para calcular un puntaje de opinión $\vec{o}$ esté compuesto de manera tal que su puntaje o_p indique aquello.

Para lograr esto, si un emoticón asociado a una polaridad p se encuentra en un documento d, se evaluará de manera inmediata la componente o_p de algoritmo basado en lexicones con un puntaje de 1 sin importar qué palabras este contenga. Esta decisión se basa en el hecho de que la plataforma de detección de tendencias no se preocupa de la intensidad de la polaridad de cada documento para visualizarlas, sólo se preocupa de qué polaridad tiene cada documento y cómo se relacionan todos los documentos de un periodo en particular con el tópico al que pertenecen. En caso de que posea ambos tipos de emoticones, se clasificará el documento en base a la cantidad de emoticones utilizados.

En base a lo expuesto en los párrafos previos, el algoritmo 5.5.1 corresponde al nuevo proceso de extracción de opiniones, donde la función `getPolarityBasedByScore` hace uso del algoritmo básico de extracción de opiniones para determinar la polaridad del documento y la función `getAppearances` devuelve la cantidad de veces que una cadena se encuentra en un documento.

Algoritmo 5.5.1: Extracción de opiniones desde documentos opinados

Data: $d, \{ep_i\}_{i\in\mathbb{N}}, \{en_i\}_{i\in\mathbb{N}}$

```
1  document = d;
2  emoticonCount = 0;
3  forall ep ∈ {ep_i}_{i∈ℕ} do
4      emoticonCount += document.getAppearances(ep);
5  forall en ∈ {en_i}_{i∈ℕ} do
6      emoticonCount -= document.getAppearances(en);
7  if emoticonCount == 0 then
8      polarity = getPolarityBasedByScore(d);
9  if emoticonCount > 0 then
10     polarity = positive;
11 if emoticonCount < 0 then
12     polarity = negative;
```

5.6. Modelo de Datos

Para poder incluir el algoritmo de selección de nuevas fuentes en el módulo de recuperación de documentos, es necesario expandir el modelo de datos para que almacene las fuentes que son visitadas inicialmente por el crawler, y además debe ser capaz de almacenar aquellas nuevas fuentes que tienen la potencialidad de ser agregadas a la lista utilizada para la recuperación de documentos. Así, se agregarán las tablas que se presentan en la figura 5.7:

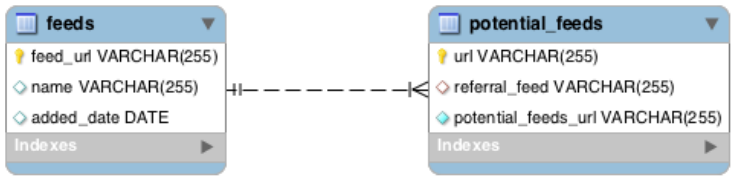

Figura 5.7: Modelo de datos para un algoritmo adaptativo de recuperación de documentos

El objetivo de la tabla **feeds** es almacenar todas aquellas fuentes que en un momento dado del tiempo son utilizadas por el crawler de recuperación de documentos. Inicialmente sólo se necesita la url desde donde se obtendrá la lista de documentos con sus respectivas urls (en el caso de una fuente RSS, esta URL también contendrá los documentos), el nombre que se le asigna a dicha fuente y la

fecha en que esta fue agregada a la base de datos.

Así mismo, la tabla **potential_feeds** cumple el rol de almacenar posibles fuentes que el modelo adaptativo de recuperación de documentos se encargará de validar o desechar según sea el caso.

Además, es necesario cambiar el modelo de datos de las fuentes

Con respecto a las otras componentes del modelo de datos, se considera innecesario realizar cambios, ya que estas no se ven afectadas por el rediseño de arquitectura de la plataforma.

En cuanto a la implementación de una capa de persistencia en el sistema, debido a las restricciones presupuestarias existentes, la naturaleza de los datos, la frecuencia de acceso y la manera en que estos serán trabajados, se recomienda utilizar el motor de base de datos MySQL. Para aumentar la escalabilidad y seguridad de la plataforma, es recomendable disponer de réplicas de sólo lectura de la base de datos para que sean utilizadas por las aplicaciones orientadas a usuarios externos.

Capítulo 6

Evaluación de los cambios propuestos

En este capítulo se realizará una evaluación de los cambios realizados a la plataforma, es decir, el algoritmo de selección de fuentes potenciales, el uso de emoticones en el algoritmo de minado de opiniones y finalmente las mejoras propuestas a la plataforma en su totalidad.

En el caso de los nuevos algoritmos primero se diseña un experimento sobre un conjunto de datos construido de manera semi automática y luego se evalúan utilizando métricas conocidas. Para evaluar la plataforma en si, se realizan tests de carga comparando el rendimiento de la plataforma con y sin los cambios.

6.1. Algoritmo selección de fuentes

A continuación se presenta la evaluación realizada al algoritmo de calificación de relevancia y selección automática de fuentes de documentos.

6.1.1. Diseño del experimento

Se definió manualmente un conjunto inicial de 20 fuentes en formato RSS las cuales fueron utilizadas para recolectar documentos a lo largo de 15 cantidad de dias días. Todas las fuentes seleccionadas poseen documentos en inglés y están relacionadas con el área de la tecnología con todo tipo de temas en discusión: productos, servicios, noticias sobre el mercado accionario, etc..

Para cada documento recuperado desde estas fuentes se almacenaron los siguientes datos: contenido original en formato *html*, fecha de publicación, enlace original, fuente y además el momento en el que el documento fue recuperado.

Luego, se ejecutó el algoritmo de selección de fuentes potenciales sobre todos los documentos recuperados con una serie de parámetros para determinar distintos conjuntos de fuentes potenciales. Cada fuente potencial detectada que tuviese disponible sus contenidos para ser minados fue calificada como relevante o no relevante de manera manual. El criterio utilizado para determinar si una fuente

potencial es relevante o no es si el tipo de contenido publicado pertenece al mismo dominio (en este caso tecnología) que está siendo analizado.

Finalmente, para cada una de las fuentes potenciales detectadas se obtuvo su punto de acceso en formato RSS y se procedió a minar sus contenidos a lo largo de 10 días para posteriormente ejecutar el algoritmo de calificación de relevancia para cada una de ellas en base a los documentos minados en este periodo.

6.1.2. Criterio de Evaluación

Este experimento tiene como objetivo medir la efectividad del modelo de calificación de relevancia de fuentes. Para ello, se utilizó como medida de evaluación el nivel de relación entre la fuente siendo evaluada y las fuentes que están siendo previamente procesadas por el sistema. Para determinar el nivel de relación entre fuentes se realizó un análisis manual de las temáticas discutidas por cada una de las fuentes y por aquella rama de la información disponible en la web se espera que la plataforma esté analizando.

Para evaluar el algoritmo de selección de fuentes potenciales se utilizó la métrica de *precision*, y bajo esta misma temática se define la métrica *precision at k* como la *precision* utilizando como parámetro del algoritmo de selección de fuentes potenciales el valor k. En este experimento la *precision* y el *recall* se definen como:

$$precision = \frac{\text{Fuentes potenciales relevantes seleccionadas para ser evaluadas}}{\text{Fuentes potenciales seleccionadas para ser evaluadas}} \tag{6.1}$$

$$recall = \frac{\text{Fuentes potenciales relevantes seleccionadas para ser evaluadas}}{\text{Fuentes potenciales relevantes}} \tag{6.2}$$

Además, para el algoritmo de calificación de relevancia se utilizaron las siguientes métricas:

$$precision = \frac{\text{Fuentes bajo análisis relevantes calificadas como relevantes}}{\text{Fuentes bajo análisis calificadas como relevantes}} \tag{6.3}$$

$$recall = \frac{\text{Fuentes bajo análisis relevantes calificadas como relevantes}}{\text{Fuentes bajo análisis que son relevantes}} \tag{6.4}$$

6.1.3. Resultados y Discusiones

Una vez ejecutado el experimento descrito, incluyendo la evaluación manual de las nuevas fuentes detectadas y el cálculo de las métricas de evaluación para la primera parte del experimento se dan a conocer los resultados de estos en las tablas 6.1 y 6.2. En total se obtuvieron cerca de 12000

documentos y 32000 enlaces a partir de los documentos minados desde las fuentes iniciales de las cuales corresponden a 1500 fuentes únicas. Estos enlaces se distribuyen como sigue:

Tipo	Cantidad
Fuentes bajo análisis	27740
Archivos	263
Redes Sociales (YouTube, Facebook, Twitter, etc.)	397
Streaming Web	45
Sitios Gubernamentales	122
Sitios Enciclopédicos (Wikipedia, IMDB, etc.)	234
Agregadores de Feeds	192
Sitios de Universidades	33
Otros	2752

Cuadro 6.1: Distribución de los tipos de enlace

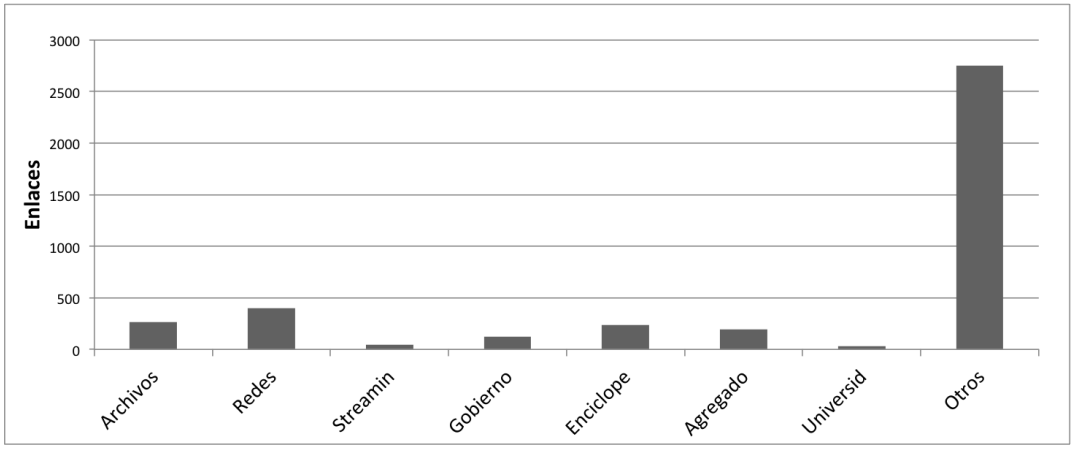

Figura 6.1: Distribución de los tipos de enlace

De estos, sólo los enlaces presentes en la categoría de otros son susceptibles de ser incluidos como fuentes, sin embargo, el algoritmo no es capaz de distinguir entre estos a excepción de las fuentes bajo análisis, sin embargo, es común que los documentos recuperados a través del formato RSS tengan enlaces a sus sitios a través de permalinks relacionados a sus fuentes sindicables, lo cual es mejorable debido a que estas urls contienen el nombre de dominio en ellas.

El promedio de aparición entre todos los enlaces recuperados por la primera etapa del algoritmo es de 2.4 con lo que 180 fuentes son consideradas para su inclusión. De estas, 82 corresponden a sitios web de servicios o marcas y 101 a blogs, sitios de noticias o similares donde 61 de estos poseen en su mayoría artículos de tecnología, con el resto publicando noticias de interés general que aún pueden ser evaluadas por el segundo paso del modelo.

Por lo tanto, considerando las fuentes extraídas, el *recall* de este modelo corresponde a 0.35. Si bien el recall es bajo debido a que la cantidad de fuentes posibles tiende a crecer a medida que el

Enlaces	Fuentes
1	1061
2	232
3 - 10	143
11 - 100	32
100 o más	5
Fuentes bajo análisis	20

Cuadro 6.2: Cantidad de fuentes en relación al número de enlaces encontrados

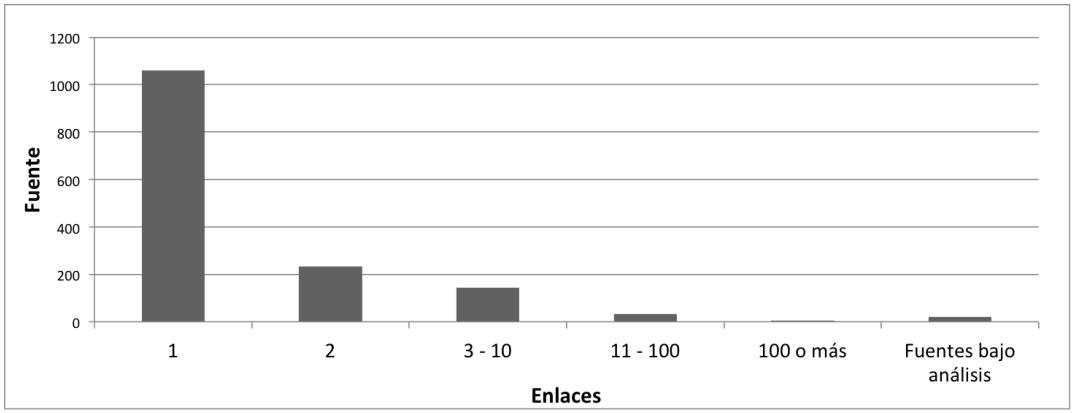

Figura 6.2: Cantidad de fuentes en relación al número de enlaces encontrados

periodo de evaluación aumenta, como la mayoría de estas fuentes sólo poseen sólo uno o dos enlaces se puede considerar que aún cuando fuesen incluidas entre las fuentes potenciales, el algoritmo de evaluación las descartaría debido a la poca conexión existente con las fuentes actuales, por lo que un recall bajo no necesariamente implica que se pierda información valiosa.

La precisión de esta parte del algoritmo corresponde a 0.56, la que se debe principalmente a la gran cantidad de sitios web de productos o servicios producto de que muchas noticias del área de la tecnología hacen referencia a compañías, sea en relación a sus productos o servicios nuevos que estén lanzando.

Para analizar la segunda parte del experimento, es decir, la evaluación de las fuentes seleccionadas previamente se agregaron los vínculos RSS para las fuentes consideradas en la parte anterior. Posteriormente se evaluaron estas fuentes para distintos valores de ρ y se obtuvieron los siguientes resultados presentados en la tabla 6.3

En la tabla 6.3 se observa que ni el recall ni la precision del algoritmo pueden ser calculados si es que se utiliza un valor de ρ muy alto ya que ninguna fuente es calificada como relevante. Además, la precisión del algoritmo aumenta a medida que el valor de ρ aumenta, ya que la fuente posee más relación con el tema que se está analizando (tecnología en este experimento), y por otro lado, el recall disminuye ya que la cantidad de fuentes seleccionadas es mucho menor debido al aumento en

ρ	Precision	Recall
0.3	0.32	0.51
0.5	0.53	0.42
0.6	0.57	0.30
0.8	-	-

Cuadro 6.3: *Precision* y *Recall* para distintos valores de ρ

las restricciones.

6.2. Algoritmo de minado de opiniones

En esta sección se presenta el experimento utilizado para evaluar los cambios en el algoritmo de minado de opiniones y los resultados de éste.

6.2.1. Diseño del experimento

Para evaluar el algoritmo de minado de opiniones se procedió a utilizar un corpus de tweets previamente clasificados de manera manual como positivos, negativos u objetivos que contuviese tweets con emoticones, el cual fue confeccionado manualmente. Luego, para cada tweet del corpus, se determinó la polaridad con los que son clasificados utilizando el algoritmo de minado de opiniones original y la variación de este que hace uso de los emoticones y se compararon estos resultados con la clasificación determinada previamente de manera manual.

6.2.2. Criterio de Evaluación

El objetivo detrás de este experimento es determinar si los cambios propuestos en relación al uso de emoticones que fueron realizados al algoritmo de minado de opiniones son un aporte significativo o no y por consiguiente verificar o refutar la hipótesis propuesta inicialmente. Como métrica de evaluación se utilizarán el *recall* y la *precision* que se definen como sigue:

$$\textit{recall at } p = \frac{\#\text{ de tweets clasificados correctamente con polaridad } p \text{ por el algoritmo}}{\#\text{ de tweets del corpus clasificados originalmente con polaridad } p} \tag{6.5}$$

$$\textit{precision at } p = \frac{\#\text{ de tweets clasificados correctamente con polaridad } p \text{ por el algoritmo}}{\#\text{ de tweets clasificados por el algoritmo con polaridad } p} \tag{6.6}$$

6.2.3. Resultados y Discusiones

En la tabla 6.4 se presentan los resultados del experimento planteado, estos consisten de la precision y el recall para el modelo de extracción de opiniones con emoticones (P' y R') y la versión

original (P y R) para cada una de las polaridades.

Polaridad	P'	P	R'	R
Positiva	0.67	0.6	0.62	0.59
Objetiva	0.53	0.53	0.49	0.49
Negativa	0.64	0.61	0.6	0.58

Cuadro 6.4: Precision y recall por polaridad del algoritmo con y sin emoticones

Se puede observar en la tabla 6.4 que no hubo cambio a la hora de detectar documentos opinados de manera objetiva lo que se debe a que estos no son acompañados por emoticones, sólo por la información que se desea mostrar; en cambio, sí se encontró un cambio tanto en las polaridades positivas como en las negativas, con el cambio más relevante en la polaridad positiva debido a que se puede expresar una opinión positiva sobre un hecho tan sólo utilizando un emoticon positivo como `:D` o `:)`, sin embargo, a la hora de demostrar opiniones negativas, estas casi siempre vienen acompañadas de un calificativo que da a conocer cuál es la razón del descontento, sea este que el usuario extraña algo en particular, lo encuentra desagradable, le provoca tristeza o cualquier otro.

6.3. Rediseño de la plataforma

A continuación se dan a conocer los experimentos que se realizarán para ver las mejoras que tuvo el sistema en comparación a la implementación previa.

6.3.1. Diseño del experimento

Se realizaron múltiples pruebas que buscaban comprobar los beneficios que se buscaban lograr tras el rediseño de la plataforma. Estas pruebas consisten en:

1. Cargar una cantidad creciente de fuentes y analizar la carga en el sistema. Además, se desea ver la cantidad máxima de fuentes que pueden ser procesadas por el sistema por unidad de tiempo.

2. Crear una aplicación orientada a usuarios simple y analizar el impacto en el rendimiento general del sistema en base los datos recibidos con una cantidad creciente de peticiones. Esta aplicación será desarrollada en PHP y recibirá peticiones de lectura y de escritura, tras lo cual realizará llamadas aleatorias de lectura o escritura de datos para así evitar que el *caché* de la base de datos afecte los resultados.

3. Minar sólo fuentes que posean documentos de gran volumen, con el fin de determinar el rendimiento del sistema en general para comparar la plataforma rediseñada con la original. Para

esto se crearon 8 *Micro Instances* en Amazon EC2 con un servidor Apache que cada una publican 7 fuentes en formato RSS formada a partir de una base de datos con documentos aleatorios generados previamente de un tamaño aproximado de 1MB de texto.

Para realizar estas pruebas se hizo uso de múltiples *Standard Small Instance* de *Amazon EC2*[1]. Cada una de estas instancias posee 1.7GB de memoria RAM y un poder de procesamiento de 1 EC2 CPU (cada unidad de EC2 CPU es equivalente a un procesador Intel Xeon E5430 a 2.66GHz). La razón de utilizar instancias en Amazon EC2 es la facilidad de crear nuevas máquinas y la seguridad de obtener múltiples máquinas de iguales características, lo que permite obtener resultados no sesgados a la hora de comparar cada plataforma.

Cada una de las instancias corre sobre el sistema operativo `Ubuntu 12.04.1 LTS`. En el caso de los servidores de aplicaciones orientadas a usuarios externos se instalará *Apache* versión `2.2.22` y *PHP* versión `5.3.10`. En el caso de los servidores de bases de datos se instalará solamente *MySQL Server* versión `5.5.24`. Finalmente, para las máquinas que alojan los distintos módulos de la plataforma es necesario instalar una versión de Java Development Kit, en particular, se instalará la versión `1.6.0_24` de *OpenJDK*[2]

6.3.2. Criterio de Evaluación

El criterio de evaluación a utilizar en cada una de las pruebas tiene que ver con la carga en el sistema, es el porcentaje de uso de procesador y de memoria RAM. En el caso de pruebas que busquen determinar los límites de la nueva y la antigua plataforma el criterio de evaluación será la cantidad de objetos que pueden ser procesados o recuperados a lo largo de una unidad de tiempo fija que en el caso de estos experimentos será de 30 minutos. Los experimentos serán ejecutados a lo largo de un día y se obtendrán los promedios para cada una de las métricas que están siendo utilizadas.

6.3.3. Resultados y Discusiones

Experimento #1

En la tabla 6.5 se presentan los datos obtenidos del experimento para el módulo de recuperación de documentos: uso promedio de RAM en la plataforma rediseñada y en la antigua plataforma, $\bar{R}'$ y $\bar{R}$ respectivamente; y uso promedio de CPU $\bar{C}'$ y $\bar{C}$.

[1] *Amazon Elastic Compute Cloud* es un servicio web que provee capacidad computacional variable en la nube diseñado para realizar aplicaciones que utilicen múltiples servidores o satisfagan una gran cantidad de clientes de manera sencilla.

[2] `http://openjdk.java.net/`

Fuentes	$\bar{R}'$	$\bar{R}$	$\bar{C}'$	$\bar{C}$
30	32MB	34MB	0.2	0.6
50	32MB	47MB	0.3	0.8
80	33MB	53MB	0.3	1.2
130	36MB	61MB	0.4	1.7
210	38MB	-	0.5	-

Cuadro 6.5: Datos obtenidos en base a la cantidad de fuentes para el experimento 3.1

En la tabla 6.5 se puede observar que el rendimiento en el módulo rediseñado es mejor tanto en uso de CPU como en uso de memoria RAM: en el caso del rediseño el uso de ambos se mantiene casi constante a medida que aumentan las fuentes, en cambio en la versión anterior aumentaba sostenidamente con la cantidad de fuentes. Además, la versión antigua del módulo de recuperación de documentos no era capaz de recolectar 210 fuentes por hora ya que cada fuente tomaba más de 17 segundos en promedio.

Experimento #2

En la tabla 6.6 se presentan los resultados del experimento al igual que el experimento anterior. Se presentarán las métricas en relación al servidor de bases de datos utilizado en base a las peticiones por minuto realizadas (PPM) considerando el uso normal de la plataforma.

PPM	$\bar{R}'$	$\bar{C}'$
0	147MB	0.1
20	147MB	0.1
30	147MB	0.1
50	147MB	0.2
80	147MB	0.3

Cuadro 6.6: Datos obtenidos en base a la cantidad de peticiones por minuto

El uso de memoria RAM en el servidor de base de datos no cambia a medida que la cantidad de peticiones aumenta ya que el motor de base de datos usa un nivel de memoria relacionado a la cantidad de información que debe ser almacenada en el caché y esta crece a medida que el tamaño de la base de datos aumenta, la cual, en el caso del experimento, no tiene un tamaño tal que pueda provocar cambios en el uso de memoria RAM por parte de este. En el caso del CPU este varía levemente a medida que aumentan la cantidad de consultas debido a que debe hacer uso de la información almacenada en la memoria RAM para responder las consultas realizadas, y por consiguiente, a medida que se realicen más consultas, más CPU será utilizado. A pesar de esto, se puede observar que el sistema se comporta de manera eficiente aún cuando se reciban peticiones

desde usuarios externos a través de una API.

Experimento #3

En la tabla 6.7 se presentan los datos obtenidos del experimento: uso promedio de RAM en la plataforma rediseñada y en la antigua plataforma, $\bar{R}'$ y $\bar{R}$ respectivamente; y uso promedio de CPU $\bar{C}'$ y $\bar{C}$. Estos se calculan promediando el uso de RAM y CPU de cada uno de los módulos del sistema.

Fuentes	$\bar{R}'$	$\bar{R}$	$\bar{C}'$	$\bar{C}$
7	110MB	130MB	0.2	0.3
10	112MB	135MB	0.2	0.3
20	119MB	147MB	0.3	0.4
30	125MB	162MB	0.3	0.8
50	134MB	179MB	0.5	1.1

Cuadro 6.7: Datos obtenidos en base a la cantidad de fuentes minadas

En la tabla 6.7 se puede observar que el uso de CPU en la plataforma antigua crecía considerablemente a medida que el tamaño y la cantidad de las fuentes crece, lo que no sucede tan drásticamente en el caso de la plataforma rediseñada debido a la distribución de las tareas y la paralelización de algunos procesos que permiten que uno o más módulos estén en descanso la mayor parte del tiempo.

6.4. Percepción del usuario

Una vez realizados los cambios en la plataforma, se consultó con el usuario sobre la percepción que este tuvo de los cambios realizados a la plataforma. A continuación se exponen los puntos relevantes de los comentarios expuestos por este.

Desde el punto de vista de negocio, la inclusión de una herramienta para incluir nuevas fuentes a la plataforma es muy beneficioso para la empresa, ya que previamente este proceso se realizaba de manera esporádica y requería muchas horas hombre para realizarlo, lo que conllevaba una merma en la cantidad de información que la plataforma puede abarcar y por lo tanto en la calidad del contenido que puede ser previsto por ella.

Si bien las mejoras realizadas al algoritmo de minado de opiniones no impactan directamente a Duam S.A. en términos de horas hombre invertidas en el funcionamiento de la plataforma, que esta ofrezca información más acertada permite que los servicios que desear promocionar la empresa a través de hacer uso de esta herramienta sean más confiables y por lo tanto, sean más útiles para el usuario final.

Finalmente, el rediseño de la plataforma permite el crecimiento a futuro del servicio ofrecido y la disminución en el uso de recursos, lo que disminuye los costos del uso de la plataforma y permite la inclusión de una gran cantidad de fuentes para lograr que la plataforma sea utilizada sin problemas para realizar múltiples investigaciones de inteligencia de mercado simultáneamente.

Capítulo 7

Conclusiones y Trabajo Futuro

El patrón *Service Oriented Architecture* que fue propuesto para la arquitectura de la plataforma permitió que cada componente del sistema interactuara de manera independiente junto con dar la flexibilidad necesaria para que estos sean paralelizables. Aún cuando el algoritmo de minado de tópicos no es paralelizable y por lo tanto este módulo no es capaz de escalar al igual que el resto del sistema, debido a que este proceso sólo sucede una vez en cada iteración de la plataforma no representa un cuello de botella que merme el rendimiento real del sistema.

Que una plataforma posea la capacidad de escalar tanto horizontal como verticalmente permite que sin importar la cantidad de elementos siendo procesados los tiempos de respuesta y los recursos utilizados sean óptimos, reduciendo así costos reales y permitiendo realizar mejoras en otros ámbitos, como realizar mejores herramientas de visualización o mejorar los algoritmos de recuperación de información.

La arquitectura propuesta para las aplicaciones de usuarios externos resulto ser bastante sencilla de utilizar, tal como se comprobó a la hora de realizar una pequeña API que pusiera en práctica este cambio. El permitir que la plataforma ponga a disposición de otros usuarios o de otras aplicaciones la información que posee abre las posibilidades sobre lo que se puede lograr al montar una plataforma de detección de tendencias como esta. Por ejemplo, se pueden crear múltiples formas de visualizar los datos o crear sitios relacionados que contengan sólo un subconjunto de la información disponible.

Por otro lado, las hipótesis planteadas al comienzo de esta memoria de título fueron comprobadas, aún cuando los valores obtenidos en los experimentos no fueron sobresalientes, permiten comprender que los enfoques escogidos tienen resultados útiles en la práctica y por lo tanto son una mejora considerable sobre el estado previo de la plataforma. Si bien la metodología de inclusión de nuevas fuentes no era considerada en la versión antigua de la plataforma, el permitir que el administrador de esta tenga la posibilidad de escoger fuentes de una lista que tenga una probabilidad considerable de ser un aporte real a la plataforma es de alto valor, como también el hecho de poder evaluarlas y

determinar posteriormente si incluirlas realmente agregan nueva información al sistema o no.

En el caso de las modificaciones al algoritmo de minado de opiniones, el incluir los emoticones permite obtener polaridades de documentos que de otra manera hubiesen sido clasificados de manera errónea como documentos objetivos. Sería deseable incluir además otras características de los documentos opinionados provenientes de Twitter como los hashtags, las menciones, etc., ya que se espera que a medida que más metadata es utilizada mejores serán los resultados de los algoritmos de minado de opiniones.

Referencias

[1] F. Alvanaki, M. Sebastian, K. Ramamritham, and G. Weikum. Enblogue: emergent topic detection in web 2.0 streams. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 1271–1274, Athens, Greece, 2011. ACM.

[2] G. M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, AFIPS '67 (Spring), pages 483–485, Atlantic City, New Jersey, 1967. ACM.

[3] N. Archak, A. Ghose, and P. G. Ipeirotis. Show me the money!: deriving the pricing power of product features by mining consumer reviews. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 56–65, San Jose, California, USA, 2007. ACM.

[4] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York., 1999.

[5] T. Berners-Lee. Short summary of the world wide web project. `https://groups.google.com/forum/?fromgroups=#!msg/alt.hypertext/eCTkkOoWTAY/bJGhZyooXzkJ`, 1991. Visitada el 17 de Octubre del 2012.

[6] D. M. Blei and J. D. Lafferty. Correlated topic models. In *NIPS*, 2005.

[7] D. M. Blei and J. D. Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 113–120, Pittsburgh, Pennsylvania, 2006. ACM.

[8] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, Mar. 2003.

[9] S. Brody and N. Diakopoulos. Cooooooooooooooollllllllllllll!!!!!!!!!!!!!! using word lengthening to detect sentiment in microblogs. In *Proceedings of the 2011 Conference on Empirical Methods in*

Natural Language Processing, pages 562–570, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

[10] J. Bulters and M. de Rijke. Discovering weblog communities: A content- and topology-based approach. In *Proceedings of the International Conference on Weblogs and Social Media*, ICWSM 07', pages 211–214, Boulder, Colorado, USA, 2007. AAAI.

[11] M. Craig, O. Iadh, and S. Ian. Overview of the trec 2007 blog track. In *Proceeddings of the 16th Text REtrieval Conference*, TREC '07, 2007.

[12] I. P. Cvijikj and F. Michahelles. Monitoring trends on facebook. In *Proceedings of the 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, DASC '11, pages 895–902, Sydney, Australia, 2011. IEEE Computer Society.

[13] T. Damer. *Attacking faulty reasoning: a practical guide to fallacy-free arguments.* Wadsworth/Cengage Laerning, Australia Belmont, CA, 2009.

[14] S. David and T. J. Pinch. Six degrees of reputation: The use and abuse of online review and recommendation systems. *First Monday*, July 2006. Special Issue on Commercial Applications of the Internet.

[15] M. Dikaiakos, A. Stassopoulou, and L. Papageorgiou. An investigation of web crawler behavior: characterization and metrics. *Computer Communications*, 28(8):880–897, 2005.

[16] P. Domingos. Mining social networks for viral marketing. *IEEE Intelligent Systems*, 20(1):80–82, 2005.

[17] A. Esuli and F. Sebastiani. SentiWordNet: A publicly available lexical resource for opinion mining. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, LREC '06, pages 417–422, Genoa, Italy, 2006. European Language Resources Association (ELRA).

[18] A. Funk, Y. Li, H. Saggion, K. Bontcheva, and C. Leibold. Opinion analysis for business intelligence applications. In *Proceedings of the first international workshop on Ontology-supported business intelligence*, OBI '08, pages 3:1–3:9, Karlsruhe, Germany, 2008. ACM.

[19] D. Garlan, F. Bachmann, J. Ivers, J. Stafford, L. Bass, P. Clements, and P. Merson. *Documenting Software Architectures: Views and Beyond.* Addison-Wesley Professional, 2nd edition, 2010.

[20] A. Ghose, P. Ipeirotis, and A. Sundararajan. Opinion mining using econometrics: A case study on reputation systems. In *Proceedings of the Association for Computational Linguistics*, pages 416–423, Prague, Czech Republic, 2007. Association for Computational Linguistics.

[21] B. Gu, P. Konana, A. Liu, B. Rajagopalan, and J. Ghosh. Predictive value of stock message board sentiments. *McCombs Research Paper No. IROM-11-06*, 2006.

[22] V. Hatzivassiloglou and J. Wiebe. Effects of adjective orientation and gradability on sentence subjectivity. In *Proceedings of the 18th conference on Computational linguistics - Volume 1*, volume 1 of *COLING '00*, pages 299–305. Association for Computational Linguistics, 2000.

[23] M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 168–177, Seattle, Washington, USA, 2004. ACM.

[24] L. Huang et al. A survey on web information retrieval technologies. *Computer Science Departament, State University of New York*, 2000.

[25] X. Jin, Y. Li, T. Mah, and J. Tong. Sensitive webpage classification for content advertising. In *Proceedings of the 1st International Workshop on Data Mining and Audience Intelligence for Advertising*, ADKDD '07, pages 28–33, San Jose, California, USA, 2007. ACM.

[26] N. Jindal and B. Liu. Opinion spam and analysis. In *Proceedings of the International Conference on Web Search and Web Data Mining*, WSDM '08, pages 219–230, Palo Alto, California, USA, 2008. ACM.

[27] A. Kennedy and D. Inkpen. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 22:2006, 2006.

[28] E. Kouloumpis, T. Wilson, and J. Moore. Twitter sentiment analysis : The good the bad and the omg ! *Artificial Intelligence*, 70(2):538–541, 2011.

[29] Y. Lin, H. Sundaram, Y. Chi, J. Tatemura, and B. Tseng. Discovery of blog communities based on mutual awareness. In *Proceedings of the 3rd Annual Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics, 15th World Wide Web Conference*, May 2006.

[30] B. Liu. Sentiment analysis and subjectivity. In N. Indurkhya and F. J. Damerau, editors, *Handbook of Natural Language Processing, Second Edition*. CRC Press, Taylor and Francis Group, Boca Raton, Florida, USA, 2010. ISBN 978-1420085921.

[31] B. Liu, M. Hu, and J. Cheng. Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, pages 342–351, Chiba, Japan, 2005. ACM.

[32] Y. Liu, X. Huang, A. An, and X. Yu. Arsa: a sentiment-aware model for predicting sales performance using blogs. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 607–614, Amsterdam, The Netherlands, 2007. ACM.

[33] R. C. Martin. *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2003.

[34] M. Mathioudakis and N. Koudas. Twittermonitor: trend detection over the twitter stream. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 1155–1158, Indianapolis, Indiana, USA, 2010. ACM.

[35] S. Matsumoto, H. Takamura, and M. Okumura. Sentiment classification using word subsequences and dependency sub-trees. *Advances in Knowledge Discovery and Data Mining*, pages 21–32, 2005.

[36] R. Mihalcea, C. Banea, and J. Wiebe. Learning multilingual subjective language via cross-lingual projections. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 976–983, Prague, Czech Republic, 2007. Association for Computational Linguistics.

[37] G. Mishne and N. Glance. Predicting movie sales from blogger sentiment. In *In AAAI 2006 Spring Symposium on Computational Approaches to Analysing Weblogs*, CAAW '06, pages 301–304. AAAI Press, 2006.

[38] A. Mislove. *Online Social Networks: Measurement, Analysis, and Applications to Distributed Information Systems*. PhD thesis, Rice University, Department of Computer Science, May 2009.

[39] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 29–42, San Diego, California, USA, 2007. ACM.

[40] T. Mullen and N. Collier. Sentiment analysis using support vector machines with diverse information sources. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, volume 4 of *EMNLP '04*, pages 412–418. ACL, 2004.

[41] V. Ng, S. Dasgupta, and S. Arifin. Examining the role of linguistic knowledge sources in the automatic identification and classification of reviews. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 611–618. Association for Computational Linguistics, 2006.

[42] B. O'Connor, R. Balasubramanyan, B. Routledge, and N. Smith. From tweets to polls: Linking text sentiment to public opinion time series. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*, ICWSM '10, pages 122–129. AAAI Press, 2010.

[43] B. Ohana and B. Tierney. Sentiment classification of reviews using sentiwordnet. *Discovery*, page 13, 2009.

[44] A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In N. C. C. Chair), K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner, and D. Tapias, editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation*, LREC '10, Valletta, Malta, May 2010. European Language Resources Association (ELRA).

[45] G. Paltoglou, S. Gobron, M. Skowron, M. Thelwall, and D. Thalmann. Sentiment analysis of informal textual communication in cyberspace. In *Proceedings of ENGAGE 2010*, ENGAGE '10, pages 13–25, Zermatt, Switzerland, 2010.

[46] B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, Jan. 2008.

[47] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, EMNLP '02, pages 79–86, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[48] G. Pant, P. Srinivasan, and F. Menczer. Crawling the web. In *In Web Dynamics: Adapting to Change in Content, Size, Topology and Use. Edited by M. Levene and A. Poulovassilis*, pages 153–178. Springer-Verlag, 2004.

[49] W. Parrott. *Emotions in Social Psychology: Key Readings*. Key Readings in Social Psychology. Taylor & Francis, 2000.

[50] A.-M. Popescu and O. Etzioni. Extracting product features and opinions from reviews. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural*

Language Processing, HLT '05, pages 339–346, Vancouver, British Columbia, Canada, 2005. Association for Computational Linguistics.

[51] M. F. Porter. An algorithm for suffix stripping. In K. Sparck Jones and P. Willett, editors, *Readings in information retrieval*, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, California, USA, 1997.

[52] J. Read. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *Proceedings of the ACL Student Research Workshop*, ACLstudent '05, pages 43–48, Ann Arbor, Michigan, 2005. Association for Computational Linguistics.

[53] L. Sarmento, P. Carvalho, M. Silva, and E. de Oliveira. Automatic creation of a reference corpus for political opinion mining in user-generated content. In *Proceedings of the 1st International CIKM Workshop on Topic-Sentiment Analysis for Mass Opinion*, CIKM '09, pages 29–36, Hong Kong, China, 2009. ACM.

[54] V. Sehgal and C. Song. Sops: stock prediction using web sentiment. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, ICDMW '07, pages 21–26, Omaha, Nebraska, USA, 2007. IEEE Computer Society.

[55] C. Shearer. The CRISP-DM model: The new blueprint for data mining. *Journal of Data Warehousing*, 2000.

[56] M. Silva, P. Carvalho, L. Sarmento, E. de Oliveira, and P. Magalhaes. The design of optimism, an opinion mining system for portuguese politics. *New Trends in Artificial Intelligence: Proceedings of EPIA*, pages 12–15, 2009.

[57] M. Steyvers and T. Griffiths. *Latent Semantic Analysis: A Road to Meaning*, chapter Probabilistic topic models. Laurence Erlbaum, 2007.

[58] J. Tatemura. Virtual reviewers for collaborative exploration of movie reviews. In *Proceedings of the 5th International Conference on Intelligent User Interfaces*, IUI '00, pages 272–275, New Orleans, Louisiana, USA, 2000. ACM.

[59] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter. Phoaks: a system for sharing recommendations. *Commun. ACM*, 40(3):59–62, Mar. 1997.

[60] R. M. Tong. An operational system for detecting and tracking opinions in on-line discussion. In *Proceedings of the Workshop on Operational Text Classification*, SIGIR '01, 2001.

[61] P. 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, Philadelphia, Pennsylvania, 2002. Association for Computational Linguistics.

[62] B. Walsh. Markov chain monte carlo and gibbs sampling, lecture notes for eeb 581, version 26, April 2004.

Apéndices

A . Listado de *stop-words* en español

a	a base de	a pesar de	a pesar de que
abajo	aca	acá	ademas
además	ahi	ahora	ahí
al	al contrario	al parecer	al respecto
al respecto	algun	alguna	algunas
alguno	algunos	algún	alla
alli	allá	allí	ambas
ambos	ante	anterior	antes
aparte	aquel	aquellas	aquello
aquellos	aqui	aquél	aquélla
aquéllas	aquéllos	aquí	arriba
asi	asimismo	así	atras
atrás	aun	aunque	aún
bajo	bastante	bien	bueno
buenos	cada	casi	caso
casos	cierta	ciertas	cierto
ciertos	como	con	con base en
con respecto a	conseguimos	conseguir	consigo
consigue	consiguen	consigues	consiguiente
cosa	cosas	creo	cual
cuales	cualquier	cualquiera	cuando
cuanta	cuantas	cuanto	cuantos
cuestion	cuestión	cuya	cuyas
cuyo	cuyos	cuál	cuáles
cuándo	cuánta	cuántas	cuánto

cuántos	cómo	dado que	de
de	de esa forma	de esta forma	de hecho
de igual forma	de la misma forma	de tal forma	decir
del	demás	dentro	desde
despues	después	dice	dicen
dicho	donde	dónde	e
el	ellas	ellos	en
en base a	en cambio	en conclusion	en conclusión
en concreto	en consecuencia	en definitiva	en efecto
en el fondo	en otras palabras	en particular	en primer lugar
en realidad	en relacion a	en relacion con	en relación a
en relación con	en resumen	en segundo lugar	en suma
en síntesis	en tercer lugar	encima	ende
entonces	entre	era	eramos
eran	eras	eres	es
esa	esas	ese	eso
esos	esta	estaba	estado
estais	estamos	estan	estas
este	esto	estos	estoy
está	están	fin	fue
fuera	fueron	fui	fuimos
ha	hace	haceis	hacemos
hacen	hacer	haces	hacia
hago	han	harta	hartas
harto	hartos	hasta	hay
i	igualmente	incluso	ir
la	largo	las	le
les	lo	los	luego
manera	mas	me	mediante
mejor	mi	mientras	mio
mis	misma	mismas	mismo
mismos	modo	mucha	muchas
mucho	muchos	muy	más

mí	mío	ni	ningun
ninguna	ningunas	ningunos	ningún
no	no obstante	nos	nosotros
o	o sea	obvia	obvias
obvio	obvios	ojala	ojalá
ojalá	otra	otras	otro
para	pero	pese a	poca
pocas	poco	pocos	podeis
podemos	poder	podria	podriais
podriamos	podrian	podrias	podría
podríamos	podrían	podrías	por
por conclusion	por conclusión	por dicha razon	por dicha razón
por dicho motivo	por ejemplo	por el contrario	por esa razon
por esa razón	por ese motivo	por otra parte	por otro
por otro lado	por que	por qué	por su parte
por supuesto	por tal motivo	por tal razón	por un lado
por una parte	por último	porque	porqué
posteriormente	primero	puede	pueden
puedo	pues	puesto	que
que	quien	quiza	quizas
quizá	quizás	quién	qué
respecto a	respecto de	sabe	sabeis
sabemos	saben	saber	sabes
se	segun	segundo	según
según parece	sentido	ser	si
siendo	siendo	similar	sin
sin duda	sin embargo	sin lugar a dudas	sobre
sobre la base de	sois	sola	solamente
solas	solo	solos	somos
son	soy	su	super
sus	sí	sólo	súper
tal	tal vez	tales	también
tan	tanta	tantas	tanto

tantos	te	teneis	tenemos
tener	tengo	tercer	tercero
ti	tiempo	tiene	tienen
toda	todas	todo	todos
tras	través	tus	tuyo
u	ultimo	un	una
unas	uno	unos	usar
ustedes	va	vais	vamos
van	vaya	vemos	ven
ver	vez	visto	vosotras
vosotros	voy	vía	y
ya	yo	él	éramos
ésa	ése	ésos	ésta
éstas	éste	éstos	último

Cuadro 7.1: Listado de *stop-words* en español

B . Listado de *stop-words* en inglés

'll	've	a	able	about
above	abst	accordance	according	accordingly
across	act	actually	added	adj
adopted	affected	affecting	affects	after
afterwards	again	against	ah	all
almost	alone	along	already	also
although	always	am	among	amongst
an	and	announce	another	any
anybody	anyhow	anymore	anyone	anything
anyway	anyways	anywhere	apparently	approximately
are	aren	arent	arise	around
as	aside	ask	asking	at
auth	available	away	awfully	b
back	be	became	because	become
becomes	becoming	been	before	beforehand
begin	beginning	beginnings	begins	behind

being	believe	below	beside	besides
between	beyond	biol	both	brief
briefly	but	by	c	ca
came	can	can't	cannot	cause
causes	certain	certainly	co	com
come	comes	contain	containing	contains
could	couldnt	d	date	did
didn't	different	do	does	doesn't
doing	don't	done	down	downwards
due	during	e	each	ed
edu	effect	eg	eight	eighty
either	else	elsewhere	end	ending
enough	especially	et	et-al	etc
even	ever	every	everybody	everyone
everything	everywhere	ex	except	f
far	few	ff	fifth	first
five	fix	followed	following	follows
for	former	formerly	forth	found
four	from	further	furthermore	g
gave	get	gets	getting	give
given	gives	giving	go	goes
gone	got	gotten	h	had
happens	hardly	has	hasn't	have
haven't	having	he	hed	hence
her	here	hereafter	hereby	herein
heres	hereupon	hers	herself	hes
hi	hid	him	himself	his
hither	home	how	howbeit	however
hundred	i	i'll	i've	id
ie	if	im	immediate	immediately
importance	important	in	inc	indeed
index	information	instead	into	invention
inward	is	isn't	it	it'll

itd	its	itself	j	just
k	keep	keeps	kept	keys
kg	km	know	known	knows
l	largely	last	lately	later
latter	latterly	least	less	lest
let	lets	like	liked	likely
line	little	look	looking	looks
ltd	m	made	mainly	make
makes	many	may	maybe	me
mean	means	meantime	meanwhile	merely
mg	might	million	miss	ml
more	moreover	most	mostly	mr
mrs	much	mug	must	my
myself	n	na	name	namely
nay	nd	near	nearly	necessarily
necessary	need	needs	neither	never
nevertheless	new	next	nine	ninety
no	nobody	non	none	nonetheless
noone	nor	normally	nos	not
noted	nothing	now	nowhere	o
obtain	obtained	obviously	of	off
often	oh	ok	okay	old
omitted	on	once	one	ones
only	onto	or	ord	other
others	otherwise	ought	our	ours
ourselves	out	outside	over	overall
owing	own	p	page	pages
part	particular	particularly	past	per
perhaps	placed	please	plus	poorly
possible	possibly	potentially	pp	predominantly
present	previously	primarily	probably	promptly
proud	provides	put	q	que
quickly	quite	qv	r	ran

rather	rd	re	readily	really
recent	recently	ref	refs	regarding
regardless	regards	related	relatively	research
respectively	resulted	resulting	results	right
run	s	said	same	saw
say	saying	says	sec	section
see	seeing	seem	seemed	seeming
seems	seen	self	selves	sent
seven	several	shall	she	she'll
shed	shes	should	shouldn't	show
showed	shown	showns	shows	significant
significantly	similar	similarly	since	six
slightly	so	some	somebody	somehow
someone	somethan	something	sometime	sometimes
somewhat	somewhere	soon	sorry	specifically
specified	specify	specifying	state	states
still	stop	strongly	sub	substantially
successfully	such	sufficiently	suggest	sup
sure	t	take	taken	taking
tell	tends	th	than	thank
thanks	thanx	that	that'll	that've
thats	the	their	theirs	them
themselves	then	thence	there	there'll
there've	thereafter	thereby	thered	therefore
therein	thereof	therere	theres	thereto
thereupon	these	they	they'll	they've
theyd	theyre	think	this	those
thou	though	thoughh	thousand	throug
through	throughout	thru	thus	til
tip	to	together	too	took
toward	towards	tried	tries	truly
try	trying	ts	twice	two
u	un	under	unfortunately	unless

unlike	unlikely	until	unto	up
upon	ups	us	use	used
useful	usefully	usefulness	uses	using
usually	v	value	various	very
via	viz	vol	vols	vs
w	want	wants	was	wasn't
way	we	we'll	we've	wed
welcome	went	were	weren't	what
what'll	whatever	whats	when	whence
whenever	where	whereafter	whereas	whereby
wherein	wheres	whereupon	wherever	whether
which	while	whim	whither	who
who'll	whod	whoever	whole	whom
whomever	whos	whose	why	widely
willing	wish	with	within	without
won't	words	world	would	wouldn't
www	x	y	yes	yet
you	you'll	you've	youd	your
youre	yours	yourself	yourselves	z
zero	&			

Cuadro 7.2: Listado de *stop-words* en inglés