Benemerita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

“PyText: Una Librería para la Minería de Textos Basada en Python”

TESIS PRESENTADA PARA OBTENER EL TÍTULO DE :
LICENCIATURA EN INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

MARTÍN GARCÍA MORGADO

ASESOR:
Dr. ABRAHAM SÁNCHEZ LÓPEZ

NOVIEMBRE 2016

Agradecimientos

A mi familia, especialmente a mi amada madre Minerva Morgado; que siempre ha estado conmigo en cualquier momento, por su apoyo y paciencia para lograr cada una de mis metas.

Quiero expresar un gran agradecimiento al Dr. Abraham Sánchez López, por creer en mi cada momento y apoyarme en diversas situaciones para lograr ser mejor cada día, siendo siempre parte fundamental de mi formación además de ser un gran ser humano y excelente amigo.

Al M.C. Juan Carlos Conde Ramírez, además de ser un buen maestro es un excelente amigo que siempre ha mostrado una gran actitud y compañerismo, agradezco todos sus consejos y atenciones para llegar hasta este momento.

También agradezco a mis amigos que han sido parte de este ciclo importante como Gabriela Rodríguez, Ricardo I. González, Enrique Diaz, Viridiana Cruz y Mario Posada. Gracias por sus consejos, ánimos y toda la motivación que hasta ahora han brindado y en general por ser siempre grandes compañeros de los cuales he aprendido mucho.

Índice

Introducción

Hoy día nuestra sociedad genera grandes cantidades de información que unido al aumento de las capacidades de almacenamiento, han hecho que todo tipo de organizaciones puedan disponer de una gran cantidad y variedad de datos relativos a su actividad diaria. Esta información ofrece a la empresa una visión perspectiva (qué se está haciendo y cómo se está haciendo) y prospectiva (cómo puede evolucionar la organización en un futuro a corto-medio plazo) y es por ello por lo que tiene una función vital en el proceso de toma de decisiones.

Sin embargo gran cantidad de información obtenida en las bases de datos no se encuentra bien estructurada resultando difícil de explotar desde el punto de vista estadístico por lo que para su utilización es necesario un proceso de tratamiento y análisis exhaustivo de los datos allí recogidos que llamamos a menudo minería de datos.

La minería de textos o text mining puede ser definido [1] como “la aplicación de algoritmos y métodos de los campos del aprendizaje automático y la estadística sobre los textos con el objetivo de encontrar patrones útiles”. Text mining es una herramienta empírica que tiene la capacidad de identificar nueva información o patrones significativos que no son evidentes a partir de una colección de documentos [2] [3].

En principio se puede utilizar cualquiera de los métodos de clasificación estándar usado en data mining para la aplicación en text mining, pero el conjunto de datos proveniente de los documentos de textos necesita un tratamiento previo [4]. Para aplicar text mining a grandes colecciones de documentos es necesario realizar un pre-procesamiento de los documentos de textos y almacenar la información de una forma estructurada [5].

Actualmente la minería de textos puede ser aplicada en el 90% de las actividades diarias que se realizan en internet, principalmente en redes sociales debido a que no solo hay información presente sino que es posible la toma de decisiones basadas en perspectivas, opiniones, comportamientos, y cierto tipo de patrones estimados por el diferente uso al que cada red social puede tomar.

Parte de las aplicaciones potenciales de la minería de textos están basados en modelos y algoritmos estadísticos, los volúmenes de información que se exploran para poder comenzar estudios y experimentos deberán mantener un conjunto de características y así obtener un resultado admisible.

La toma de decisiones, la exploración de nuevos umbrales para la designación de nichos de mercado, la validez de opiniones en un medio social, la aceptación política y respuestas que podrán afirmar el éxito de una idea o negocio; son solo una lista de las posibilidades que la minería de textos ofrece, para este proyecto se escribió una librería escrita en python y algunas utilidades para obtener, procesar, y mostrar resultados respecto a un análisis dado.

En el **Capítulo 1**, se puede observar la necesidad de implementar técnicas para la extracción de información, la importancia de la toma de decisiones y sus respectivas medidas de alcance; en referencia al estudio de la información y los valores que pueden ser mostrados para interpretar resultados respecto a la aplicación de la minería de textos.

En el **Capítulo 2**, se estudiará todo el marco teórico referente a las tecnologías relacionadas con la minería de textos y su alta importancia y relación con el estudio de la minería de datos. Su objetivo es ofrecer un panorama general desde el punto de vista tecnológico, el tipo de técnicas, algoritmos y modelos estadísticos propuestos. También tomar en cuenta los conceptos fundamentales que son requeridos para comenzar a codificar un programa de análisis de información.

En el **Capítulo 3**, se describe el objetivo del proyecto y los requerimientos de software para codificar el conjunto de programas que incluirá la librería de python. Se incluirán algunas herramientas de soporte para extraer, manipular y presentar diversos tipos de información. Posteriormente se ofrece una descripción suficientemente general de las herramientas utilizadas, con el fin de justificar su elección y mostrar diferencias, aplicaciones, usos y soluciones establecidas de acuerdo al tipo de plataforma, escenario y ambiente.

El **Capítulo 4**, cubre de manera general el proceso que se estableció para escribir la librería, sus procesos y análisis requeridos descritos en el Capitulo 3. Si bien el proyecto se establece para demostrar la capacidad de análisis con software, es importante decir que en primera instancia se debe conocer lo que la tecnología de desarrollo es capaz de hacer. En esta sección se agregan los análisis y pruebas para conocer mejor las herramientas y su alcance respecto a un escenario dado.

Por último, el **Capítulo 5** describe globalmente el resultado que se obtuvo al aplicar todos los elementos analizados en capítulos anteriores. Como prueba principal del proyecto en minería de textos se escribió una librería en lenguaje python para demostrar la utilidad del análisis de información. Este capítulo es el marco de referencia del objetivo alcanzado.

Capítulo 1 La Extracción de Información

La minería de datos se enfoca en el análisis de grandes bases de datos. Debido a ello, sus métodos consideran solamente información estructurada, principalmente numérica y booleana, y descuidan tipos de información textual. Como consecuencia de esta situación, muchos logros de la minería de datos parecen tareas muy difíciles de realizar con datos no-estructurados o semi-estructurados.

Se define a la minería de opinión o análisis de sentimiento a la aplicación del procesamiento de lenguaje natural para identificar y extraer la información subjetiva de un texto.

1.1 Extracción de la Información

La idea detrás de la minería de opinión es determinar la actitud del autor de una o más acciones que pueda realizar, como una opinión que califique o determine si algo puede ser bueno o malo respecto a un determinado tema. Esta actitud puede ser si bien su propia evaluación, o algún estado afectivo (estado emocional, de ánimo o percepción de la idea o contexto).

Los trabajos en este campo están orientados a determinar la polaridad de un texto subjetivo, esto es, decidir si un texto es positivo o negativo empleando para ello diferentes métodos, desde la detección de adjetivos que conlleven una dimensión emocional a la clasificación automática mediante el uso de textos previamente descritos.

El primer factor para dar inicio a la minería de textos es la recopilación de documentos relevantes. La principal tarea es limpiar las muestras y asegurar la calidad de estas, debido a que se pueden encontrar diversos formatos entre los documentos. Con la transformación del texto a datos numéricos es posible que los datos cambien de una codificación clásica a data mining. Por lo que la presentación de los datos para data mining y text mining son solo diferentes en una presentación inicial. Existen muchas variantes de representación de documentos, la mayoría de los enfoques están basados en dos enunciados: El primero define a las palabras como atributos y los documentos como muestras, así se podrá formar el conjunto de datos que permitirá desarrollar los diferentes métodos de aprendizaje.

Las fuentes de conocimiento externas benefician en términos de magnitud a muchos de los elementos que componen un sistema de text mining, entregando limitaciones o información adicional sobre los conceptos que se encuentran a través de la aplicación de métodos de aprendizajes, así como servir en la etapa de validación de los resultados [3].

Existen diversas aplicaciones para la minería de textos, de las cuales destacan:

- La **Extracción de información**: El número de documentos y páginas informativas es casi indeterminado en relación a la magnitud que presenta el crecimiento de la web. Es limitado el número de los que pueden llegar a ser parte de un análisis de minería de datos; debido a la escasa estructura, descripción y alto indexado que presentan. En este caso, la minería de textos puede ser de gran utilidad, siendo parte de un conjunto de análisis que permiten la extracción de información relevante de cantidades extensas de textos, y que permite definir entidades y sus relaciones, revelando información semántica significativa, la cual puede ser utilizada como metadatos (es decir una descripción de cierto tipo de información), que a su vez, pueden ser agregados a contenidos web, y así poder facilitar la comprensión de estos documentos. Para esto la objetividad de la Web Semántica debería ser utilizado a priori: dotar de significado explícito los contenidos de la web.
- El **Análisis de sentimientos** o minería de opiniones: Consiste en el análisis de opiniones que generan usuarios en medios masivos de intercambio de información como las redes sociales. Esto ayuda a revelar información importante sobre un tema específico apoyando, por ejemplo, a campos como la inteligencia de negocios y jugando un papel importante en la toma de decisiones de los consumidores. Así si alguna marca requiere el conocimiento de la posición y valor de algún producto en específico, la empresa responsable podría identificar las opiniones y comentarios de los consumidores en Twitter sobre el producto, definiendo las posibles situaciones a las que se puede enfrentar , y con esto, la empresa puede tomar decisiones sobre el rumbo que debe tomar su producto.
- La **Clasificación de documentos**: Es especialmente útil para facilitar la recuperación y navegación de documentos. Clasificar es un factor de alta importancia para el análisis de información debido a que los diversos registros históricos de proyectos, costos, nuevas incorporaciones en el mercado y en general un conjunto de datos que permita una conclusión respecto a una decisión dada, en documentos, si se deseara obtener información de las diferentes áreas de desarrollo de los proyectos, seguramente resultaría una labor sumamente complicada. Gracias al uso de algoritmos de minería de textos es posible agrupar los documentos, obteniendo información descriptiva de cada grupo para facilitar la comprensión de cada uno de ellos.

- La **Elaboración de resúmenes**: Su objetivo es obtener la descripción general de un conjunto de documentos pertenecientes a un tema en específico. De esta manera, los métodos de minería de textos pueden ser clasificados en dos categorías diferentes, 1) Resumen extractivo, es decir, resúmenes conformados por unidades de información extraídas de los textos, o 2) Resumen abstracto, donde la información sintetizada no necesariamente está formada por unidades de información contenida en los textos.

- La **Extracción de conocimiento**: A través de minería de datos, es posible realizar la representación de información extraída a través de modelos de conocimiento.

Como se puede observar, algunas de estas aplicaciones se encuentran ligadas a la web semántica, pues la minería de textos puede apoyar en la anotación semántica de contenidos web, a través de la obtención de información que se menciona en un conjunto de textos.

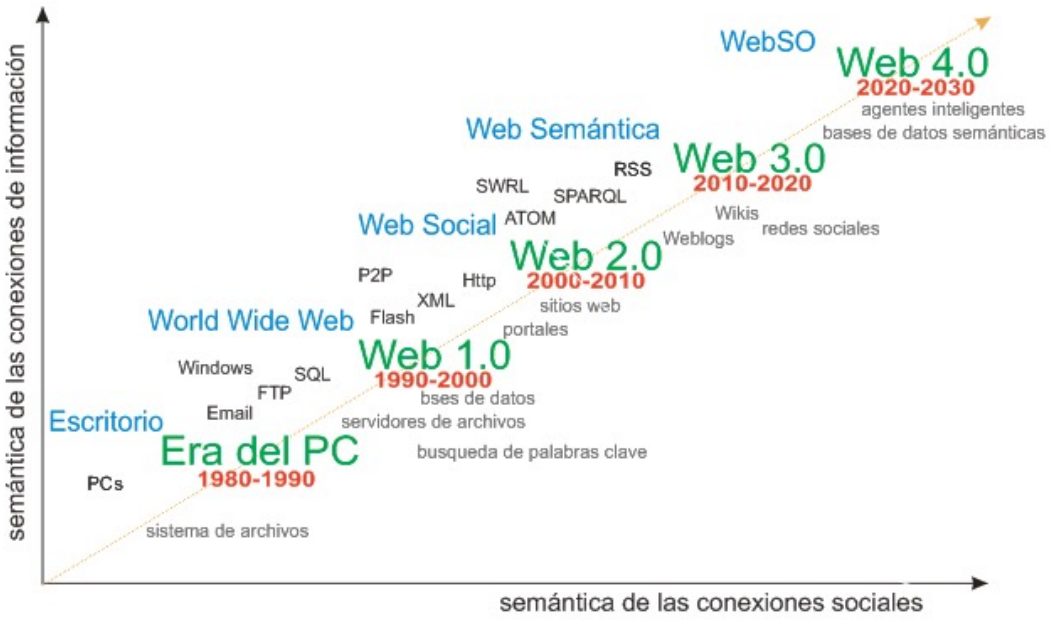


Figura 1.1 La evolución de la Web.

Sin embargo, es importante mencionar que completar manualmente los objetivos de la web semántica implicaría un gran esfuerzo, porque habría que enfrentar dos factores, principalmente por mantener la costumbre que los generadores de contenidos no doten de significado la información que agregan y por otro, la interminable tarea de incluir anotaciones semánticas de los contenidos ya

existentes. Esto si se pudiera lograr hará necesario una gran cantidad de recursos en relación al tiempo y memoria que emplearía además del robusto apoyo de disciplinas como el procesamiento del lenguaje natural (PLN).

Por lo pronto, la minería de textos seguirá requiriendo el soporte en gran medida del procesamiento del lenguaje natural para conseguir que la información contenida en la web tenga significado y pueda ser de fácil acceso aumentando su empleo y utilidad.

De manera independiente, esta disciplina deberá enfrentar algunos desafíos como los que se muestran a continuación:

- Hacer uso del contexto en el cual se generó algún tipo de contenido en las diferentes tareas de la minería de textos. Es importante conocer datos que definan una relación directa; estos pueden describir y agregar características propias de la definición de uno o varios elementos, sólo así se permitirá comprender el sentido de dicha información.

- Extender la búsqueda y la obtención simple de información; buscar la extracción de conocimiento, lo cual implicará la transformación de la información extraída en diferentes tareas del área, a un lenguaje formal que sea legible para las máquinas y así, lograr completamente el entendimiento, por parte de las máquinas, de grandes cantidades de información que carecen de la estructura y/o metadatos suficientes.

- La necesidad de analizar grandes volúmenes de información. Integrar la eficiencia de este proceso en cuestiones de tiempo implica definir algoritmos paralelos que aprovechen de mejor manera la infraestructura y recursos de cómputo actual, en este punto es posible apoyarse de áreas como cloud computing.

El panorama que tiene la minería de textos respecto a la necesidad de hacer formal la búsqueda e integración de la información no sería posible sino existieran procesos bien definidos y establecidos. Se estudió que una metodología para introducir un análisis de datos deberá estar basada en procesos que la definen, de esta forma las técnicas para implementar data mining no solamente involucran el uso de algoritmos, cálculos, determinaciones gráficas y/o exposición y demostración de datos, sino el hecho de hacer factible la minería de textos de modo homogénea.

Existen suficientes aplicaciones que hacen posible que algunas de las propiedades de los sistemas más robustos como: escalabilidad, disponibilidad, interoperabilidad, entre otras; puedan mediante data y text mining implementar y utilizar amplios procesos, que se utilizarán a lo largo de este proyecto como los que muestran a continuación.

1.2 El Proceso Knowledge Discovery in Databases KDD

Todo análisis de información requiere un conjunto de datos establecido, definido y que cumpla características para la precisión de los cálculos pertinentes; esto es posible gracias a un conjunto de técnicas y procesos dedicados al pre-procesamiento y procesamiento de la información. Al momento de asignar tareas para la exploración y así poder involucrar un análisis de información es de vital necesidad establecer un proceso en el cual la información deba ser interpretada.

El proceso KDD (Knowledge Discovery in Databases) [6] está definido como el **“proceso no trivial de identificar patrones válidos, nuevos, potencialmente útiles y en última instancia comprensibles en los datos”**. El término proceso implica que KDD involucra una serie compleja de procedimientos y el término no trivial implica que no se trata de cálculos sencillos.
El proceso KDD es interactivo e iterativo, involucra numerosos pasos con muchas decisiones que deben ser tomadas por el usuario:

- Selección: determina la creación de un conjunto de datos objetivo, seleccionando un conjunto de datos o centralizándolo en un subconjunto de variables o un muestreo de los datos, sobre el cual el descubrimiento de conocimiento será realizado. En este punto las distintas fuentes de datos pueden ser combinadas.

- Pre-procesamiento: tiene por objetivo la depuración de datos con el fin de obtener datos consistentes. Las operaciones básicas incluyen eliminar la entropía, es decir el posible ruido que está incluido en el sistema si es necesario además de estrategias para el manejo de los campos de datos vacíos.

- Transformación: consiste en el cambio que sufren los datos utilizando principalmente la reducción de la dimensionalidad o métodos de transformación simple. Los datos son transformados en formas apropiadas para la minería de datos y/o se seleccionan los atributos más útiles capaces de representar estos datos dependiendo de la aplicación propuesta.

- Data Mining: es un proceso donde diversos métodos de inteligencia son aplicados con el propósito de realizar una búsqueda y extracción de patrones de datos de interés en una forma particular de representación, por ejemplo: clasificación o clustering (agrupamiento).

- Interpretación/Evaluación: consiste en la forma en la que los datos son mostrados y así determinar su interpretación respecto al objetivo dado y de los patrones encontrados. Se identifican los patrones realmente interesantes basados en alguna medida de interés.

El proceso KDD debe ser precedido por el desarrollo de un entendimiento del dominio de aplicación es decir el área específica donde se aplicará el proyecto de minería de datos y la correcta identificación de las metas desde el punto de vista del usuario final [7]. Una vez finalizado el proceso KDD, **el conocimiento descubierto puede ser usado directamente, incorporado a otros sistemas** para futuras acciones o simplemente documentado y reportado a las partes interesadas [8].

KDD puede involucrar importantes iteraciones y contener bucles entre pares de etapas La mayor parte del trabajo en el proceso KDD está enfocado en la etapa del data mining, aunque el resto de las etapas son igualmente importantes para la exitosa aplicación de KDD en la práctica.

1.3 El Modelo de Proceso Cross Industry Standard Process for Data Mining CRISP-DM

La interoperabilidad de las aplicaciones en la minería de datos puede ser posible debido a la alta formalidad de las operaciones. Este modelo está basado en una metodología que proporciona una descripción del ciclo de vida del proyecto de minería de datos. El modelo consiste en un ciclo de seis fases [9]:

- Business Understanding (Comprensión del ámbito de aplicación): enfoca la comprensión de los objetivos y requerimientos del proyecto desde una perspectiva no técnica, para luego convertir este conocimiento de los datos en la definición de un problema de minería de datos y en un plan preliminar diseñado para el alcance de los objetivos.

- Data Understanding (Comprensión de los datos): esta fase comienza con la recopilación inicial de datos y continúa con las actividades que permiten familiarizar los datos y poder verificar su calidad.

- Data Preparation (Preparación de datos): trata de cubrir todas las actividades necesarias para construir el conjunto de datos que será utilizado en las herramientas de modelado a partir de los datos puros iniciales. Las tareas incluyen selección de atributos junto con limpieza, integración y formato de datos.

- Modeling (Modelado): Las técnicas de modelado son seleccionadas y aplicadas. Se realiza el diseño para la posterior evaluación del modelo.

- Evaluation (Evaluación): Los resultados finales obtenidos del modelo son evaluados y se revisa cada etapa del proceso realizado. Luego se compara el modelo obtenido con los objetivos que inicialmente se plantearon

- Deployment (Desarrollo): en esta fase el conocimiento obtenido es organizado y presentado en el modo en el que el usuario final pueda usarlo. Se establece una planificación de la monitorización y del mantenimiento del proceso, se generan los informes finales y se realiza la revisión del proyecto.

En la Figura 1.2 se muestra el proceso CRISP-DM, donde la secuencia de las fases no es en un solo sentido, ya que el flujo entre fases diferentes es siempre requerido. Las flechas indican las dependencias más importantes y frecuentes entre fases.

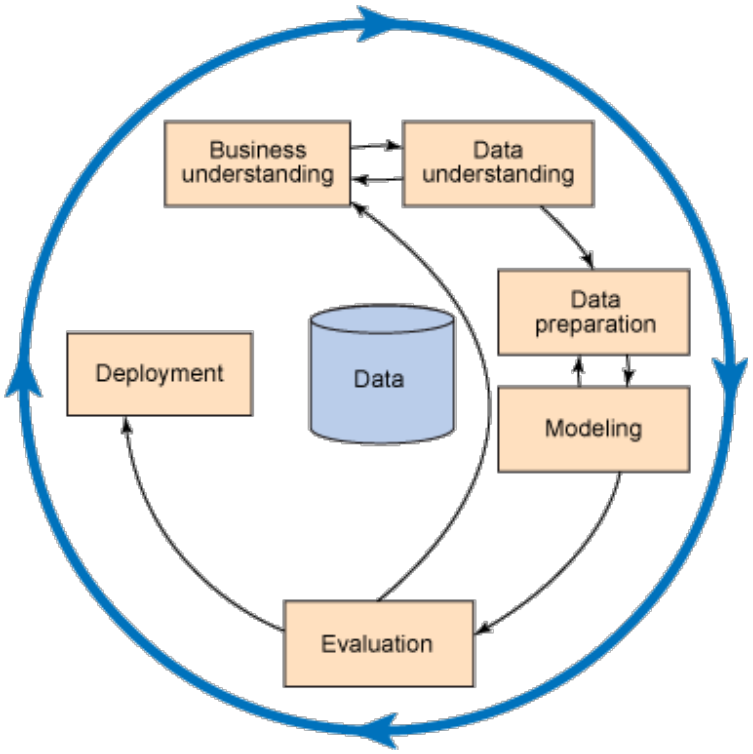


Figura 1.2 El proceso CRISP-DM

CRISP-DM puede ser visto como una implementación formal del proceso KDD, guiando a los usuarios en la puesta en práctica de data mining en sistemas reales [10]. En la Tabla 1.1 se puede observar las relaciones de las etapas entre KDD y CRISP-DM, donde la etapa de comprensión del ámbito de aplicación puede ser identificada con el desarrollo del entendimiento del dominio de aplicación, el conocimiento previo relevante y las metas de los usuarios finales, mientras que la etapa de desarrollo puede ser identificada con la consolidación a través de la incorporación del conocimiento al sistema.

KDD	CRISP-DM
Pre-KDD	Comprensión del Ámbito de Aplicación
Selección	Comprensión de los Datos
Pre-Procesamiento	
Transformación	Preparación de los Datos
Data Mining	Modelado
Interpretación / Evaluación	Evaluación
Post-KDD	Desarrollo

Tabla 1.1 Relaciones entre KDD y CRISP-DM

Conclusiones del Capítulo

Se muestran los principales factores para la extracción de información además de resaltar su importancia. Así la exploración en los diversos tipos de contenido y recolección de información darán lugar a la aplicación que el análisis de información va a tomar. En primera instancia, la formalidad para dar origen a un análisis de datos está basado en los procesos de data mining que serán utilizados, se observó que no depende de una serie de procedimientos previamente definidos para obtener el alcance del análisis sino de una correcta designación de un landscape tecnológico para su desarrollo.

La importancia de la comprensión de los procesos KDD y CRISP-DM son el punto inicial para el análisis de datos que se propone en este proyecto. Lo fundamental en este desarrollo no solo es la forma en la que se tienen que realizar las actividades, sino las posibles decisiones que se deberán tomar al momento de obtener resultados. Esta librería además de involucrar a estos procesos y de la parte experimental propuesta, estará inmersa hacia sistemas que muestran información de diversas fuentes.

Capítulo 2 Fundamentos de Minería de Textos

Esta investigación está fundamentada por una serie de conceptos que contemplan el funcionamiento de las técnicas para implementar minería de datos y textos. En primera instancia es importante diferenciar su utilidad y al mismo tiempo considerar en todo momento su relación e importancia en el desarrollo de este proyecto.

La idea principal de que cada uno de los conceptos sean definidos es el marco de referencia y estudio necesarios al momento de escribir la librería. Los conceptos si bien ilustrarán las situaciones e importancia de que sean implementados en diferentes escenarios.

2.1 Minería de Textos (Text mining)

La minería de datos, conocida como el **descubrimiento del conocimiento de los datos**, se puede definir como “la ciencia de extraer conocimiento útil a partir de robustos repositorios de datos”. De acuerdo a esto, la minería de texto se refiere a un proceso de descubrimiento de tal conocimiento cuando **el origen de los datos en cuestión es el texto**.

Text mining es una herramienta empírica que tiene la capacidad de identificar nueva información o patrones significativos que no son evidentes a partir de una colección de documentos.

Estrictamente hablando, en lugar de áreas específicas del conocimiento por sí mismos, la minería de datos y la minería de textos en general, deben ser consideradas como campos interdisciplinarios orientados a la aplicación. En el caso particular de la minería de texto, que se puede encontrar para estar estrechamente relacionado con disciplinas tales como el procesamiento del lenguaje natural, la lingüística computacional y la recuperación de la información, así como para depender de las contribuciones importantes de las estadísticas, el aprendizaje automático y la inteligencia artificial [11]. Debido a su estrecha relación y dependencia de estas disciplinas afines, las definiciones precisas del ámbito de aplicación de la minería de texto y sus fronteras con estas otras disciplinas no pueden ser fácilmente representadas. En este sentido, el concepto de minería de texto sólo se muestra para una determinada técnica o aplicación bajo el esfuerzo de descubrir el conocimiento de una gran colección de datos textuales.

2.1.2 Representación de Documentos

La representación de un documento de texto está basada en las palabras, modelo conocido como Vector Space Model o “bag of words”. Dada esta representación un documento es considerado simplemente como una colección de palabras que ocurren al menos una vez. El orden de las palabras, la combinación en las cuales ellas ocurren, la estructura gramatical, la puntuación y el significado de las palabras son todos ignorados.

Al conjunto de atributos típicamente se le llama diccionario [3]. Existen dos enfoques para desarrollar un diccionario: local y global. El enfoque de diccionario local se puede utilizar cuando se conoce a priori las distintas categorías a las que pueden pertenecer los documentos, elaborando para cada categoría un diccionario distinto, solo con las palabras que aparecen en los documentos clasificados en la categoría específica. En el enfoque de diccionario global se incluyen todas las palabras que ocurren al menos una vez en cualquiera de los documentos. El primer enfoque logra que cada diccionario sea relativamente pequeño pero tiene un alto costo en tiempo para su construcción.

De manera general, los documentos de texto tienen un gran número de atributos y la mayoría de ellos con una baja frecuencia. Con frecuencia circunstancial, se desea trabajar con un diccionario más pequeño y la información sobre la frecuencia de ocurrencia de las palabras puede ser muy útil para la reducción del diccionario además de poder mejorar el desempeño de predicción para algunos métodos. Tanto las palabras más frecuentes como las menos frecuentes podrían ser descartadas.

Una vez determinado que el número final de atributos es N, se puede representar los términos en el diccionario en algún orden arbitrario como $t_1,t_2,...,t_N$. Así se puede representar el i-ésimo documento como un conjunto ordenado de N valores: $(X_{i1}, X_{i2}, ... , X_{iN})$. El conjunto de vectores que representan todos los documentos en consideración es llamado un Vector Space Model. En general, se puede decir que el valor X_{ij} es una medida ponderada de la importancia del j-ésimo término t_j en el documento i [4].

Los elementos del vector son representados por un término (una palabra o un grupo de palabras) de la colección de documentos, es decir, el tamaño del vector está definido por el número de palabras de la colección completa de documentos.

Vector Space Model analiza eficientemente grandes colecciones de documentos, a pesar de su estructura simple y sin el uso de información semántica. Este modelo fue introducido principalmente para *information retrieval*, pero actualmente es aplicado extensamente en text mining[13].

Existen varias formas de representar la importancia ponderada de un término dentro de un documento: frecuencia, binaria y TF-IDF [12] [13].Una de las formas más comunes de calcular la importancia ponderada es contar el **número de ocurrencia para cada término en un documento dado**. Una representación binaria consiste en que el valor 1 indica que el término está presente en el documento, de lo contrario, el valor 0 indica la ausencia del término. Una forma más compleja de calcular la importancia ponderada es llamada TF-IDF(term frequency-inverse document frequency), la cual combina la frecuencia de un término con una medida de rareza del término en el conjunto completo de documentos.

2.1.3. Tokenización

El primer paso para el manejo de texto es separar el **stream de caracteres** para transformarlo a palabras o a *tokens*. El proceso de tokenización es altamente dependiente del lenguaje sobre el cual se está trabajando y las reglas específicas de este, por lo que esta tarea resulta trivial para una persona familiarizada con la estructura del lenguaje, mientras que para un programa computacional puede convertirse ser altamente complejo [3].

Para identificar las distintas palabras (o tokens) es necesario primero definir los delimitadores de los tokens, los que generalmente corresponden a los signos de puntuación y otros caracteres distintos a las letras del alfabeto [13]. Luego los delimitadores de tokens se separan de las palabras y son reemplazados por un espacio blanco simple [10]. De esta forma cada palabra queda separada por un espacio blanco simple y facilita la tokenización.

2.1.4. Stopwords

Stop words es un **método de filtración** que consiste en remover palabras comunes que resultan inútiles para la caracterización de un documento [12].

El objetivo de utilizar una lista de stop words es remover palabras que inciden muy poco o no contienen información útil, como artículos, pronombres, conjunciones, preposiciones, etc. Las palabras que aparecen en la mayoría de los documentos aportan poca información para distinguir los distintos tipos de documentos, así como las palabras que ocurren muy raramente es probable que no tengan una relevancia estadística y pueden ser removidas del diccionario [13].

El hecho de remover las palabras mediante stop words podrá lograr una reducción significativa el tamaño del diccionario, pero no existe una lista fija de stop words que sea universalmente utilizada [12].

2.1.5. Stemming

En cuanto el documento de texto ha sido separado en una secuencia de tokens, lo que ahora se realizará es **convertir cada token a una forma estandarizada**, proceso llamado stemming.

La aplicación de stemming se basa en la observación de que las palabras en los documentos a menudo tienen muchas variantes morfológicas. Las palabras que tienen una misma raíz lingüística pueden ser tratadas como una única palabra, la cual entrega probablemente una mejorar descripción del contenido del documento, lo que podría no ocurrir si se utilizara cada palabra individualmente [12].

El objetivo de stemming es reconocer los conjuntos de palabras que pueden ser tratadas como equivalentes. Frecuentemente no hay necesidad para mantener el singular y plural de una misma palabra, así como los verbos pueden ser almacenados es su forma infinitiva. También se puede extender el concepto hacia los sinónimos.

Algunos de los efectos de la aplicación de stemming es la reducción del número total de atributos dentro del texto (o reducción del tamaño del diccionario [14]) y el incremento de la frecuencia de ocurrencia de algunos atributos [12].

En este caso específico de stop words, no hay un algoritmo de stemming que sea universal, donde el idioma es el principal factor. Por otro lado la aplicación de stemming debe ser implementada con precaución para no eliminar palabras del diccionario que puedan resultar relevantes, dado que no se considera la semántica de las palabras [13].

2.2 Análisis de Sentimientos

Las opiniones son fundamentales para casi todas las actividades humanas, porque son factores importantes de influencia en nuestros comportamientos.

El análisis de sentimientos (AS), también llamado minería de opinión es un campo de estudio que analiza las opiniones, sentimientos, evaluaciones, actitudes de las personas hacia entidades como productos, servicios, organizaciones, individuos, cuestiones, eventos, tópicos y sus atributos. Dicha área tiene un amplio rango de aplicaciones casi en todos los dominios, la industria es la más interesada en la proliferación de aplicaciones comerciales, además sin mencionar que por primera vez en la historia, se cuenta con una gran cantidad de información en los medios de comunicación social y la web en general, lo cual ha originado que el análisis de sentimientos sea el centro de investigación de los medios sociales. La investigación de AS no solo ha tenido un gran impacto en el procesamiento de

Lenguaje Natural (NLP), si no también ha tenido un profundo impacto en las ciencias de la administración: ciencias políticas económicas y sociales.

El análisis de sentimientos se conoce también como clasificación de sentimientos, o clasificación de sentimiento a nivel documento, ya que se considera todo el documento como una unidad. El problema se define de la siguiente manera:

Dado un conjunto de documentos de texto de evaluación *D* que contienen opiniones (o sentimientos) acerca de objetos, se pretende extraer atributos y componentes de objetos que han sido comentados en cada documento *d* en *D* y determinar si los comentarios son positivos, negativos o neutros

Consta de dos pasos: Identificación y Clasificación.

- Identificación: Se identifican las partes con enunciados subjetivos en el texto (opiniones).

- Clasificación : Se realiza la clasificación de las opiniones: Positiva, Neutra y Negativa.

En la figura 2.1 se muestra el proceso del análisis de sentimientos que consta de 3 fases, la obtención de las opiniones, la identificación de subjetividad y la clasificación de las opiniones.

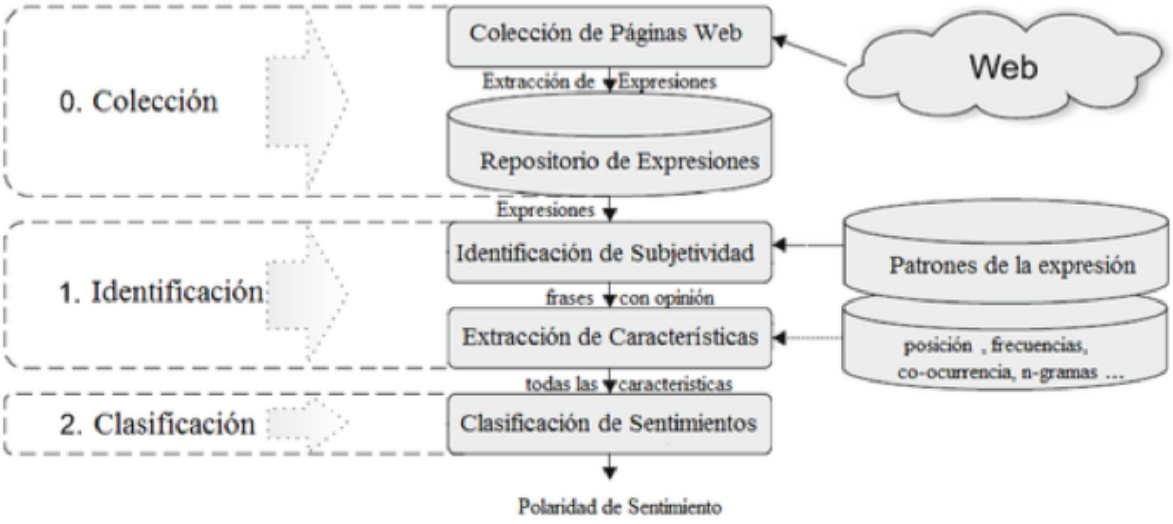


Figura 2.1 Arquitectura de Análisis de Sentimientos.

Para este problema existen dos formulaciones basadas en los tipos de valores que toma *s*, donde *s* es el conjunto de características, si *s* toma valores categóricos como positivo, negativo, entonces es un problema de clasificación. Si *s* toma valores numéricos o puntuaciones dentro de un rango, entonces es un problema de regresión. [15]

Las opiniones y los sentimientos tienen una característica importante a diferencia de la información factual, son subjetivos, por lo que se necesita analizar las opiniones de varias personas y no solo de una. El tamaño influye en la dificultad de las opiniones, pues es un factor determinante para lograr una alta precisión en el análisis de sentimiento. Por ejemplo las opiniones obtenidas de twitter que son de a lo mas 140 caracteres, las hace opiniones más centradas y más enfocadas con poca información irrelevante, siendo más fácil de analizarlas que por ejemplo, una opinión en un foro en donde las personas interactúan una con la otra y la dimensión de la opinión dependerá del dominio del tema. Las opiniones sobre productos y servicios son generalmente más fáciles de analizar. Discusiones sociales y políticas son mucho más difíciles debido a la complejidad del tema y el sentimiento, expresiones, sarcasmo e ironía.

Los indicadores más importantes de sentimientos, son las palabras que expresan sentimiento, llamadas palabras de opinión (opinion words). Estas son palabras que comúnmente son usadas para expresar sentimientos positivos o negativos. Por ejemplo, *bueno, maravilloso y genial* son palabras que expresan sentimiento positivo, en cambio *malo, pésimo y terrible* son ejemplos de palabras que expresan sentimiento negativo, a dichas palabras se les conoce comúnmente como lexicón de opiniones (sentiment lexicon o opinion lexicon). A pesar de que las palabras y frases con sentimiento son muy importantes para el análisis de sentimientos no son suficientes para obtener éxito, la tarea es mucho más compleja, es decir que el lexicón de opiniones es necesario pero no suficiente para el AS. A continuación se describen algunas situaciones que hacen de AS un problema complejo.

- Identificación: Se identifican las partes con enunciados subjetivos en el texto (opiniones).

- Una palabra que expresa un sentimiento negativo o positivo puede tener orientaciones opuestas, según el contexto de la oración. Por ejemplo "sencillo" usualmente indica un sentimiento positivo como en la oración "El método es sencillo de entender", y en otra oración como "El evento fue muy sencillo para la ocasión" tiene un sentimiento negativo.

- Una oración que contiene una palabra considerada como expresión de sentimiento, puede no expresar un sentimiento. Dicho fenómeno ocurre en oraciones interrogativas y condicionales. Por ejemplo, "¿Qué automóvil es bueno?" y "Si encuentro un auto bueno, lo compraré.", en donde ambas

oraciones contienen la palabra "*bueno*" y en ningún caso se expresa algún sentimiento positivo o negativo. Sin embargo no todas las oraciones interrogativas y condicionales expresan algún sentimiento, como en la oración "¿Alguien sabe como reparar este error fatal?" y "Si buscas un auto bueno, la opción es Hyundai."

- Oraciones Sarcásticas como "Que fantástico smartphone! Deja de funcionar al instante", estas oraciones no son muy comunes en opiniones de productos o servicios, pero son muy comunes en discusiones políticas.

- Existen oraciones que no contienen palabras que expresan algún sentimiento, es decir oraciones objetivas que expresan información factual y sin embargo son oraciones con sentimiento negativo o positivo. Un ejemplo es "Este celular se descarga muy rápido" lo cual expresa una opinión negativa acerca del celular. Otra oración es "Después de probar el servicio, encontramos muchas irregularidades", que también es una opinión negativa acerca del servicio.

- Las opiniones *spam* han llegado a ser el mayor problema, ya que cualquier persona tiene acceso a la web y es libre de expresar una opinión sin tener la necesidad de identificarse, lo que ha originado consecuencias indeseables, puesto que personas con identidades ocultas e intenciones maliciosas y haciéndose pasar por público en general realicen publicaciones falsas para promover o bien desacreditar algún producto, servicio, organización o individuos sin ser descubiertas sus verdaderas intenciones. Dichos individuos son llamados escritores de falsas opiniones (opinion spammers) [25].

2.2.1 Opinión

Para definir el concepto de opinión, se presenta un ejemplo:
Publicado por: John Smith Date:10-09-2011 "(1) Compré una cámara Canon G12 hace seis meses. (2) Simplemente me encanta. (3) La calidad de las imágenes es impresionante. (4) La duración de la batería es larga. (5) Sin embargo, mi esposa piensa que es muy pesada para ella."

La pregunta es: ¿Qué extraer de la opinión?

Del ejemplo anterior podemos detectar que contiene frases con un sentimiento positivo y negativo acerca de la cámara G12. La oración (2) expresa una opinión positiva acerca de la cámara, la oración (3) expresa una opinión positiva de la calidad de las imágenes, la oración (4) expresa una opinión positiva de la duración de la batería, y la expresión (5) es una opinión negativa acerca del peso de la cámara.

Una opinión consiste de dos componentes clave: un objeto *g* y un sentimiento *s* del objeto. (*g,s*) En donde *g* puede ser una entidad o aspecto de la entidad acerca de la opinión expresada y s es un sentimiento positivo, negativo, neutro o una puntuación que expresa la fuerza o intensidad del sentimiento. Positivo, negativo y neutro son llamados sentimientos (u opiniones) orientaciones o polaridades. Por ejemplo la oración (3) podría descomponerse de la siguiente manera:

(Cannon-G12, Calidad-imagen)

Otro aspecto importante es que la opinión habla acerca de dos personas, quien son llamadas fuentes de opinión (opinion sources) y emisor de la opinión (opinion holder). Y finalmente también la fecha es importante ya que frecuentemente se quieren conocer las opiniones a lo largo del tiempo y como van cambiando.

A continuación se da la **definición formal de opinión**:

Una opinión es una cuádrupla

$$(g,s,h,t) \tag{2.1}$$

Donde *g* es la opinión, *s* es el sentimiento de la opinión, *h* es la persona que expresa la opinión, y *t* es el tiempo o la fecha en que se expresa la opinión.

Una entidad e es un producto, servicio, tópico, persona, organización o evento. Se describe con un par e: *(T,W)*, donde *T* es una jerarquía de partes, subpartes, etc. Y W es un conjunto de atributos de *e*.

Por ejemplo continuando con el ejemplo de la cámara, un modelo de cámara en particular es una entidad y sus atributos son la calidad de la imagen, el tamaño, peso y su conjunto de partes son el lente, visor y la batería. La batería a su vez también tiene un conjunto de atributos, la vida de la batería y el peso de la misma. De lo anterior, puede concluirse que una entidad se define como una descomposición jerárquica de sus partes, donde la raíz es la entidad. Las jerarquías de dos niveles se pueden simplificar utilizando términos llamados aspectos o características para denotar partes y atributos.

Después de la descomposición antes mencionada se puede redefinir la opinión como:

Una opinión es una quíntupla

$$(e_i, a_{ij}, s_{ijkl}, h_k, t_l) \tag{2.2}$$

Donde e_i es el nombre de la entidad, a_{ij} es un aspecto de e_i, ,s_{ijkl}, es el sentimiento en el aspecto a_{ij} de la entidad e_i, ,h_k es el autor de la opinión, y t_l es el tiempo en el cual es expresada la opinión por h_k. Es decir una opinión está dada por s_{ijkl}, que está dada por un autor h_k acerca de aspectos a_{ij} de una entidad e_i en un tiempo t_l.

Ahora se definirá el concepto de modelo de entidad, un modelo de documentos de opinión y el objetivo de la minería, también nombrados aspectos basados en minería de opinión

Modelo de entidad

Una entidad e_i es representada por un conjunto de aspectos, $A_i = \{a_{i1}, a_{i2}, ..., a_{in}\}$. Cada aspecto a_{ij} ε A_i de la entidad puede ser expresado por algún conjunto finito de expresiones de aspectos $AE_{ij} = \{ ae_{ij1}, ae_{ij2}, ..., ae_{ijm}\}$.

Por ejemplo.

Sean $A_i = \{a_{calidad}, a_{peso}, ..., a_{color}\}$, los aspectos de una cámara, donde cámara es una entidad.

2.2.2 Modelo de Documentos de Opinión

Un documento de opinión *d*, contiene opiniones sobre un conjunto de entidades $\{e_1, e_2, \ldots, e_r\}$ expresadas por un conjunto de autores de opinión $\{h_1, h_2, ..., h_p\}$. La opinión de cada entidad e_i expresa la entidad misma y su subconjunto de aspectos.

Ahora bien el objetivo de la minería de opinión es:

Dada una colección de documentos de opinión *D*, descubrir las opiniones o quíntuplas $(e_i, a_{ij}, s_{ijkl}, h_k, t_l)$ en *D*.

Para la realización de este objetivo son necesarias las siguientes tareas:

- Extracción de entidades y agrupamiento
- Extracción de aspectos y agrupamiento

- Determinación del autor de la opinión y el tiempo de la opinión.
- Generación de la quíntupla

La dificultad de la minería de opinión, radica en que todas las tareas anteriores son problemáticas recientemente abordadas y por tanto no resueltas, además también hay problemáticas en la sintaxis usada puesto que una frase puede no mencionar explícitamente algunas piezas que están implícitas como pronombre, convenciones y contexto.

Una vez definidos los conceptos importantes en AS, a continuación se presentan algunas características utilizadas para la representación de documentos en el AS.

2.3 Representaciones Textuales

Para realizar el trabajo presentado, 4 representaciones son de interés, N-gramas, etiquetado POS, teoría de la valoración y TF-IDF o bolsa de palabras.

2.3.1 N-gramas

Es una representación tradicional en recuperación de la información, que consiste de palabras individuales (unigramas), o conjuntos de palabras (n-gramas), con sus frecuencias asociadas. En algunos casos podemos representar mejor un concepto mediante la unión de *n* palabras que se encuentran adyacentes al término principal, lo que se le conoce como *n-gramas*. La importancia de esta representación radica en que la posición de las palabras es considerada, puesto que el significado de una palabra, no tiene sentido sin las palabras adyacentes que le acompañan en cualquier texto, por lo que la posición de una palabra afecta potencialmente en el sentido del significado de la oración, es decir el sentimiento o la subjetividad dentro de una unidad textual.

Para el trabajo realizado se utiliza *n*-gramas de tamaño n=2, es decir, **bigramas**.

Un bigrama o digrama, es un caso especial del n-grama, es un grupo de dos letras, dos sílabas, o dos palabras. Los bigramas son utilizados comúnmente como base para el simple análisis estadístico de texto. Se utilizan en uno de los más exitosos modelos de lenguaje para el reconocimiento de voz [16].

2.3.2. Partes de la Oración (POS)

Del inglés “Part of the speech”. Es una técnica de representación muy utilizada está basa en las reglas lingüísticas, donde las palabras y frases son categorizadas

como sustantivos, verbos, adjetivos y adverbios. De acuerdo con Turney, son características gramaticales que tienen la capacidad de expresar subjetividad [17]. Existen investigaciones enfocadas principalmente en adjetivos y adverbios, como en el trabajo reportado por Farah Benamara et al [18], en donde expone que las expresiones de una opinión dependen principalmente de algunas palabras, por ejemplo, la palabra "*bueno*" es utilizada comúnmente para una opinión positiva, y la palabra "*malo*", para algo negativo, dichas palabras son identificadas como adjetivos en términos lingüísticos.

En general los adjetivos son importantes indicadores en una opinión, son considerados características especiales, sin embargo no significa que otras partes de la oración no contribuyan a la expresión de sentimientos. Existen trabajos en donde los sustantivos, verbos, adverbios y sustantivos subjetivos también han tenido buenos resultados [19].

2.3.3. TF-IDF (Term Frequency-Inverse Document Frequency)

Es un esquema de ponderación de términos comúnmente utilizado para representar documentos de texto como vectores, que se ajusta al modelo denominado bolsa de palabras, donde cada documento es representado como serie de palabras sin orden. Se trata de una medida estadística de que tan importante es una palabra para un documento en un corpus. Dicha técnica es utilizada para hacer ranking u ordenaciones de los resultados de búsqueda, generación de resúmenes de texto, agrupación y clasificación de documentos, identificación de la autoría de algún texto, recomendación de documentos, etc.

Cálculo del TF

Un término t_j que aparece muchas veces en un documento d_i es más importante que otro que aparece pocas.

$$tf_{ij} = \frac{(n_{ij})}{\sum_{i=1}^{N} n_{ij}} = \frac{(n_{ij})}{|d_i|} \qquad (2,3)$$

Donde n_{ij} es el número de veces que aparece el término t_j en el documento d_i y $\sum_{i=1}^{N}$ es la sumatoria del número de veces que aparece el término t_j en todos los documentos.

Cálculo del IDF

Un término t_j que aparece en pocos documentos, discrimina mejor que uno que aparece en muchos.

$$idf_i = \log\left(\frac{N}{n_j}\right) \qquad (2,4)$$

Donde N es el número total de documentos, y n_j es el número de documentos que contiene el término t_j.

Representación final del documento

Cada elemento queda representado como un vector de características d_j:

$$d_j = (d_{j1},...,d_{jn}) \qquad (2,5)$$

donde, $d_{ij}=tf_{ij}*idf_{ij}$

Es decir finalmente se seleccionan *n* términos con los valores más altos en todos los documentos.

2.3.4. Teoría de la Valoración Utilizando Reglas Sintácticas

La teoría de la valoración propuesta por Peter R.R White [20], se ocupa de los recursos lingüísticos por medio de los cuales las personas expresan alguna opinión. Particularmente del lenguaje (expresiones lingüísticas), la valoración, la actitud y la emoción del conjunto de recursos que explícitamente posicionan de manera interpersonal las propuestas y proposiciones textuales. Es decir trabaja con los significados de las palabras que hacen variar o modificar los términos del compromiso del hablante en sus emisiones, es decir, que modifican lo que está en juego en la relación interpersonal.

Dicha técnica fue implementada por Víctor Morales en su trabajo [21], que haciendo uso de un diccionario de aptitud, el cual contiene características de la teoría de la valoración *(juicio, apreciación y afecto)*, utilizan sintagmas adverbiales con el fin de que dichas reglas obtengan una valor más preciso acerca del significado de cada palabra. El objetivo es contabilizar los valores de positivo, negativo, juicio, apreciación y afecto, que están presentes en una opinión

cualquiera, como si se tratase de una bolsa de palabras ponderada, sin embargo las reglas sintácticas juegan un papel primordial en este proceso, ya que dependiendo del tipo de regla, los valores pueden aumentar, disminuir, o intercambiarse, afectando de esa manera los valores finales asignados al sentimiento de cada opinión.

2.4 Clasificación

Para predecir la categoría de clase es necesario primero establecer el problema de la clasificación. Para ello es importante definir el concepto de aprendizaje automático:

Los algoritmos de aprendizaje automático son métodos que dado un conjunto de ejemplos de entrenamiento infieren un modelo de las categorías en las que se agrupan los datos, de tal forma que se pueda asignar a nuevos ejemplos una o más categorías de manera automática mediante analogía de patrones en dicho modelo.

Las técnicas más utilizadas para la determinación de las clases, son los métodos de clasificación de documentos que llevan a cabo el aprendizaje supervisado, el cual cuenta con una clase que contiene las etiquetas de las instancias, y también las técnicas de aprendizaje no supervisado, o agrupamiento, que no cuentan con información adicional a los datos, como es el caso de el aprendizaje supervisado, la diferencia entre ambos se muestra en la tabla 2.2, en donde se puede observar, que el aprendizaje supervisado realiza un proceso de entrenamiento y prueba, en el cual existe una partición de los datos para cada proceso respectivamente, una vez realizado el proceso de entrenamiento, el algoritmo se prueba y se obtiene un resultado, el cual es comparado con la clase etiquetada, si es alcanzado el resultado adecuado según la métrica definida, entonces el proceso termina, si no lo es, entonces se repite el proceso de entrenamiento, hasta obtener un resultado satisfactorio. En el caso de el aprendizaje no supervisado, no se cuenta con información adicional a los datos, Por lo que la condición de paro en este caso, será hasta que las instancias estén suficientemente separadas y los grupos formados sean diferenciables.

Para el **análisis de sentimientos se utiliza la técnica de aprendizaje supervisado.**

El Análisis de sentimientos (AS), como ya se menciono, es usualmente formulado como un problema de clasificación de texto en el cual son consideradas dos clases: positiva y negativa. Por lo general la clase neutra no es utilizada.

Para esta tarea de clasificación, se han aplicado métodos de aprendizaje supervisado como Máquina de Soporte Vectorial (SVM), Naive Bayes (NB). El primer trabajo realizado que tomó éste enfoque de clasificación fue realizado en [22], en el cual se clasificaron opiniones de películas considerando dos clases, usando unigramas (bolsa de palabras) como características y los dos clasificadores antes mencionados.

Para la realización del análisis de sentimiento, primero se necesita representar los textos de manera computacional para su análisis. Para construir la representación, es importante considerar que las palabras relacionadas en cada texto a analizar son las características principales, y es aquí donde se presenta la primera dificultad del problema, puesto que la clave para poder obtener resultados satisfactorios en el Análisis de Sentimientos, es la ingeniería de selección del conjunto de características efectivas [23].

No Supervisado	Supervisado
Selección del Método de Agrupamiento	Selección y Extracción de Características
Agrupamiento de Datos	Selección de un Modelo de Clasificación
Detección de Desviaciones	Entrenamiento / Prueba
Segmentación	Árboles de Decisión
Reglas de Asociación	Inducción Neuronal
Patrones Secuenciales	Regresión, Series temporales

Tabla 2.1 Tipos de aprendizaje

2.4.1 Combinación de Clasificadores

La idea principal de un ensamble de clasificadores, es combinar un conjunto de clasificadores para resolver una tarea en conjunto, en donde el objetivo principal es combinar las salidas de los clasificadores base, para generar una salida en donde sean considerados todos los clasificadores y dicha salida sea mejor que la obtenida por cualquier clasificador base.

Un ensamble de clasificadores es un grupo de clasificadores quienes individualmente toman decisiones que son fusionadas de alguna manera, para finalmente obtener una decisión por consenso.

La selección de los clasificadores base se puede realizar de dos maneras: estático o dinámico. En el enfoque estático, se aplica el mismo subconjunto de clasificadores base que se seleccionan para todas las muestras de prueba, en el enfoque dinámico la selección se realiza para cada nueva instancia individualmente.

Posterior a la obtención de la colección de los clasificadores base, el siguiente paso es combinar las salidas en orden de obtener una decisión final, en esta fase, debe ser considerado principalmente el tipo de información que se va a combinar y qué método de combinación se va a aplicar.

Otro aspecto importante son las salidas de los clasificadores, diferentes métodos de combinación utilizan diferentes tipos de salidas de clasificadores base, por ejemplo una etiqueta de clase o una distribución de probabilidad. Un enfoque alternativo es utilizar predicciones como un conjunto de atributos para formar una función de combinación en términos de meta-aprendizaje [24]. En años pasados, estudios experimentales realizados por la comunidad de machine learning, mostraron que la combinación de las salidas de múltiples clasificadores reducen la generalización del error. Los métodos de ensamble son muy efectivos, debido principalmente a que varios tipos de clasificadores tienen sesgos inductivos, y provocan que la diversidad de los clasificadores utilizados reduzca el error de la varianza, sin incrementar el error bias [25].

La combinación de clasificadores y por lo tanto la creación de ensamble de clasificadores fue propuesto para mejorar los resultados obtenidos por los clasificadores base. La llave para producir un ensamble exitoso, es elegir los métodos de clasificación apropiados y seleccionar los clasificadores base indicados para el problema planteado.

2.5 Definición de Clasificador

Un clasificador es una función

$$D : R_n \rightarrow \Omega \qquad (2,6)$$

En el "**modelo canónico de un clasificador**" [26], se consideran un conjunto de *c* funciones discriminantes $G = g_1(x), ..., g_C(x)$,

$$g_i : R^n \rightarrow R \qquad (2,7)$$

$i=1,\ldots,c$

Cada uno produciendo un puntaje para la clase respectiva. Por lo general, x está etiquetado en la clase con la puntuación más alta. Esta elección de etiquetado se denomina la regla de máxima afiliación, la cual se describe a continuación:

$$D(x) = w_{i*} \in \Omega \leftrightarrow g_{i*}(x) = \min_{i=1\ldots c}\{g_i(x)\} \qquad (2,8)$$

Donde w_i son las características de cada instancia, Ω es el dominio del conjunto de características. g_i son las funciones discriminantes de cada clase o etiqueta. La ecuación 2.8, obtiene la clase de la función que minimiza su valor, es decir, la instancia *x* se asignará a la clase que minimice el valor de g_i. Los empates se rompen al azar, es decir, *x* se asignan al azar a una de las clases.

2.6 Clasificadores Base

Una vez teniendo las representaciones del corpus podemos clasificar las instancias, para lo cual es importante conocer la definición de clasificación.

La Clasificación es la tarea de predecir una variable discreta *"y"* usando un conjunto de características $x_1, x_2, ..., x_n$ como variables independientes. Para realizar el entrena- miento del clasificador se necesita una función hipótesis *h* de una colección de ejemplos de entrenamiento. Dicha colección tiene la forma *(X,Y)* y usualmente se refiere a un conjunto de datos *(dataset)*. Cada entrada del conjunto de datos es una tupla *(x,y)*, donde *x* es el conjunto de características y *y* es la clase o etiqueta la cual es una variable discreta con *c* posibles categorías. Cuando los resultados posibles son restringidos a valores binarios,
$y_i \varepsilon +1,-1, \forall i \varepsilon 1,...,N.$

En la figura 2.2 se muestra la partición del conjunto de datos en n particiones, que son entrada del algoritmo de aprendizaje, el cual utiliza una función hipótesis para realizar la clasificación.

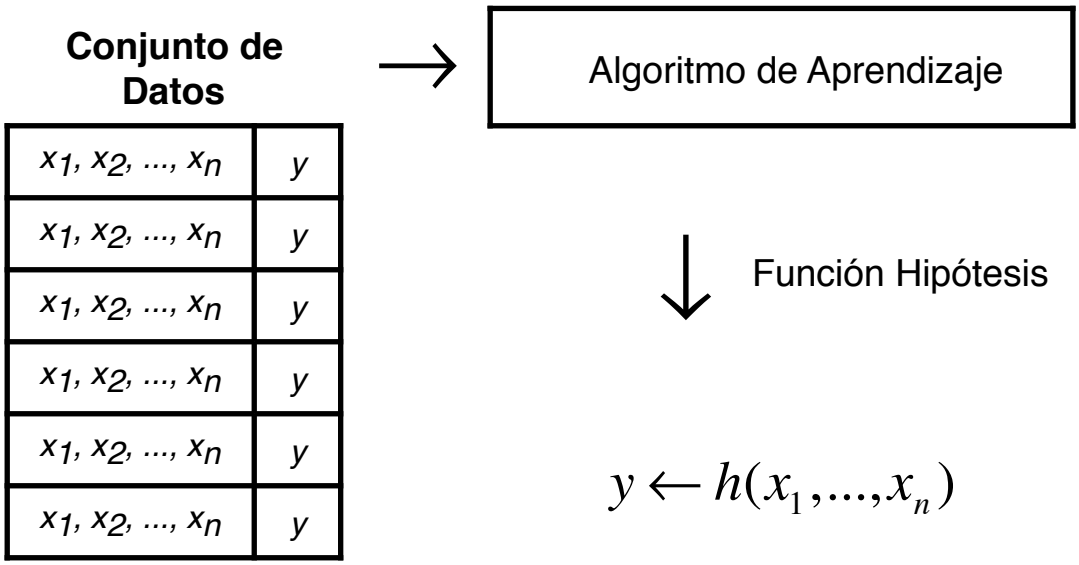


Figura 2.2 Partición de los datos y función hipótesis del algoritmo de aprendizaje.

Se utilizan 3 algoritmos de aprendizaje supervisado,Naive Bayes y Árboles de Decisión y Máquina de Soporte Vectorial. Cada uno de estos algoritmos se describe brevemente a continuación:

2.6.1 Naive Bayes

Es un clasificador probabilístico que aplica el Teorema de Bayes para estimar la probabilidad posterior *P(y | x)* de la clase y dada la variable *x*

$$P(y|x) = \frac{P(y|x)P(y)}{P(x)} \tag{2.9}$$

Naive Bayes está centralizado en las probabilidades *P(x | y)* que se refieren a la verosimilitud y representan la probabilidad de observar el valor *x*, dado el valor de clase *y*. Debido a esto Naive Bayes es considerado un **clasificador generativo**.

De la ecuación 2.9 se puede observar que el denominador *P(x)* es constante para algún valor de *y*, por lo que no es necesario calcularlo en orden de tomar una predicción. Por lo tanto podemos usar la siguiente aproximación:

$$P(y|x)P(y|x)P(y) \tag{2.10}$$

El valor *P(y)* se refiere a la probabilidad anterior y puede ser estimada directamente por los datos. Sin embargo *P(x / y)* depende de la distribución conjunta de *x* dado *y* . Y dado que *x* es una variable aleatoria **multivariable**, *P(x / y)* es muy costoso de estimar.

De acuerdo a la regla de la cadena, la distribución conjunta de *P(x / y)* puede ser expresada de la siguiente manera:

$$P(x_1,...,x_n|y) = P(x_1|y)P(x_2|x_1,y)...P(x_n|x_{n-1},...,x_2,x_1,y) \qquad (2.11)$$

A manera de evitar la costosa estimación de *P(x / y)*, el clasificador considera una fuerte suposición, que todos los pares de características x_i y x_j son independientes para cada evidencia de *y* dada. De esta manera se tiene $P(x_i / x_j, y)=P(x_i / y)$ para algún par $i,j \varepsilon [1,n]$. Por lo tanto la función de verosimilitud puede ser representada de acuerdo a la siguiente expresión:

$$P(x|y) = P(x_1|y)P(x_2|y)...P(x_n|y) = \prod_{i=1}^{n} P(x_i|y) \qquad (2.12)$$

De esta manera las probabilidades $P(x_i / y)$ pueden ser estimadas directamente de los datos.

2.6.2 Árboles de Decisión

Un árbol de decisión describe un conjunto de reglas organizadas de forma jerárquica, que implementan una **estructura de decisión**. Se compone de hojas y nodos. Una hoja registra una respuesta (clase) y un nodo especifica algunas condiciones de las pruebas que se llevarán a cabo en un valor único, rasgo de una instancia, con una rama y sub-árbol para cada posible resultado de la prueba. Para un determinado vector, se toma una decisión partiendo de la raíz de un árbol, y se mueve a través del árbol determinado por el resultado de una prueba de estado en cada nodo, hasta que se encuentra una hoja. El proceso de construcción de un árbol de decisión es una partición recursiva de un conjunto de entrenamiento.

A continuación se mostrarán algunas de sus características

- Si todos los objetos son **distinguibles**, entonces podemos construir un clasificador árbol con error de re-sustitución cero. Esto así permite que el árbol sea capaz de memorizar los datos de entrenamiento, para que pequeñas alteraciones pudieran conducir a un clasificador árbol estructurado de manera diferente. La inestabilidad puede ser una ventaja más que un inconveniente cuando se consideran los conjuntos de clasificadores

- Los clasificadores de árboles son intuitivos porque el proceso de decisión puede ser rastreado como una secuencia de **decisiones simples.**

- Para tal método son adecuadas las características cuantitativas y cualitativas, Con un pequeño número de categorías son especialmente útiles porque la decisión puede ser fácilmente diversificada. Para características cuantitativas, un punto de división tiene que ser encontrado para transformar la función en datos categóricos. Los árboles de decisión no se basan en un concepto de distancia en el espacio de características. Son principalmente útiles cuando se tienen características categóricas o mixtas. Por esta razón los árboles de decisión se consideran como **métodos no métricos para la clasificación**.

2.6.3 Máquina de Soporte Vectorial SVM

Del inglés Support Vector Machine (SVM) es un clasificador binario discriminante, dirigido a encontrar el hiperplano óptimo *($w^T * x+b$)* que separa los dos posibles valores de la variable etiquetada *y* $\varepsilon\{+1,-1\}$, de acuerdo al espacio de características representado por *x*. El hiperplano óptimo es aquel que maximiza el margen entre las instancias positivas y negativas en el conjunto de datos de entrenamiento formado por *N* observaciones. La tarea de aprendizaje de una SVM se formaliza con el siguiente problema de optimización:

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \varepsilon_i \qquad (2.13)$$

sujeto a $y_i(w^T x_i + b) \geq 1 - \varepsilon_i \forall i \in \{1,...,N\}$

$$\varepsilon_i \geq 0, \forall i \in \{1,...,N\}$$

El objetivo del problema se enfoca en dos aspectos, el primero, obtener el máximo margen en el hiperplano y minimizar el error $\sum_i^N xi_i$ .El parámetro C se refiere al parámetro suave de regularización de margen y controla la sensibilidad de la SVM para los posibles **valores atípicos.**

También es posible hacer que las SVM encuentren patrones no lineales, de manera eficiente usando el kernel trick. Una función $\phi(x)$ que realiza el mapeo del espacio de características x usando un espacio dimensional alto conocido como el espacio de Hilbert, donde el producto punto $\phi(x)\phi(x')$ es conocido como la función kernel $K(x,x')$. De esta manera, el hiperplano es calculado en un espacio de dimensión alta $w^T \phi(x) + b$. La formulación dual de las SVM permite reemplazar cada producto punto por una función kernel como se muestra en la siguiente expresión:

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \qquad (2,14)$$

Sujeto a $\alpha_i \geq 0, \forall i \varepsilon \{1,...,N\}, \sum_{i=1}^{N} \alpha_i y_i = 0$

Donde los parámetros $\alpha_i,\ i\ \varepsilon\{1,...,N\}$ corresponde a los multiplicadores de Lagrange del problema de optimización. Una vez que los parámetros se determinaron, es posible clasificar nuevas observaciones x_j de acuerdo a la siguiente expresión.

$$sign\left(\sum_{i=1}^{N} \alpha_i y_i K(x_i, x_j) + b\right) \qquad (2,15)$$

2.7 Ensamble de Clasificadores

Lo primero que se deberá realizar para construir un ensamble de clasificadores involucra el proceso de generación de una colección de clasificadores base, un enfoque es aplicar *N*-diferentes métodos de aprendizaje, con un solo conjunto de datos de entrenamiento, para obtener *N*-diferentes modelos de clasificación [27].

Otro enfoque es crear *N*-diferentes particiones de los datos de entrenamiento y emplear un solo algoritmo de aprendizaje con cada partición [28]. El principal problema en este enfoque es la conversión del conjunto de datos originales en una colección de diferentes conjuntos de datos de entrenamiento. En algunas técnicas, el conjunto de datos original está dividido en *N* subconjuntos seleccionados aleatoriamente. Otros trabajos involucran la manipulación de los datos de acuerdo a la distribución de los mismos.

Dado el potencial uso del ensamble de clasificadores, existen algunos factores que deben ser diferenciados entre los varios métodos de ensamble. Los principales factores se listan a continuación:

- 1. Relación inter-clasificadores. ¿Cómo cada clasificador afecta a otros clasificadores? Los métodos de ensamble pueden ser divididos en dos principales tipos: secuenciales y concurrentes.

- 2. Método de combinación. La estrategia de combinar los clasificadores generados por un algoritmo de inducción. El combinador simple determina la salida exclusiva- mente a partir de las salidas de los inductores individuales.

- 3. Generador de diversidad. Con el objetivo de realizar un ensamble eficiente, debe existir diversidad entre los clasificadores involucrados. La diversidad puede ser obtenida a través de diferentes presentaciones de entrada de datos, como en bagging, variaciones en el diseño de aprendizaje, aplicando una sanción a las salidas para fomentar la diversidad.

A continuación se describirán las técnicas más populares de ensamble de clasificadores.

2.7.1 Cascada

También conocida como generalización de cascada, es una arquitectura para combinar clasificadores como se muestra en la figura 2.3, puede presentar *n* niveles, sin embargo normalmente presenta dos niveles, en donde el nivel 1 es entrenado con el conjunto de datos original, el nivel 2 con un conjunto de datos aumentado, el cual contiene las características del conjunto de datos original junto con la salida del clasificador del nivel 1. La salida del clasificador del nivel 1 es un vector que contiene la distribución de probabilidad condicional $(p_1,...,p_C)$, donde c es el número de clases del conjunto de datos original, y p_i es la estimación de probabilidad calculada por el clasificador del nivel 1, de que la instancia pertenezca a la clase *i*.

El entrenamiento del clasificador del nivel 2 es influenciado por el clasificador del nivel anterior, debido a que considera su salida obtenida, derivando un esquema global sobre entrenado. Sin embargo, en cascada se reduce este problema debido a dos razones: en cada nivel se utiliza un clasificador de diferente naturaleza al otro y además el clasificador del nivel 2 no se entrena únicamente con la salida del clasificador de nivel 1, sino que además tiene en cuenta las características originales.

Esta técnica combina dos clasificadores, seleccionando aquel clasificador que obtenga un error bajo de bias, y otro con valor de varianza baja también, para conseguir uno nuevo que tenga valores bajos en ambas medidas. En el trabajo realizado en [29] se ocupa el clasificador con error de varianza baja en el nivel 1 y el clasificador con error de bias baja en el nivel 2, debido a que seleccionando métodos con bajo bias en el nivel superior, es posible ajustarse áreas de decisión más complejas, teniendo en cuenta las superficies estables, trazadas por el clasificador o los clasificadores de nivel inferior. En [29], se realiza una validación experimental utilizando 26 conjuntos de datos del repositorio de la universidad de California, Irvine *(University of California, Irvine UCI)* [30], que da soporte a realizar el ensamble de clasificadores de ésta manera.

En la figura 2.3, se muestra el proceso de Cascada a dos niveles, en donde el clasificador del primer nivel (árbol), tiene como entrada los datos originales del problema y el clasificador del segundo nivel (SVM), los datos de entrada son los datos originales, pero también se agrega la salida del clasificador del primer nivel.

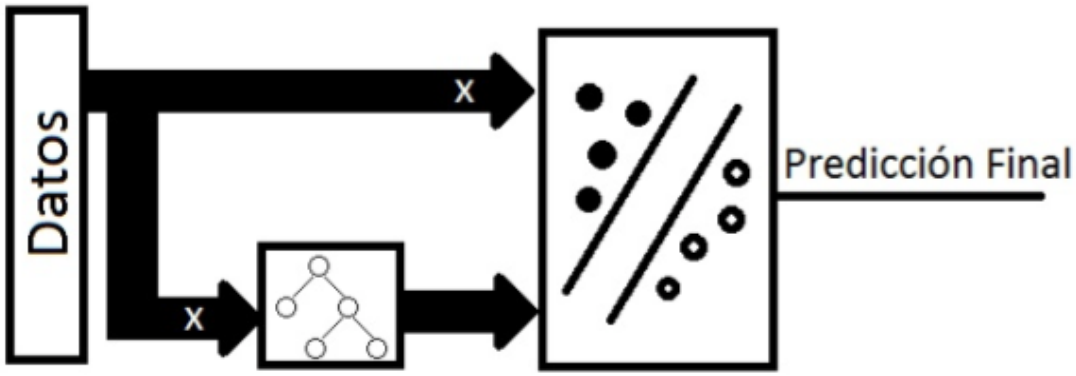


Figura 2.3 Estructura de Cascada de 2 niveles.

2.7.2 Mayoría de Votos

Es un método simple de combinación de clasificadores base, en el cual todos los clasificadores incluidos proveen un voto a alguna de las clases, el método realiza la sumatoria de dichos votos y la clase que recibe más votos es seleccionada como la decisión final.
Este método es representado por la siguiente ecuación:

$$x \rightarrow wifw = \arg(\underset{w\in\theta}{máx} \sum_{i=1}^{T} 1(C_i(x) = w)) \qquad (2,16)$$

$$1(C_i(x) = w) = \begin{cases} 1 & \text{si } C_i(x)=w \\ 0 & \text{en otro caso} \end{cases} \qquad (2,17)$$

x es una instancia, *θ* es el conjunto de etiquetas de clase, *w* es la clase asignada para la instancia *x* y $C_1,...,C_T$ son los clasificadores base.

Este método a pesar de ser tan simple ha sido ampliamente utilizado en diferentes áreas, un ejemplo de ello se describe en [31] donde es utilizado junto con la transformada de *"Haar"* para mejorar la efectividad de los sistemas de autenticación basados en reconocimiento de iris. Otro trabajo más donde también se utiliza el esquema de mayoría de votos se describe en [32], en el cual dicho método se utiliza para detectar canales cubiertos maliciosamente dentro de una red a través de un túnel DNS.

2.7.3 Ventanas

El método de Ventanas es una técnica general, que tiene por objetivo mejorar la eficiencia de los métodos de aprendizaje o clasificadores utilizados, mediante la identificación de un subconjunto adecuado de instancias de entrenamiento. Dicho método se lleva a cabo mediante el uso de un procedimiento de submuestreo.

El método funciona de la siguiente manera: Se selecciona un subconjunto aleatorio de instancias para el entrenamiento de un clasificador (una ventana), el resto de instancias son utilizadas para los datos de prueba, si la precisión obtenida del clasificador es insuficiente, las instancias de prueba clasificadas erróneamente se eliminan de las instancias de prueba y se añaden al conjunto de instancias para el entrenamiento en la siguiente iteración.

En 1993 [33] Quinlan propone dos formas diferentes de la formación de una ventana: en la primera, la ventana actual se extiende hasta un límite especificado. En la segunda, se identifican varios casos clave en la ventana actual y el resto son reemplaza- dos. Así, el tamaño de la ventana se mantiene constante. El proceso continúa hasta que se obtiene una precisión suficiente.

El método de ventanas también ha sido estudiado por Fürnkranz en 1997, [34] en donde se muestra que para este tipo de algoritmo, se pueden presentar mejoras significativas en la eficiencia, solo en dominios libres de ruido. En dicho trabajo se propone un versión de ventanas en donde se elimina de los datos de entrenamiento todos los casos que han sido clasificados correctamente, y se agregan todos los casos que han sido clasifica- dos erróneamente. La eliminación de instancias desde la ventana mantiene su pequeño tamaño y por lo tanto disminuye el tiempo de ejecución.

En conclusión, en ambos casos el método de ventanas construye una secuencia de clasificadores para obtener una muestra final. Es importante mencionar que ventanas no combina clasificadores, su tarea radica en mejorar el resultado de un clasificador.

Conclusiones del Capítulo

Hasta este momento se ha definido a la minería de datos formalmente, partiendo del concepto de la extracción de la información además de las técnicas de clasificación existentes, es muy importante denotar que desde la representación de los documentos hasta el proceso que debe ser involucrado para aplicar funciones clasificadoras; se deberán tener en cuenta todo el tiempo en el que se escribe la librería las funciones fundamentales y sus algoritmos para representar cada módulo.

Además de la formalidad para definir a los clasificadores, que son la parte principal de escribir esta librería, se hicieron notar las particularidades que en términos computacionales como el tiempo de ejecución y el costo de cada operación; es posible conocer lo límites de la misma.

Durante la redacción de este capítulo se ha podido fundamentar y dejar claro que para realizar un análisis de datos, se deberán tomar en cuenta algunos pasos intermedios que requieren uso de algunas tecnologías actuales; bases de datos, el uso de archivos de diversos, algoritmos especiales. Si la idea principal es realizar text mining se observaron diferentes y amplios conceptos que deben tomarse en cuenta antes de escribir cualquier código. Como resultado, esta librería de python será capaz de analizar texto ya sea utilizando algunos módulos de algoritmos específicos y en conjunto con una API dependiendo del servicio de información que sea requerido; además de mostrar la información en algún tipo de reporte.

Capítulo 3 Objetivo del Proyecto y Herramientas de Software

Para el desarrollo de este proyecto, no solo se pensó en la forma en la que se escribiría la librería; sino en los diferentes escenarios en dónde sería útil. La principal motivación de esto es destacar el buen uso de la información en especial el texto que proviene de redes sociales. Se aplicarán algoritmos de diversas utilidades y esta librería será escrita en python 2.7. La decisión del uso de esta versión de python fue hecha debido a la disposición de componentes de software que pueden implementarse, algunas versiones superiores no cubren la compatibilidad necesaria para el desarrollo de este proyecto, como la 2.3.

En principio la idea de este proyecto es la correcta extracción de información, es decir los tipos de datos que podremos obtener de diversas fuentes al utilizar esta librería de python, el análisis de dicha información es importante y será necesario implementar herramientas de soporte en las cuales se puedan mostrar gráficas, reportes y algunos tipos de representación de información textual específicos.

3.1 Objetivos del Proyecto

El objetivo principal de este proyecto fue desarrollar un conjunto de programas en python que apliquen las técnicas de text mining.en diversos escenarios, el principal es aplicando minería de opinión en redes sociales, comenzando con Twitter dada a su abstracción y a su API establecida. Estos programas conformarán una librería que hará referencia a algoritmos y técnicas de la minería de textos. Es decir existe un programa llamado por ejemplo: “Bag_of_words.py” que es código en python que podrá hacer una o más soluciones al algoritmo de *bag of words,* en general se planteó que las principales técnicas del text mining funcionen en esta librería.

Un objetivo específico hace referencia a utilizar python por su gran utilidad de scripting, así se escribió menos código utilizando módulos y referencias a alguna API establecida. Este objetivo destaca en este proyecto, debido a qué la cantidad de código no es la principal razón de elaborar una librería, sino el resultado y las aplicaciones que esta tiene.

Otro objetivo importante es el uso de la información y su representación, el uso de herramientas que permitan elaborar reportes detallados de información mediante

gráficas y visualizaciones de texto, así el alcance de este proyecto se podría potenciar.

Por lo tanto, el paso inicial fue elegir las herramientas de software necesarias para que todo esto fuera posible, no sólo en términos de desarrollo sino de mejor solución en términos de visualización. Para esto fue necesario limitar el alcance, conocer las posibilidades del uso de cierto software y si era compatible con diversas plataformas. En términos generales, las herramientas de software utilizadas colaboran entre sí.

3.2 Redes Sociales

Se podría definir a las redes sociales como estructuras sociales compuestas por personas interconectadas entre sí por relaciones de tipo diverso tales como amistad, parentesco o intereses comunes.

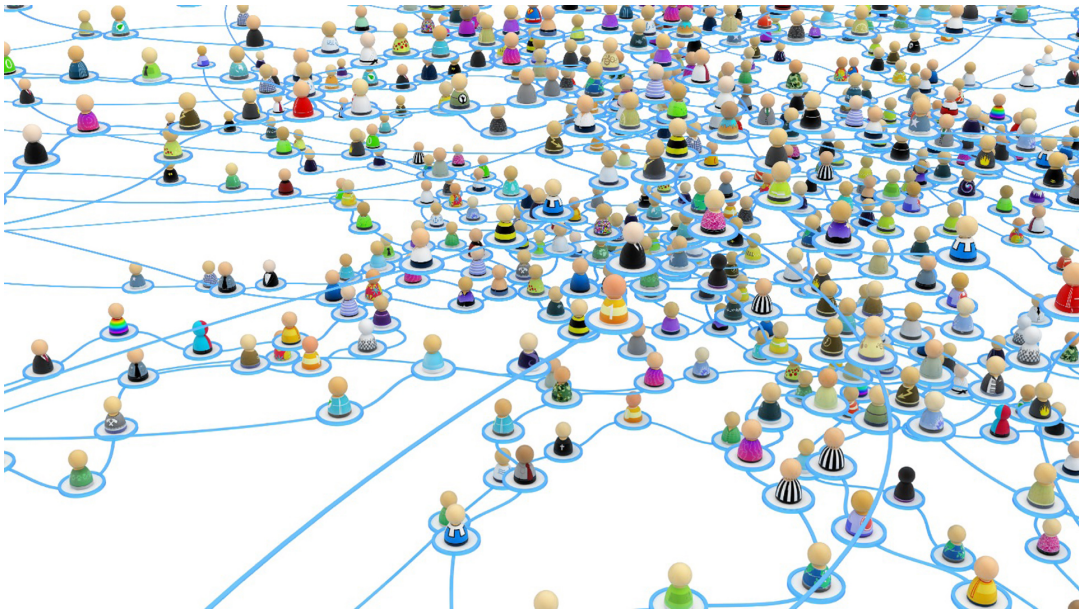


Figura 3.1 Representación de una red social.

Tal y como ocurre en la vida real, esta definición de red social se puede trasladar a Internet, surgiendo así los SNS (Social Networking Service or Social Networking Site), sitios web que ofrecen sus plataformas para construir nuevas relaciones sociales entre personas que comparten intereses, actividades, ideas o incluso relaciones personales o de parentesco.
Habitualmente en estos servicios cada persona posee un perfil con información básica como fecha de nacimiento, foto principal del perfil o enlaces de interés, junto con una serie de servicios adicionales variados entre los que destaca la

mensajería instantánea a veces complementada con vídeo para interactuar en tiempo real y enriquecer la experiencia.

Las redes sociales se basan en una gran medida en la teoría conocida como seis grados de separación, propuesta inicialmente en 1930 por el escritor húngaro Frigyes Karinthy, aunque no sería hasta el año 1967 cuando el psicólogo estadounidense Stanley Milgram probó la teoría mediante el experimento del mundo pequeño [35]. Así, todos los habitantes del mundo estaríamos conectados entre nosotros con un máximo de seis personas como intermediarios en la cadena. El concepto está basado en la idea de que el número de personas conocidas en la cadena crece exponencialmente conforme el número de conexiones crece, dando por resultado que sólo se necesita un número pequeño de estas conexiones para formar una red que nos conecta a todos.

Los SNS (Social Networking Service or Social Networking Site) se pueden clasificar en dos tipos:

- Generalistas, cuyo propósito es el de comunicar a sus miembros permitiendo que estos compartan información de cualquier índole. Entre ellas destacamos a las tres grandes redes sociales del momento: Facebook, Twitter o Google+ que se encuentran presentes a nivel mundial, si bien existen diversos servicios locales en cada país con mayor o menos éxito, siendo un caso especialmente relevante el de China, que a pesar de ser la nación con más usuarios conectados a Internet del mundo [36], su censura política ha frenado la expansión de las grandes [37] [38], proliferando sitios alternativos que copan las visitas de los internautas chinos.

- De propósito especifico, cuando sus miembros comparten información sobre una temática concreta. Así nos encontramos con sitios como Youtube para publicar vídeos, Flickr, comunidad social formada por fotógrafos y aficionados, LinkedIn, servicio con una orientación profesional, para establecer relaciones laborales o Badoo, sitio web para conocer gente nueva.

3.2.1 Twitter

Se trata de un servicio de red social estadounidense creado en 2006 con la intención de facilitar el intercambio de mensajes de texto breves entre sus miembros.
Considerado por muchos como un servicio de *microblogging,* su principal virtud junto a la sencillez de uso y simplicidad, es la limitación de los mensajes o actualizaciones que reciben el nombre de tweet (trino, gorjeo) a 140 caracteres, lo

que ha supuesto que numerosos medios y autores describan esta herramienta como los ”SMS de Internet”[39].

Esta limitación en la longitud de los mensajes lo hace ideal para intercambiar contenido de forma rápida, el cual se almacena de manera persistente en los servidores de Twitter, y tiene su explicación en el origen de esta herramienta.

Twitter se comenzó a fraguar en 2006 en el seno de una compañía dedicada a los *podcast* (archivos multimedia por suscripción) como idea para comunicar a pequeños grupos de personas partiendo del concepto de los SMS (Short Message Service) en telefonía móvil [40]. A esta idea la llamaron *twttr* y tras un periodo de pruebas interno, fue lanzada al público el 15 Julio de 2006 [41].

Las relaciones entre los usuarios de Twitter puede ser de dos tipos: *following* y *follower*, que podría traducirse como seguidor y seguido. En efecto, se denomina *following* al conjunto de miembros a los que otro usuario sigue, suscribiéndose a sus publicaciones. Del mismo modo, estos usuarios seguidos pasan ahora a disponer de un nuevo seguidor o *follower*. Si dos usuarios se siguen mutuamente, se podría considerar entonces que hay una relación mutua de “amistad”, similar a la que se da en otras redes sociales [42].

Esto no tendría sentido sin el concepto de *timeline* o línea de tiempo que es el lugar de la red social en el que aparecen los contenidos de los usuarios. Cada miembro de la red dispone de una espacio personal o perfil en el que se muestran sus mensajes ordenados cronológicamente a modo de bitácora. Si el usuario ha iniciado en la plataforma, visualizará además todos los mensajes de sus usuarios favoritos, o personas a las que ha decidido seguir, si las hubiere.

Por si fuera poco, cada usuario puede compartir información de manera pública, en la que el mundo entero incluso fuera de la red social, tiene acceso a las publicaciones de un usuario o de forma privada, en el que sólo se otorgan permisos para visualizar los mensajes a un conjunto reducido de miembros: sus *followers*.

Para que los usuarios de Twitter interactúan entre sí, la red social pone a su disposición los siguientes mecanismos:

- Menciones. Es la forma en la que un usuario se dirige expresamente a otro de manera pública para después mantener una conversación o para notificar a este de algún contenido de interés. Siguen el formato @usuario seguido de un texto o contenido multimedia.

- Mensajes privados. Otra manera de comunicación entre usuarios, en esta ocasión de manera privada, esto es, no visible por el resto de usuarios o el público en general.

- *#Hashtags*. Son palabras o etiquetas que comienzan con un (#) y se emplean para agrupar los mensajes que versan sobre un determinado tema, para su posterior categorización. De esta forma se puede incluir un *#hashtag* en cada *tweet* para después rastrear todos los mensajes que contengan dicha etiqueta y descubrir lo que otros miembros opinan. Muy empleado en eventos y en programas de televisión para interactuar con la audiencia.

- *Trending topics*. Son las tendencias o temas de actualidad más repetidos en un determinado momento en Twitter, siendo clasificados a partir de los *hashtags*. Estas tendencias se muestran en la página de inicio de cada usuario pudiendo elegir el ámbito geográfico al que aplican: a todo el mundo o bien a una localización concreta. Especialmente utilizados por la prensa y los medios de comunicación, a menudo son empleados para medir la popularidad de un tema.

No hay duda de que el éxito de Twitter se debe en gran medida a la existencia de una API (Application Programming Interface) gratuita con limitaciones que ha posibilitado la creación de aplicaciones de terceros para conectarse a la red social e interactuar con ella, y para extraer y analizar información de la misma, siendo algunas de estas aplicaciones más populares que el cliente oficial facilitado por la propia empresa californiana.

Sin embargo, los últimos cambios introducidos en las políticas de uso del API [43] pretenden limitar el acceso a la plataforma y a la información que se obtiene de ella, causando gran malestar en los desarrolladores que tendrá n que adaptar sus aplicaciones o cerrarlas por completo,haciendo incluso campaña a favor de abandonar Twitter por otras redes sociales con API menos restrictivas

3.3 Elección de Herramientas de Software

Para la elección de estas herramientas, se tomó en cuenta la capacidad de cómputo requerida. Era necesario crear un bot para realizar pruebas al hacer diversos posts en Twitter, con una computadora que funcionara como servidor fue suficiente: Se eligió el servidor Dell PowerEdge sus características esenciales 20GB de RAM e Intel Xeon a 3.10GHz, suficiente para elaborar tareas en background.

Respecto a la compatibilidad de software, python funciona en todas las plataformas. Para el desarrollo de la librería se efectuó con estos requerimientos de software: Windows: Python v2.7.10 (Windows y Mac), Matlab v2014, MacOS: CodeRunner v1.3, Xcode v6.4.

3.3.1 Python

Python fue desarrollado por Guido van Rossum a finales de los años 80, es uno de los lenguajes de alto nivel más populares para los programadores. Está diseñado para ser a la vez legible y accesible. Python puede crear un programa en menos líneas de código que C++ o Java, lo que proporciona programas entendibles, ya sea que se trate de un pequeño o de un proyecto grande.

Python soporta el paradigma de orientación a objetos, y los paradigmas de programación funcional, imperativo y de procedimiento, debido a su elegante diseño y sintaxis simple. Por lo que es especialmente útil en proyectos donde intervienen más de un programador.

- Entre sus características, destacan:
- Orientación a objetos muy intuitiva.
- Modulable, incluyendo soporte de paquetes jerárquicos.
- Gestión de errores basado en excepciones.
- Soporte de meta clases, decoradores y tipado dinámico de los datos.
- Permite al desarrollador escribir sus programas de manera potente y rápida.
- Potente y extensa librería estándar y gran cantidad de módulos de terceros para casi cualquier tarea.
- Posibilidad de emplear módulos escritos en diferentes lenguajes como C, C ++ o Java.

Python soporta múltiples paradigmas de programación, como la programación orientada a objetos o la programación funcional e imperativa. Igualmente cuenta con una gestión de memoria automática similar a la de otros lenguajes interpretados como Ruby, Scheme, Perl o Tcl y se ofrece para múltiples sistemas operativos haciendo que un programa se comporte de la misma forma en varias plataformas siempre que cuenten con el intérprete apropiado.

- CPython es la implementación original que se ofrece en el sitio oficial de Python. Escrita en C.

- Jython, implementación escrita en Java que permite importar cualquier clase Java y compilar a *bytecode* de ese lenguaje.

- IronPython hace lo propio para .NET y Mono. Escrita en C#.

- PyPy, intérprete y compilador JIT escrito en Python, más rápido y eficiente que CPython.

Python está administrado por la PSF, una organización sin ánimo de lucro dedicada a la difusión de este lenguaje de programación, gestionando el desarrollo de nuevas versiones y la obtención de financiación. Python posee una licencia de código abierto, denominada Python Software Foundation License que es compatible con la GNU GPL a partir de la versión 2.1.1.

3.3.2 FreeLing

FreeLing es una librería de código abierto para el procesamiento multilíngüe automático, que proporciona una amplia gama de servicios de análisis lingüístico para diversos idiomas. FreeLing ofrece a los desarrolladores de aplicaciones de Procesamiento del Lenguaje Natural funciones de análisis y anotación lingüística de textos, con la consiguiente reducción del coste de construcción de dichas aplicaciones. FreeLing es personalizable y ampliable, y está fuertemente orientado a aplicaciones del mundo real en términos de velocidad y robustez.

Se pueden utilizar los recursos lingüísticos por defecto (diccionarios, lexicones, gramáticas, etc), ampliarlos, adaptarlos a dominios particulares, o dado que la librería es de código abierto desarrollar otros nuevos para idiomas específicos o necesidades especiales de las aplicaciones.

3.3.3 OpenNLP

Es un conjunto de herramientas basadas en aprendizaje máquina para el procesado de textos en leguaje natural escritas en Java y mantenidas por la propia Apache Software Foundation. Se podría decir que se trata de un conjunto de proyectos de código abierto relacionados con el procesamiento del lenguaje natural cuya función principal es estimular y facilitar la colaboración de investigadores y desarrolladores en este tipo de proyectos.

OpenNLP soporta la mayoría de las tareas de NLP entre las que se incluyen:

- Detector de oraciones.

- Segmentador de palabras o *tokenizer*.
- Buscador de nombres y entidades.
- Etiquetado POS (Part of Speech).
- Analizador sintáctico.

Todavía en desarrollo, OpenNLP se ofrece de forma gratuita bajo una licencia de tipo Apache versión 2.0 que permite la libertad de usarlo para cualquier propósito así como distribuirlo o modificarlo sin necesidad de hacer públicas las modificaciones aunque si se debe conservar un fichero con información sobre la licencia original.

Está disponible en paquetes pre-compilados para numerosos sistemas operativos junto con el código fuente de la aplicación el cual se puede compilar mediante la herramienta Apache Maven. Para ejecutar las aplicaciones es necesario contar al menos con la versión 5 del JRE (Java Runtime Environment).

3.3.4 LingPipe

LingPipe es una herramienta multipropósito de procesado de texto mediante técnicas de lingüística computacional. Escrita en Java, y por tanto multiplataforma, puede ser ejecutada en sistemas que cuenten con esta máquina virtual, siendo empleada para tareas como la búsqueda de nombres de personas, organizaciones o localizaciones, la clasificación automática en categorías de los resultados de búsqueda en Twitter o para sugerencias de corrección ortográfica.

Este software se emplea en numerosos proyectos de investigación en universidades de todo el mundo, presenta una arquitectura que promete ser eficiente, escalable, robusta y reutilizable, permite no sólo ejecutar aplicaciones de terceros, sino también desarrollar las nuestras propias utilizando sus librerías. Para ello provee los siguientes mecanismos:

- Soporte multi-idioma, multi-dominio, modelos multi-género.
- Entrenamiento con nuevos datos para nuevas tareas.
- Extensible mediante modelos: para cada idioma y corpus de entrenamiento.

- Analizador de sentimiento de dos dimensiones: de polaridad (negativo/positivo) y subjetividad (objetivo/subjetivo).
- Detector de frases.
- Etiquetado POS (Part of Speech).

Acompañando al software, en la página web del fabricante se ofrecen dos libros para iniciar al desarrollador al procesamiento del lenguaje natural con esta herramienta, junto con otros tutoriales entre los que se incluyen la detección de idioma o análisis de sentimiento.

Es un software comercial propiedad de la empresa Alias-i, Inc distribuido en varias ediciones dependiendo de nuestro perfil: *Royalty Free*, *Developer*, *Startup* o *Enterprise Server*, siendo la *Royalty Free* una versión limitada sin soporte pero gratuita. Esta licencia permite el uso de este software con fines de investigación, obligando a hacer públicos los datos procesados con LingPipe y limitando su uso en producción.

3.3.5 NLTK

NLTK responde a las siglas de Natural Language Toolkit. Se trata de un conjunto de módulos Python, datos lingüísticos y documentación para la investigación y el desarrollo de aplicaciones de análisis de textos mediante el NLP siendo muy apropiado para investigadores, estudiantes, educadores, lingüistas e incluso la industria. Esta es la razón por la que se emplea en numerosas universidades de todo el mundo para la enseñanza aplicada del procesamiento del lenguaje natural.

Aunque el software se centra en conjuntos de datos en inglés es posible trabajar con prácticamente cualquier idioma, empleando los modelos de datos y los corpus apropiados.
Para ello se proporcionan más de 50 *corpus*, ofreciendo además la posibilidad de crear y emplear nuevos modelos recopilados por los usuarios.

Entre las herramientas que proporciona NLTK se encuentran:

- Lectores de *corpus*.
- Segmentador de palabras, oraciones y párrafos.
- Lematizadores (análisis morfológico).
- Constituyentes básicos o *chunkers*.

- Etiquetadores POS (Part of Speech).
- Analizadores sintácticos.
- Árboles de decisión,clasificadores de entropía máxima, Bayesiano ingenuo(Naive Bayes).
- Estimadores: uniforme, máxima verosimilitud, Laplace, etc.
- Soporte de WordNet.

NLTK está desarrollado en Python, requiriendo el intérprete de Python en su versión 2.5 o superior, y se proporcionan versiones disponibles para Windows, Mac OS X, y Linux. Junto al software se acompaña documentación de gran calidad incluyendo publicaciones como el libro de NLTK, tutoriales y *wikis* que se complementan con un gran soporte de la comunidad a través de foros y listas de distribución.

Además cabe destacar que nos encontramos ante un software libre y gratuito realizado por la comunidad habiendo adoptado una licencia Apache 2.0, por la que se permite su libre utilización y modificación sin la obligatoriedad de liberar dichos cambios.

Por otra parte, la documentación publicada se distribuye en los términos de la licencia Creative Commons Attribution Noncommercial No-Derivative-Works 3.0 de los EEUU mientras que los corpus incluidos poseen diferentes licencias, todas ellas libres.

3.3.6 Tweepy

Se trata de un módulo escrito en Python para acceder e interactuar con la API de Twitter, desarrollado por Joshua Roesslein y liberado a la comunidad a través de GitHub mediante la licencia MIT

Tweepy es un envoltorio de la API REST de Twitter que encapsula las peticiones HTTP como funciones simples en Python de forma que el desarrollador no se tiene que preocupar de lo que haya por debajo. Gracias a esta librería es posible utilizar las funcionalidades de Twitter en nuestra aplicación, capturando los *tweets* públicos que coincidan con los parámetros especificados en cada experimento, empleando para ello el llamado Streaming API
Durante el desarrollo del proyecto, Twitter ha efectuado numerosos cambios en su API, desde la forma en la que se devolvían los resultados a la forma de autenticar las llamadas

3.3.7 OAuth

Uno de los cambios más sonados llevados a cabo por Twitter, es el de la forma en la que se autenticaban las llamadas a la API. Hasta Junio de 2013, estas llamadas podían autenticarse o bien con el nombre de usuario y la contraseña de cada miembro de la red social transmitido de forma segura mediante HTTPS o bien mediante OAuth, un protocolo abierto que permite una autorización segura para el uso de un API sin necesidad de introducir la contraseña.

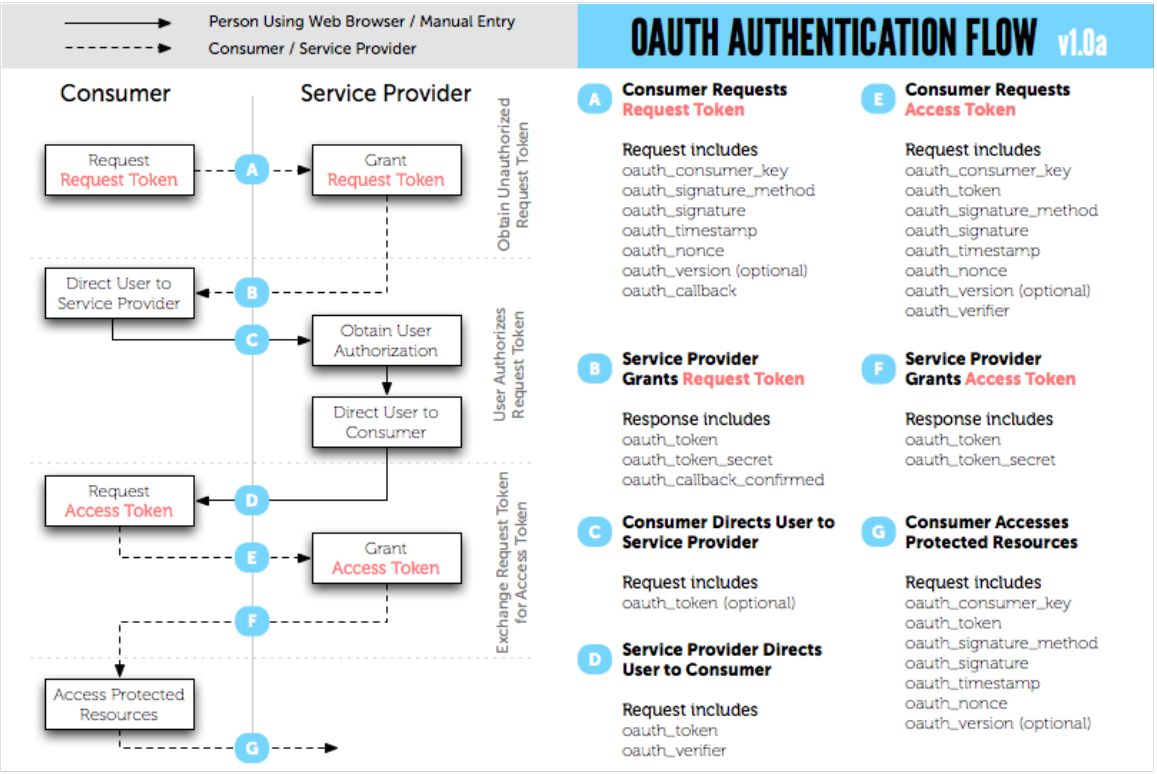


Figura 3.2 Diagrama de Autorización OAuth.

Con este protocolo de autorización, los usuarios de Twitter deben otorgar su consentimiento expreso a que las aplicaciones accedan a las funciones de su cuenta. Dicho consentimiento es otorgado o denegado en la propia web de la red social, a la que serán redirigidos durante la interacción con cada una de estas aplicaciones.

Cuando una aplicación es autorizada, Twitter le entrega las claves (**access token key** y **access token secret**) necesarias para realizar acciones en la red social en nombre del usuario sin necesidad de pedir su contraseña.

Este mecanismo de autorización tiene una gran ventaja para los usuarios, puesto que así evitan entregar la contraseña de sus cuentas a las aplicaciones, permitiendo también revocar el acceso otorgado previamente a cada aplicación en cualquier momento.

3.3.8 Wolfram Alpha

Es un buscador de respuestas desarrollado por la compañía Wolfram Research. Es un servicio en línea que responde a las preguntas directamente, mediante el procesamiento de la respuesta extraída de una base de datos estructurados, en lugar de proporcionar una lista de los documentos o páginas web que podrían contener la respuesta, tal y como lo hace Google. Fue anunciado en marzo de 2009 por el físico británico Stephen Wolfram y está en funcionamiento desde el 15 de mayo de 2009.

3.3.9 MySQL

En el modelo de datos relacional, los datos se almacenan en tablas diferentes entre las cuales se establecen una serie de relaciones siendo el álgebra relacional el conjunto MySQL es un sistema de administración de bases de datos relacional, multihilo y multiusuario propiedad de la empresa MySQL AB, subsidiaria de Sun Microsystems y absorbida por Oracle. A pesar de ello, el proyecto continúa siendo de código abierto aunque ha adoptado una doble licencia: GNU GPL con versiones gratuitas para proyectos de software libre y no comerciales o una licencia comercial para productos que redistribuyan MySQL y no hayan liberado su propio código fuente.

Este motor de bases de datos forma parte de la arquitectura MAMP (Apache, MySQL, PHP/Python/Perl) tan habitual en Internet, ofreciendo alto rendimiento, flexibilidad y escalabilidad, y es usado por organizaciones del tamaño de Google, Facebook, Wikipedia o Youtube.

Para poder realizar operaciones en bases de datos MySQL desde Python es necesario el empleo de un conector llamado MySQLdb, también gratuito y de código abierto empaquetado bajo el nombre python-mysqldb**.**

3.3.10 Javascript y jQuery

JavaScript, comúnmente abreviado como JS es un lenguaje de programación interpretado, derivado del estándar ECMAScript, que se emplea principalmente

como parte de un navegador web permitiendo interfaces de usuario mejoradas, transmisión asíncrona de datos y páginas web dinámicas.

Este lenguaje que sigue una sintaxis y convenciones similares a la de Java, con el que todavía hoy se le confunde, destaca por ser orientado a objetos y estar basado en prototipos, por tener un tipado débil y por ser dinámico.

jQuery es una librería de JavaScript que permite interactuar con los documentos HTML, manipular el árbol DOM (Document Object Model), gestionar eventos, animar elementos y aprovechar todas las posibilidades que ofrece AJAX de una manera rápida y sencilla, siendo compatible con todos los navegadores y soportando los últimos estándares web aprobados por el W3C (World Wide Web Consortium). Al igual que otras librerías, jQuery ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

Conclusiones del Capítulo

Este capítulo mostró las características y los posibles alcances que tiene una librería, el concepto de redes sociales de manera general contribuyó a establecer el primer paso para implementar el software deseado. La alta compatibilidad y el desarrollo en diversas plataformas y ambientes de desarrollo claramente facilitaron la construcción de este proyecto; además que puede estar presente casi en todas las plataformas.

Algunas ventajas de la realización de este proyecto: No está directamente asociado a una red social, es decir funciona con una red social en específico, pero diversas herramientas otorgan el soporte para que funcione en otros ambientes; es un hecho que diversos componentes de software se pueden integrar, utilizando las debidas autorizaciones en términos de API.

Conocer todas estas características y particularidades de software requeridos es fundamenta, con esto además de dar inicio a la escritura del proyecto, se crean los escenarios para realizar la experimentación pertinente. A partir de este momento, se prueba todo lo que se tiene que realizar para que las herramientas funcionen y proporcionen resultados.

Capítulo 4 Pruebas y Análisis de la Librería en Python

Una vez definidas las tecnologías necesarias, es decir la infraestructura, el ambiente y los alcances en términos de plataformas. Ahora que los objetivos han sido planteados de forma general y específica, es necesario aplicar ciertas pruebas y definir de un nivel superior lo que se ha hecho en esta librería.

Primero se realizaron pruebas para conocer el comportamiento de redes sociales, de datos extraídos de fuentes externas, de históricos y/o reportes que son necesarios para mostrar información. Si bien la idea no es crear una suite de text mining que sea capaz de proveer información de cualquier tipo dado un texto; en realidad la idea es implementar a través de python, funciones y procedimientos que trabajen en conjunto con otras plataformas y ambientes de desarrollo para aplicar técnicas de minería de textos. Aquí se proporciona la idea más general del comportamiento de la librería, la documentación necesaria de esta y todo su alcance en términos de funcionamiento.

4.1 Instalación y Configuración de Python

La versión requerida de Python es la 3.5, está se descarga desde su sitio oficial: https://www.python.org/downloads/ Una vez descargada, se debe evaluar que no existan otras versiones instalas en el sistema operativo, debido a que los componentes externos pueden presentar problemas de incompatibilidad si encuentran o hacen referencia a una versión diferente.

Para verificar nuestra instalación, ya sea en Linux, Mac o Windows. Solo es necesario escribir python en una terminal o línea de comandos. Es probable que después de la instalación esto sea posible debido a que el instalador de python escribe las variables de entorno correspondientes que hacen referencia a la versión de python instalada.

Nota: en Mac OS python es nativo para la ejecución de scripts de sistema operativo. por lo que al escribir python en la línea de comandos iniciará, pero siempre hay que verificar la versión, en el caso que sea una diferente a la instalada solo se tiene que buscar el perfil y crear un alias que haga referencia a la versión que se utiliza, en este caso es la 3.5.

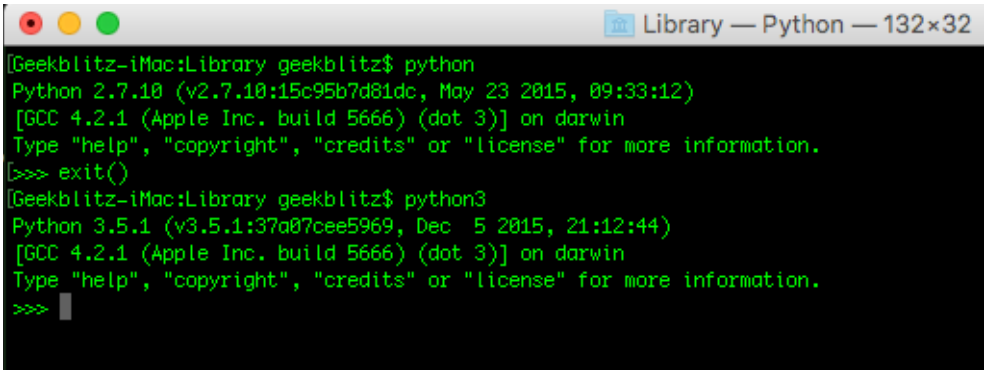


Figura 4.1 Versiones de Python instaladas.

Cada vez que se utilice python en la terminal de Mac, el alias *python3* será la versión 3.5.1 la que requiere este proyecto; y *python* será la nativa de Mac 2.7.10. Una vez instalado python, a continuación se instalan los componentes y módulos auxiliares.

En este proyecto se requieren 3: NLTK, Twython y Tweepy, estas herramientas pueden ser instaladas mediante la línea de comandos utilizando el comando pip de instalación, si el sistema operativo fuera alguna versión de Linux, con el comando apt-get install se instalan los componentes deseados.

4.2 Comando which

En las instalaciones de nuevos componentes y módulos de python es muy importante conocer la versión a la que se le instalará un componente, debido a que es un lenguaje de tipeado, los datos pueden hacer referencias a módulos existentes y tener conflictos al ejecutarse. El comando which <python_version>, o alias de python; lista los filesystems de las instalaciones correspondientes.

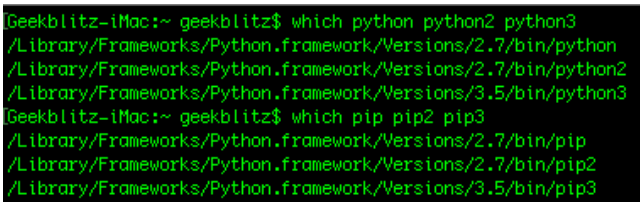


Figura 4.2 Filesystems de python con el comando which.

4.3 Módulos de Python para Text Mining

Algunas instrucciones requieren el comando pip o su alias de acuerdo a la versión de python a utilizar seguido del nombre del nombre componente. Para este proyecto se instalaron 3, twython para realizar pruebas en twitter al mismo tiempo que tweepy y NLTK para visualizar algunos algoritmos establecidos.

Comandos utilizados para la instalación:
twython: sudo pip3 install twython //necesita permisos de root en Unix, pip3 es un alias que hace referencia a la versión 3.

Figura 4.3 Instalación de twython.

tweepy: pip install tweepy==3.3.0 //pip es un alias que hace referencia a la versión 1, también es necesaria la versión de tweepy que se va a instalar.

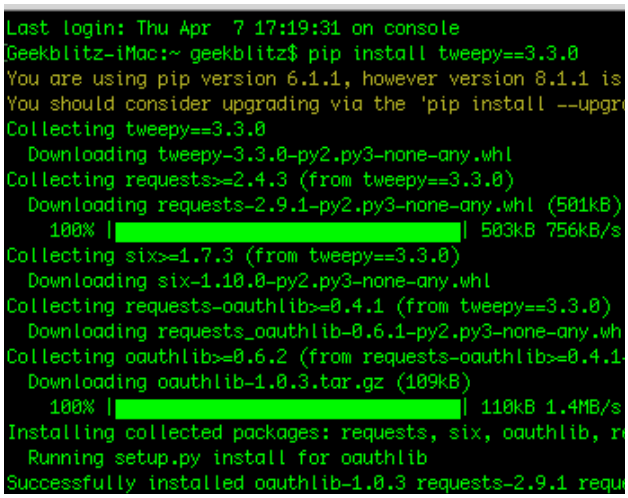


Figura 4.4 Instalación de tweepy.

NLTK: sudo pip install -U nltk //necesita permisos de root en Unix, pip es un alias que hace referencia a la versión 1, esto también puede actualizar el componente en el caso que haya uno existente.

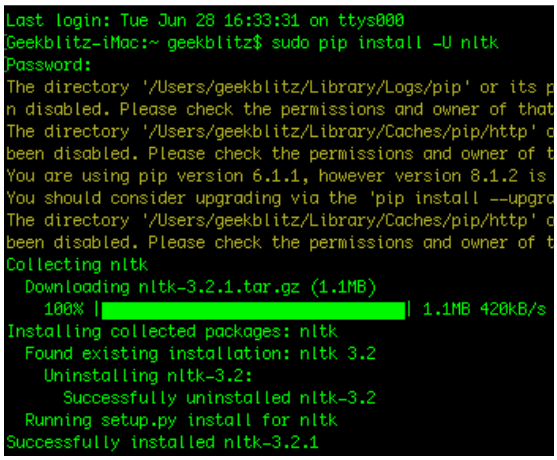


Figura 4.5 Instalación de NLTK

4.4 Requerimientos de Código

Antes de comenzar a escribir programas y realizar pruebas, es fundamental considerar lo que puede tener esta librería y todos sus componentes. También las posibles definiciones que requiere. En la tabla 4.1 se muestran los componentes de funcionamiento de la librería.

Herramienta	Tecnología Requerida
FreeLing	C++
OpenNLP	Java
Lingpipe	Java
NLTK	Python
Twython	Python
Tweepy	Python

Tabla 4.1 Herramientas de para la librería de text mining

Las herramientas de Natural Language Processing (NLP) son fundamentales para el desarrollo de software dedicado a la minería de textos, en un principio tendría que cubrir los algoritmos generales que denoten reglas y clasificadores. También se implementaron una serie de programas que utilizan estas herramientas, en la tabla 4.2 se muestran las opciones que ofrecen.

Función	**Herramientas NLP**			
	Freeling	**OpenNLP**	**Lingpipe**	**NLTK**
Segmentación de Oraciones	1	1	1	1
Segmentación de Palabras	1	1	1	1
Named Entity Recognition	1	1	1	1
Etiquetado POS	1	1	1	1
Análisis Sintáctico	0	0	1	1
Análisis Morfológico	1	1	1	1
Lectores de Corpus	0	1	1	1
Clasificadores	0	1	1	1
Estimadores	1	1	1	1
Desambiguación	1	0	1	1
Soporte multi-idioma	1	1	1	1
Extensible	0	0	1	1

Tabla 4.2 Funciones que aportan las herramientas NLP.

4.5 Modelo de Datos

Todos los sistemas que manipulen información, requieren de un modelo de datos, el cual ayuda a los desarrolladores a permitir la escalabilidad de la aplicación, programa o sistema propuesto. Para definir un modelo de datos en la librería, primero se obtuvo una visualización general de lo que puede contener, en este caso este modelo de datos es aplicable para realizar operaciones de text mining en twitter.

Como la información está bien definida de manera textual en twitter, resulta mucho más fácil abstraer el posible contenido que obtendremos, se sabe bien que no solo contenido textual está presente en la red social, puesto que su evolución ahora permite trasmitir videos en tiempo real, imágenes en formato gif y tweets que pueden ser de más de 140 caracteres utilizando un sistema que los deposita en url. Todas estas características claramente no se pueden procesar con los algoritmos tradicionales de minería de textos, pero el texto que es requerido para describir esta información si puede ser procesado.

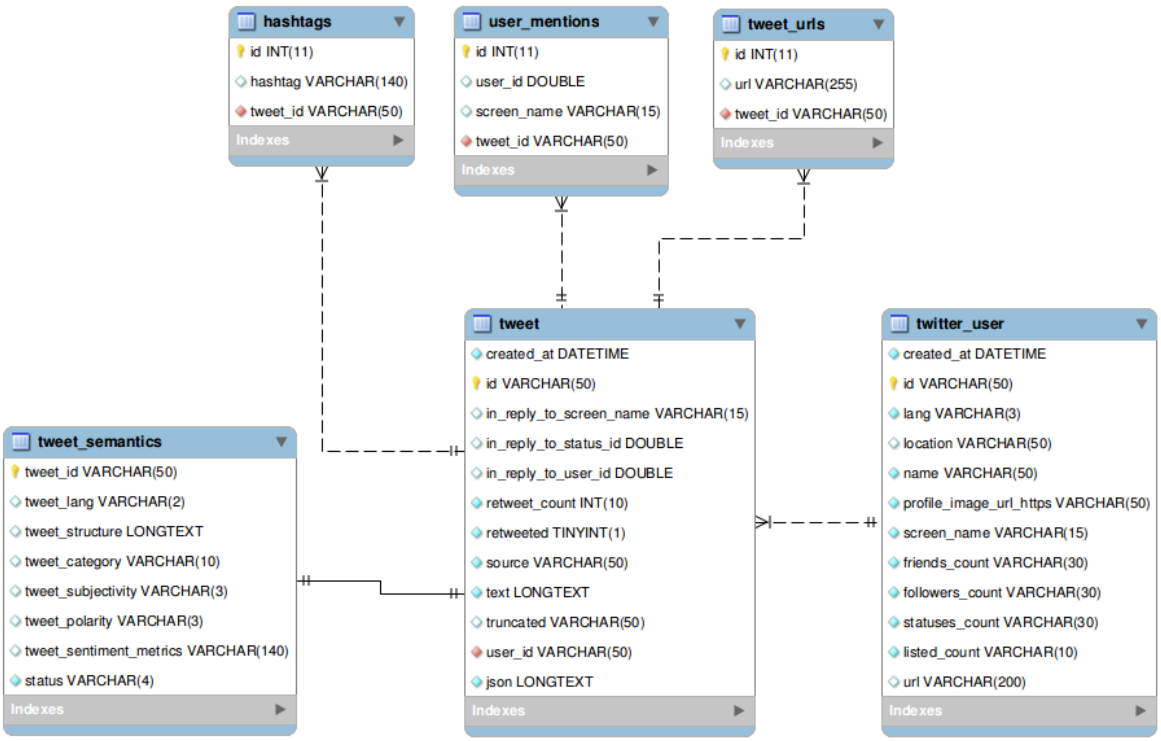


Figura 4.6 Modelo de Datos para la librería en Twitter

4.6 Captura de Tweets

El procedimiento de captura de *tweets* se encarga de interactuar con el API de Twitter empleando el Streaming API, que permite obtener publicaciones de esta red social en tiempo real de acuerdo a unos términos de búsqueda y a un tipo de consulta a elección de los usuarios.

Tal y como se muestra en la figura 4.7, la obtención de los mensajes se realiza a través de un *script* que recibe algunos parámetros y si estos son correctos se realiza una autenticación en Twitter y se procede a invocar la rutina de obtención de los mensajes teniendo en cuenta los límites definidos.

Es en esta rutina donde se realiza la detección de idioma antes de introducir los mensajes en la base de datos, y también donde se realiza un seguimiento de la métrica de *TPS (Tweets Per Second)* capturados en el experimento.

4.6.1 Detección del Idioma

Para el proceso de detección de idiomas se ha empleado el corpus de idiomas [32] que nltk pone a nuestra disposición, que contiene los *trigramas* más habituales en los principales idiomas.

Dependiendo del valor de n se puede decir que se forman *unigramas*, si n = 1, *bigramas* si n = 2 o *trigramas* si n = 3. Esto significa que los *trigramas* son asociaciones de 3 elementos, que en el caso del corpus de idiomas se efectúan a nivel de letras. Así, dicho corpus consiste en un listado de secuencias de tres letras junto con su frecuencia de aparición en numerosos idiomas y sobre él se opera

Así, para la detección de idiomas se separa cada *tweet* en palabras y sobre éstas se extraen los *trigramas*, que son comparados con los *trigramas* más habituales para los idiomas inglés y español. Esto nos plantea un problema y es que con este procedimiento lo que establece es una medida de parecido entre conjuntos de *trigramas* pero no se obtiene el idioma del texto en cuestión.

Establecer esa correlación con el resto de idiomas disponibles supone un aumento de la tasa de error, al haber *trigramas* comunes al español en idiomas como el catalán o gallego amén de un aumento del tiempo de procesado.

Para solventar este handicap necesitamos emplear una detección de idioma adicional la cual se lleva a cabo mediante la librería Chromium CLD (Compact

Language Detector), extraída del navegador de código abierto Chromium y que provee una interfaz Python para detectar el idioma de un texto.

Si tras este procesado los resultados coinciden, con mucha probabilidad nos encontraremos ante un texto escrito en tal idioma. Si no coinciden y la librería CLD detecta un idioma distinto de español o inglés, el *tweet* se descarta.

La detección de idioma se realiza en tiempo de captura de *tweets*, que si bien supone un coste algo mayor en cuanto a carga de procesado, nos evita realizar un post-procesado y filtrado de los *tweets* y garantiza que prácticamente todos los mensajes de nuestra base de datos cumplan el requisito de idioma.

En el caso de los nombres, a pesar de que NLTK anuncia soporte para la técnica NER (Named-Entity Recognition), que pretende detectar y reconocer nombres de entidades en textos a partir de un corpus etiquetado, parece que no es del todo fiable, no habiendo sido posible emplear este mecanismo y recurriendo a un fichero con un listado de nombres propios contra el que se efectúa la comprobación.

4.6.2 Categorización

La clasificación mostrada en el estudio se había llevado a cabo de forma exclusivamente manual debido a la cantidad de divergencias que había entre los diferentes miembros del equipo a la hora de clasificar los mensajes, ya que, según el individuo, el mismo *tweet* podía pertenecer a diferentes categorías.

Se podría decir por tanto que la clasificación en las categorías descritas en la publicación es algo subjetivo y por tanto muy difícil de automatizar.

Se han definido las reglas para que un *tweet* pertenezca a una determinada categoría:

- Questions to followers: *tweets* que contengan alguna pregunta, excepto si la cuestión está al principio del mensaje y existe un texto a continuación, habitualmente una opinión, en cuyo caso la categoría será *Opinions/ Complaints*.

- Location: mensajes enviados a través de clientes de servicios de localización como Foursquare/Swarm.

- Information Sharing: *tweets* que contienen URLs ya sean con o sin redirección, imágenes o vídeos. Citas de autores. *Retweets*.

- Self Promotion: URLs adjuntas a un mensaje cuyo dominio coincide con la página web listada en perfil de usuario (si existe).

- Anecdote me: *tweets* en pasado cuyo sujeto es la primera persona del singular (yo). Anecdote others: mensajes en pasado cuyo sujeto es la tercera persona del singular o plural.(él, ella, ellos).

- Me now: *tweets* con verbo en gerundio cuyo sujeto es la primera persona del singular (yo).

- Presence maintenance: mensajes que no tienen predicado.

- Random thoughts: *tweets* cuyo análisis sintáctico arroje sujeto y predicado con una conjugación diferente a las anteriormente descritas.

- Opinions/Complaints: todo aquello que no encaje en las categorías anteriores. Las opiniones, junto con la compartición de información, conforman la mayoría de los mensajes enviados a la red social Twitter, por lo que un gran número de mensajes pertenecerán a esta categoría.

4.7 Implementaciones, PyText

PyText es el nombre de la librería. A continuación se muestra una serie de implementaciones en código de los módulos que la forman, la idea y concepto general de todo esto fue ilustrar todo el trabajo que se hizo durante este proyecto. Hasta este punto se puede afirmar que se ha escrito una librería funcional, dado a que puede aplicar una técnica de text mining en una red social.
El código que se presentó contiene 4 caracteres al final de cada ConsumerKey, ConsumerSecret, AccessToken y AccessTokenSecret; esto para proteger la información debido a que es un tipo de password, estos provistos por Twitter en su sección de desarrolladores.

4.7.1 PyText: tweet_streaming

```
# -*- coding: utf-8 -*- #Codificación que permite acentos
##   PyText tweet_streaming #Nombre del programa
##  Created by Martín G. Morgado for MOVIS lab.#Author
##  Copyright © 2016 geekblitz. All rights reserved. #No Piracy

#Twython wrapper de Twitter https://pypi.python.org/pypi/twython
#Twitter libraries https://dev.twitter.com/docs/twitter-libraries
#Importando el objeto twython
from twython import Twython
```

```
import sys
#Definiendo las 4 cadenas creadas al utilizar OAuth
#Las keys se encuentran en los ajustes de la twitter app creada
ConsumerKey = "XXXXqJ9zzaDcEICz9BxDxXXXX"
ConsumerSecret = "XXXXbJkIiXCwcZVWl0Ig5Ed1EeJqgQHyR0g2Wf76tPxKuXXXX"
AccessToken = "XXXX8316-TTJiiR4lsWdGf7SFCRRu1nwEJxVP92X2Gzh1QXXXX"
AccessTokenSecret = "XXXXrgb21TfuoBSjT4g9N5zYUd3XZ8jR1k0KeSrwXXXX"
#Instanciando el obj Twython dando acceso a la cadena token como
parámetros
twitter = Twython(ConsumerKey,
     ConsumerSecret,
     AccessToken,
     AccessTokenSecret)
#Hasta esta parte, todos los programas se definen de la misma forma.
#Elaborando la búsqueda especificando texto query en el argumento keyword
q
print ("Escribe el término a buscar:")
#Escribe el término
#python 2.x raw_input() 3.x input()
result = twitter.search(q = (input()) )
#Iterando en la lista de resultados e imprimir cada status (tweet)
for status in result["statuses"]:
     #Escribir dirtecto al archivo
     #sys.stdout=open("streaming.txt","w")  //escribir en el archivo
     print(status)
#Restringiendo el resultado navegando la estructura del JSON, obteniendo
solo "usuario" y "texto"
#for status in result["statuses"]:
     #print("user: {0} text: {1}".format(status["user"]["name"],
status[“text"]))
```

4.7.2 PyText: bag_of_words

```
#Abrir el archivo en la ruta actual
f = open('example_text.txt', 'r')
#Imprimir archivo
#print ("\n", f.read())
#Procesando el archivo utilizando NLTK
#enunciado = ("These techniques and processes discover and present
knowledge. ")
#Procesa el enunciado
#print (nltk.word_tokenize(enunciado))
#Procesa e imprime el resultado del archivo
print ( nltk.word_tokenize(f.read()) )
```

4.7.3 PyText: followers_list

```
#Obtener la lista de seguidores
print ("Obtener la lista de seguidores de @: \n")
followers = twitter.get_followers_list(screen_name= (input()) )
#Iterar la lista de followers
for follower in followers["users"]:
     print(" {0} \n ".format(follower))
```

4.7.4 PyText: timelines

```
#Obteniendo nuestra timeline de twitter
timeline = twitter.get_home_timeline()
#Para obtener timeline de 25 tweets de un @usuario específico
#tl = twitter.get_user_timeline(screen_name = "geekblitz", count = 25)
#for tweet in tl:
#    print(" User: {0} \n Created: {1} \n Text: {2}
".format(tweet["user"]["name"], tweet["created_at"], tweet["text"]))
#Iterando la timeline imprimiendo usuario y texto
for tweet in timeline:
     print(" User: {0} \n Created: {1} \n Text: {2} "
     .format(tweet["user"]["name"],
     tweet["created_at"],
     tweet[“text"]))
```

4.7.5 PyText: trending_topics

```
#Para obtener las tendencias locales requiere un ID utilizando WOEID
#WOEID utiliza Yahoo! Query Language (YQL) como si fuera SQL
#Ejemplo de query select * from geo.places where text="Mexico, MX"
result = twitter.get_place_trends(id = 23424900)
if result:
     for trend in result[0].get("trends",[]):
          print("{0} \n".format(trend["name"]))
```

4.7.6 PyText: user_search

```
print ("Escribe el término a buscar: ")
search_results = twitter.search(q=(input()), count=20)
#print(search_results)
```

```
try:
    for tweet in search_results["statuses"]:
        screenname = "@" + tweet['user']['screen_name']
        print(screenname)
except TwythonError as e:
    print ("Error: " + e)
```

4.7.7 PyText: followers_text_list

```
print ("Obtener la lista de seguidores de @: \n")
username = input()
followers = []
next_cursor = -1
while(next_cursor):
     get_followers = twitter.get_followers_list(screen_name=username,
count=200, cursor=next_cursor)
     for follower in get_followers["users"]:
          followers.append(follower["screen_name"].encode("utf-8"))
          next_cursor = get_followers["next_cursor"]
print  ( str(len(followers))," followers found")
```

4.7.8 PyText: term_retweet

```
#La búsqueda de twitter como variable, y hace RT
print ("Escribe el término a retwittear: ")
search_results = twitter.search(q=input(), count=10)
try:
    for tweet in search_results["statuses"]:
        twitter.retweet(id = tweet["id_str"])
except TwythonError as e:
    print (e)
```

4.7.9 PyText: new_tweet

```
#Solicito y escribo lo que deseo publicar
print("¿Qué estás pensando...?")
twitter.update_status(status=input())
```

4.7.10 PyText: followers_analysis

```
#Listas del Documento
names = []
```

```
usernames = []
ids = []
locations = []
follower_count = []
#Escribe la fecha en formato Y-M-D en el nombre del documento
datestamp = datetime.datetime.now().strftime("%Y-%m-%d")
print ("Obtener la lista de seguidores de @: \n")
username = input()
followers = []
next_cursor = -1
while(next_cursor):
     get_followers = twitter.get_followers_list(screen_name=username,
count=250, cursor=next_cursor)
     for follower in get_followers["users"]:
          #Agregando detalles a la lista de usuarios
          names.append(follower["name"].encode("utf-8"))
          usernames.append(follower["screen_name"].encode("utf-8"))
          ids.append(follower["id"])
          locations.append(follower["location"].encode("utf-8"))
          follower_count.append(follower["followers_count"])
          next_cursor = get_followers["next_cursor"]
#Definiendo el tipo de archivo .csv comma-separated values
open_csv = open(username+"-"+datestamp+".csv","wb")
#Escritura al archivo
followers_csv = csv.writer(open_csv)
#Creando título de fila
names.insert(0, "@%s tiene %s seguidores (%s)" % (str(username),
str(len(followers)), str(datestamp)))
usernames.insert(0, "")
ids.insert(0, "")
locations.insert(0, "")
follower_count.insert(0, "")
#Nombrando cada columna con su propio título
names.insert(1, "\nNombre\n")
usernames.insert(1, "\nUsername (@)\n")
ids.insert(1, "\nUser ID\n")
locations.insert(1, "\nUbicación\n")
follower_count.insert(1, "\n# de sus Seguidores\n")
#Haciendo un merge de nuestras listas
rows = zip(names, usernames, ids, locations, follower_count)
#Escribiendo las filas una por una en el documento
for row in rows:
    followers_csv.writerow(row)
#Guardar y cerrar el documento
open_csv.close()
```

4.7.11 PyText: naive_bayes

```
def bagOfWords(tweets) :
     wordsList = []
     for (words, sentiment) in tweets:
          wordsList.extend(words)
     return wordsList
def wordFeatures(wordList) :
     wordList = nltk.FreqDist(wordList)
     wordFeatures = wordList.keys()
     return wordFeatures
def getFeatures(doc) :
     docWords = set(doc)
     feat = {}
     for word in wordFeatures:
          feat['contains(%s)' % word] = (word in docWords)
     return feat
#Utilizando 2 conjuntos de tweets, positivos y negativos
positiveTweets = [('@stellargirl I loooooooovvvvvveee my Kindle2. Not
that the DX is cool, but the 2 is fantastic in its own right.',
'positive'),
negativeTweets = [('?Obama Administration Must Stop Bonuses to AIG Ponzi
Schemers ... http://bit.ly/2CUIg', 'negative'),
('started to think that Citi is in really deep s&amp;^t. Are they gonna
survive the turmoil or are they gonna be the next AIG?', 'negative')]
#Creando el corpus con el conjunto de ambos tipos de tweets exluyendo
palabras de 3 caracteres o menos
corpusOfTweets = []
for (words, sentiment) in positiveTweets + negativeTweets:
     wordsFiltered = [e.lower() for e in nltk.word_tokenize(words) if
len(e) >= 3]
     corpusOfTweets.append((wordsFiltered, sentiment))
#Obteniendo todos los features de las palabras
wordFeatures = wordFeatures(bagOfWords(corpusOfTweets))
#Obteniendo el training set
training = nltk.classify.apply_features(getFeatures, corpusOfTweets)
#Entrenando el algoritmo Naive Bayes
classifier = nltk.NaiveBayesClassifier.train(training)
#Imprimiendo las palabras más frecuentes o informativas utilizadas por
este clasificador
print (classifier.show_most_informative_features(30))
```

4.7.12 PyText: sentiment_analysis

```
#Realizando la búsqueda
print ("Escribe el término a buscar: ")
result = twitter.search( q = (input()) )

#Iterando en la lista de resultados e imprimir cada sentimiento por tweet
for status in result["statuses"]:
     print("Tweet: {0} \n Sentimiento: {1}"
     .format(status["text"],
     classifier.classify(extract_features
     (status["text"].split())))
```

Conclusiones del Capítulo

A este momento, todas las expectativas que estaban creadas para escribir la librería se cumplieron. La complejidad de abstracción pudo disminuirse gracias a la existencia del modelo de datos definido. Las pruebas para verificar la ejecución se evaluaron en el siguiente capítulo en dónde también se mostrarán representaciones de datos que se pueden obtener de diversas herramientas.

El marco teórico fundamentó eficientemente la algoritmia y modelos matemáticos necesarios para la escritura de la librería, el naive_bayes y sentiment_analysis que definen toda esta librería están formalmente descritos.

Fueron suficientes horas de trabajo invertidas en la construcción de esta librería, no solo se escribió código, hubo una elección de fórmulas, diagramas y modelos a seguir para definir este conjunto de programas.

Como se observó, la API de twitter es muy diversa y permite hasta ahora obtener mucha información, para los escenarios que se muestran en el último capítulo, estas herramientas de software en conjunto con la librería se complementan para entregar un resultado diverso, muy bien informado y sobre todo que reporte cada escenario de acuerdo a su objetivo.

Capítulo 5 Desarrollo de PyText y el Text Reporting

En esta etapa del proyecto existen diversas formas de representar la información obtenida, es necesario diferenciar el usuario o el conjunto de usuarios que deseen aplicar un método de minería de textos para ajustar los parámetros y reportes a sus necesidades. No todos los usuarios buscan las mismas respuestas aún partiendo con la misma información, este es el caso más trivial respecto a la ambigüedad en la web y será mostrado fácilmente.

En resumen, todo lo antes mencionado en los capítulos anteriores hasta este punto tendrá sentido y razón de ser. La librería se ejecutará, algunas capturas de pantalla con la información más relevante dado a que hay suficientes combinaciones con los diversos problemas aplicables.

5.1 PyText: twitter_bot

Existe la necesidad de automatizar el flujo de creación de la información, en términos estrictos; la creación de contenido en una red social. Los términos, frases e interacciones que un usuario sufra en una red social es responsabilidad de lo que opina, comenta y hace parte de su forma de expresión.

Muchas de las ideas de la automatización de posts en redes sociales, se ve afectada por técnicas de marketing, branding, que por razones monetarias y/o de posicionamiento son necesarias y totalmente aplicables.

Parte importante de PyText como librería, era tener un bot que posteara algunas frases cada n-segundos es decir cierto tiempo establecido por el usuario. Esto además de poder enriquecer un campo de información puede tener usos diversos, desde experimentos sociales, hasta campañas políticas; por mencionar algunos.

```
 # -*- coding: utf-8 -*-
##   PyText Twitter_bot
##  Created by Martín G. Morgado for MOVIS lab.
##  Copyright © 2016 geekblitz. All rights reserved.
#Bot que twitteará aleatoriamente las frases que estén en una lista

from twython import Twython, TwythonError
import time
import random
```

```
import sys
ConsumerKey = "XXXXJ9zzaDcEICz9BxDxXXXX"
ConsumerSecret = "XXXXrbJkIiXCwcZVWl0Ig5Ed1EeJqgQHyR0g2Wf76tPxKuXXXX"
AccessToken = "XXXX316-TTJiiR4lsWdGf7SFCRRu1nwEJxVP92X2Gzh1QXXXX"
AccessTokenSecret = "XXXXrgb21TfuoBSjT4g9N5zYUd3XZ8jR1k0KeSrwwXXXX"
#Instanciando el objeto Twython dando acceso a la cadena token como
parámetro
twitter = Twython(ConsumerKey,
     ConsumerSecret,
     AccessToken,
     AccessTokenSecret)
#Lista para mandar tweets
list = [
     "En breve me pongo a investigar quién va ganando la elección",
    "#ImVotingBecause",
     "Hillary Clinton",
    "Daryl!",
    "Si Trump gana x_x",
    "My today...!!!",
    "1 minute in...",
    "I'm done, git commit",
     "OK wait...",
     ">.<",
     "LOL",
     "Because we need... each other",
     "#SAP basis ?",
     "Random 2-1 2-2",
     "Split it out!!!",
     "New #nintendo",
     "Where's my #Kirby ?",
     "C O F F E E.",
     "...",
     "#BlackMirror time!",
     "¬¬",
     "Mejor ni dormía",
     "Ya ni juego Heroes of the Storm",
     "Crash dump!",
     "Almost there...",
     "No he visto #TWD D:"
    ]

#While para escribir
while True:
    try:
        if len(list) > 0:
```

```
            toTweet = list[random.randint(0,len(list))-1]
            twitter.update_status(status=toTweet)
            list.remove(toTweet)
            time.sleep(900) #Tiempo en s, cada 15 m
        else:
#Se terminan nuestros tweets
            twitter.update_status(status="I'm kind of empty =(")
            break
    except TwythonError as e:
        print (e)
```

Los resultados de la ejecución se observan inmediatamente. No importa la cantidad de frases depositadas, en este caso una lista está pre-compuesta de un número de frases que aleatoriamente van a ser twitteadas. Existen variantes, la primera es obtener información de un archivo y enlistarlas para inmediatamente twittearlas. En este ejemplo sencillo solo se pretende mostrar la necesaria automatización de tweets para su análisis.

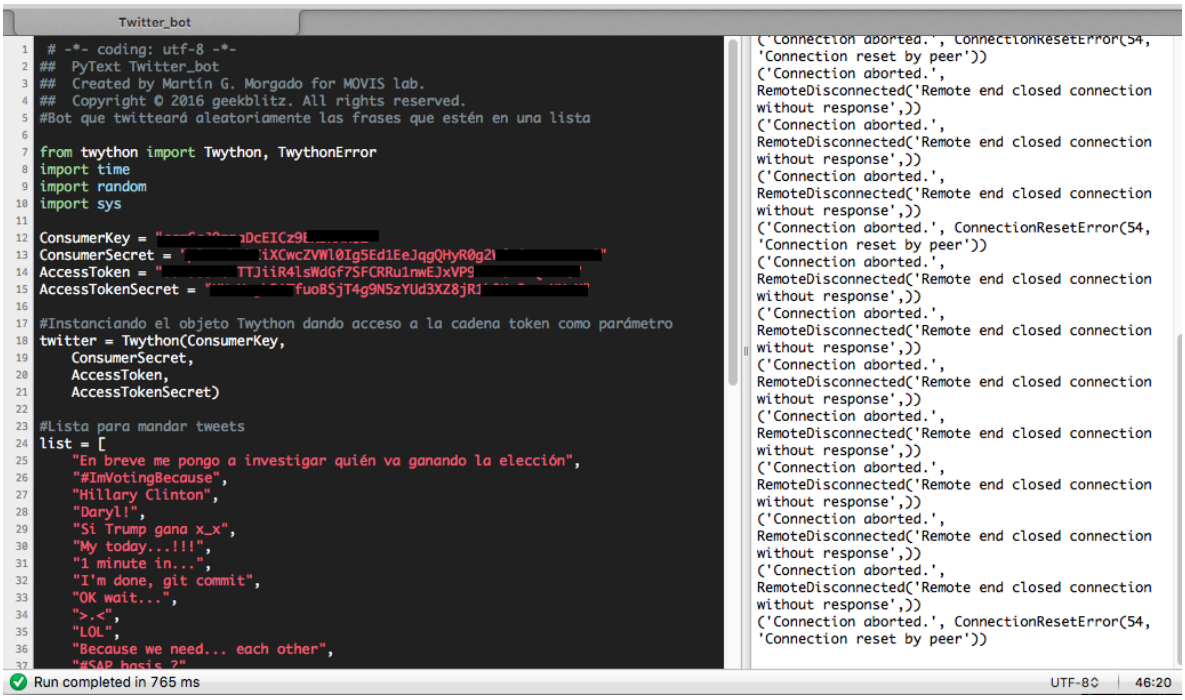


Figura 5.1 Ejecución del Bot.

La ejecución del programa muestra en la consola una línea cada vez que escribe algo en twitter, esto es porque solicita y envía una conexión. Un token es enviado

por cada tweet. ('Connection aborted.', ConnectionResetError(54, 'Connection reset by peer')) en realidad, lo único que hace es conectar y desconectar para que el peer no quede esperando esos 900ms todo el tiempo.

En la figura 5.2 se pueden observar los tweets escritos por el bot, en un tiempo de 15 min de diferencia entre cada uno. Los detalles son necesarios para saber cual es el cliente que utilizó twitter para comunicarse, en este caso fue PyText quién envió el tweet, debido a que tiene todas las autorizaciones del OAuth a través de la API con los tokens de seguridad necesarios.

Figura 5.2 Tweets publicados desde el bot en forma aleatoria.

En la figura 5.3 se puede observar el origen del tweet, esto es importante para el text reporting dado a que es muy diversa la forma en la que interactúan los

usuarios dependiendo la plataforma en la que se encuentran, por ejemplo: no se ve la misma cantidad de información en un portal web que en una aplicación móvil.

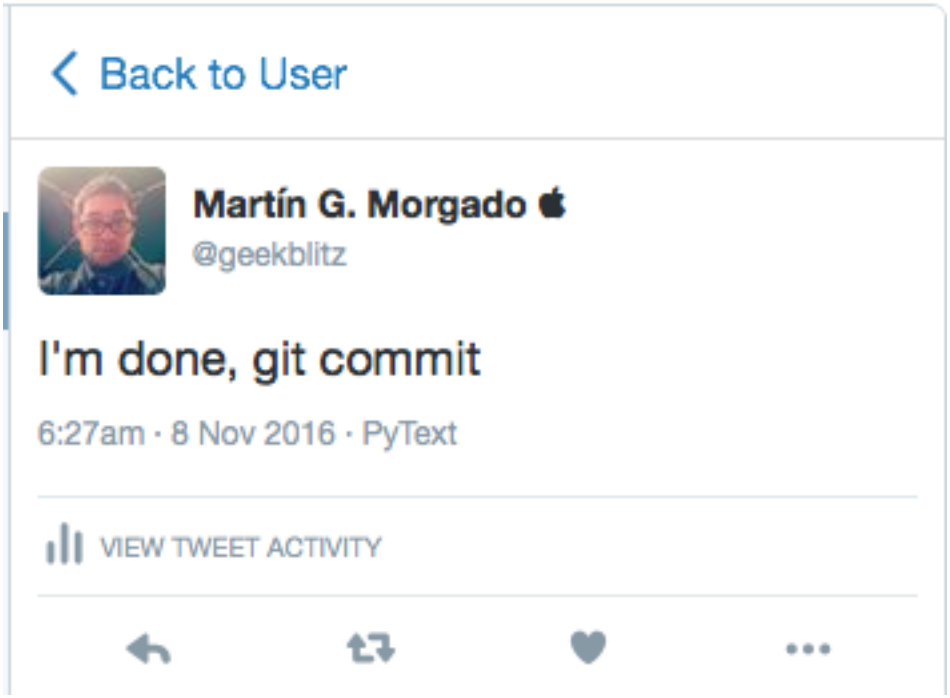


Figura 5.3 Ejecución del Bot.

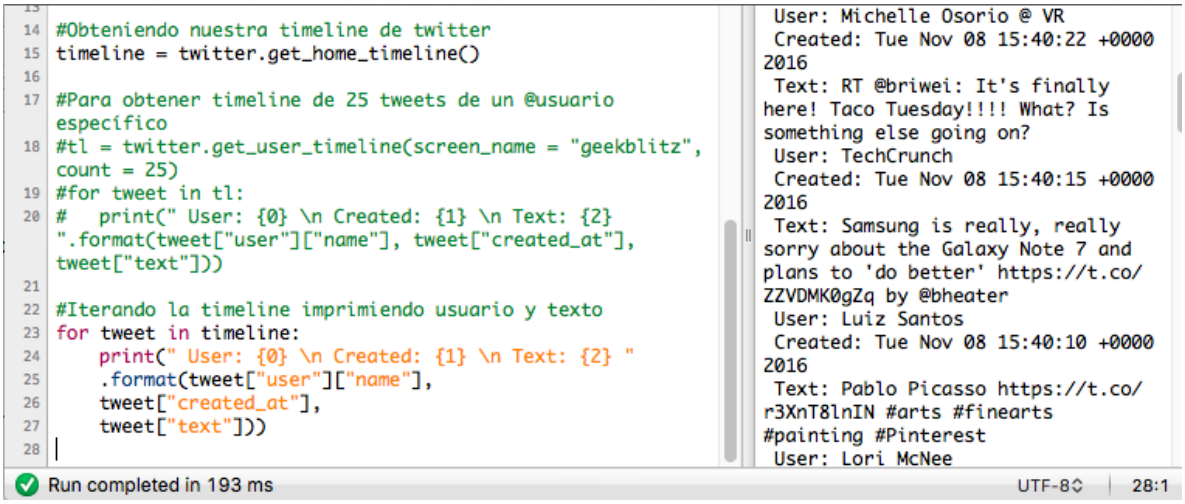
5.2 PyText: Tests

A continuación se muestran algunas de los resultados del uso de la librería en texto plano. Algunos programas importan/exportan archivos para el uso de información diversa.

No todos los ejemplos pueden ejecutarse en segundos, algunos requieren suficiente tiempo de ejecución para pre-procesar y escribir a archivos de diversos tipos. Los ejemplos más relevantes fueron descritos en código al final del capítulo 4 en donde se implementan en python.

Una característica fundamental de estos programas es su longitud, si se hubiera utilizado un lenguaje de no scripting comoC o C++, posiblemente algunos de los programas realizarían demasiadas llamadas a funciones o a encabezados de sistema para funcionar, esta es una de las ventajas más grander de hacerlo en python.

En las figuras 5.4 y 5.5 se pueden observar los tweets en texto, es necesario



mencionar que si el contenido de un tweet es url, audio/video o una fotografía; el programa permite describir el tipo de información proporcionada.

Figura 5.4 Streaming de tweets en texto plano.

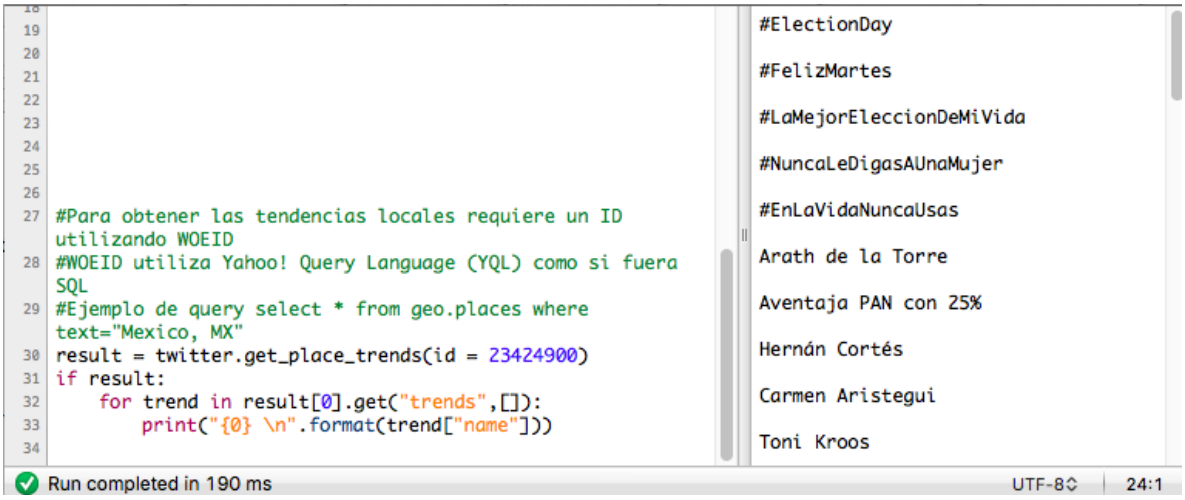


Figura 5.5 Lista de los trending topics en texto plano.

5.3 Text Reporting

La información en calidad de texto no puede ser útil del todo, es necesario que obtenga o sea verificada por un proceso de transformación. Así no solo hay datos que contienen números, letras, fechas y/o cantidades sino que esto puede informar de forma considerable.

A continuación se presentan algunas propuestas de información reportada, el origen fue texto. Los servicios utilizados fueron Twitter Analytics y Wolfram Alpha.

5.3.1 Word-Topic frequency

El objetivo de este reporte es mostrar diversas palabras que definen la idea principal, en este caso la relación entre la frecuencia y su tamaño es la mostrada. Este ejemplo es muy utilizado al momento de definir qué es algo.

Figura 5.6 Una lista de word-frequency de diversos temas.

Las frecuencias en las palabras pueden ser útiles como en el caso de la implementación de bag_of_words en la librería, pero cuando el texto a analizar es demasiado grande, eliminar la entropía de este hace que no sea una buena forma de analizarlo.

5.3.2 Facebook Analysis

Hasta el momento esta utilidad de Wolfram es gratuita, se puede encontrar en el apartado de “Personal Analytics” en dónde el usuario proporciona permiso a que accedan a su información y la herramienta hace cálculos y muestra algunos reportes de información relevante.

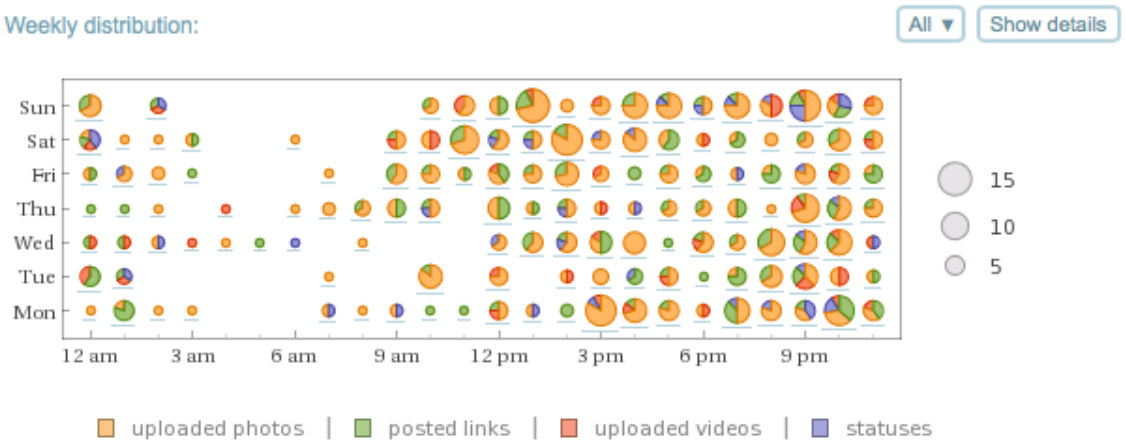


Figura 5.7 Frecuencia de los usos de Facebook en un perfil.

5.3.3 Twitter Analytics

Twitter proporciona reportes de información de los usuarios respecto a sus seguidores, en principio está herramienta era solo para desarrolladores; ahora está disponible al público.

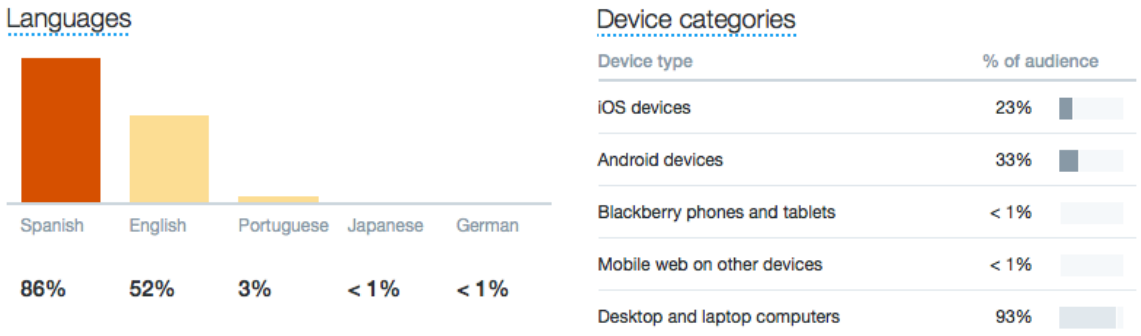


Figura 5.8 Información del usuario en twitter respecto a sus seguidores.

5.4 Resultados PyText

PyText no es capaz de generar un reporte de información, debido a que únicamente analiza información, realiza un pre-procesamiento y procesamiento de información a tiempo real. Algunas operaciones son muy costosas como la ejecución del análisis de sentimientos para generar el corpus. O la obtención de la lista de seguidores de una persona.

En general la librería funciona como se esperaba, su implementación se hizo de la forma más simple argumentando cada algoritmo que la forma, naive-bayes que desde el punto de vista teórico podría no funcionar en todos los casos al utilizar muchas operaciones y funciones. Pero en general solo requiere una herramienta de reporte para mejorar sus resultados.

PyText fue desarrollada en python 3.5, cuenta con 10 programas independientes con un total de 189 líneas de código.

Conclusiones del Capítulo

En este punto se observó que es muy necesaria la implementación de una utilidad de reporting, en el mercado hay suficientes pero son altamente costosas; por el tiempo y la disponibilidad que presentan y también por los recursos de cómputo que requieren.

El desarrollo de esta librería ha permitido la exploración de campos de estudio más específicos, se observo que el rendimiento del algoritmo varía por una poca cantidad de información y lo puede hacer inestable.

Las pruebas que se hicieron para validar la calidad de la librería solo fueron propuestas al momento de su desarrollo ya que para una aplicación específica requiere un ajuste en el código. Esto es por las diferentes APIs que puede integrar.

Conclusiones

La realización de este documento es la máxima referencia para cubrir de forma conceptual todos los elementos necesarios para comprender la minería de textos y poder desarrollar una implementación, en primera instancia se hace notar la importancia de la información y su potencial uso. Por otro lado el tratamiento de esta información puede ser llevado a cabo de muchas formas y técnicas, las que aquí se describieron fueron estudiadas previamente para solo atacar al problema principal: La minería de textos.

Con estas conclusiones, a lo largo de estos capítulos que poco a poco profundizaron esta idea de desarrollo e investigación, se comprueba la utilidad de los conceptos científicos y toda la base tecnológica necesaria para llevar a cabo un proyecto como este. No sólo requería bases matemáticas, sino de ramas de la inteligencia artificial y casos particulares del análisis de datos. A este nivel ahora es posible visualizar otros enfoques que antes no estaban ni siquiera contemplados, este proyecto ha mejorado la perspectiva personal del estudio de la información.

En general esto es lo que se ha aportado con este proyecto:

- Considerar la alta importancia de la información en diversos formatos.
- Definir las situaciones en las que es necesario realizar un análisis de datos mediante scripting.
- Comprobar que las técnicas de análisis de datos sirven a tiempo real sin altos recursos de cómputo.
- Comprender que las optimizaciones no siempre se pueden aplicar a los sistemas.
- Aportar de forma directa a la creación de código para enfrentar diversos problemas, en este caso la parte de visualización es un problema que se pudo observar y desarrollar.

Hay muchas cosas por mejorar, pero esto es un excelente comienzo, los objetivos se cumplieron considerando las peores situaciones para ejecutar la librería. El siguiente paso es proporcionar la formalidad necesaria para que además de un análisis online también reporte de forma semántica y pueda ser útil para la toma de decisiones inclusive involucrar este código con sistemas expertos.

Bibliografía

[1] A. Hotho, A. Nürnberger and G. Paab. “A brief survey of text mining”. In LDV Forum GLDV Journal for Computational Linguistics and Language Technology, 20(1):19-62, May 2005.

[2] V. Gupta and G. S. Lehal. “A survey of text mining techniques and applications”. In Journal of Emerging Technologies in Web Intelligence, 1(1): 60-76, August 2009.

[3] S. M. Weiss, N. Indurkhya and T. Zhang. “Fundamentals of predictive text Mining”. Springer, London, UK, 2010.

[4] M. Bramer. “Principles of data mining” (undergraduate topics in computer science). Springer, London, UK, 2007.

[5] G. ADAM, C. BOURAS and V. POULOPOULOS. Efficient extraction of news articles based on RSS crawling. In International Conference on Machine and Web Intelligence (ICMWI), pages 1-7, October 2010.

[6] U. M. FAYYAD, G. PIATETSKY-SHAPIRO and P. SMYTH. From data mining to knowledge discovery in databases. In AI Magazine, 17(3): 3754, 1996.

[7] M. W. BERRY and J. KOGAN. Text mining applications and theory. John Wiley & Sons, United Kingdom, 2010.

[8] P. CHAPMAN, J. CLINTON, R. KERBER, T. KHABAZA, T. REINARTZ, C. SHEARER and R. WIRTH. CRISP-DM 1.0 Step-by-step data mining guide. Technical Report, CRISP-DM Consortium, August 2000.

[9] A. AZEVEDO and M. F. SANTOS. KDD, SEMMA and CRISP-DM: a parallel overview. In Proceedings of the IADIS European Conference on Data Mining, pages 182-185, 2008.

[10] B. BERENDT. Text mining for news and blogs analysis. In C. Sammut & G. Webb (Eds.), Encyclopedia of Machine Learning, pages 968-972. Springer, 2011.

[11] C. C. AGGARWAL and C. ZHAI. Mining text data. Springer, 2012.

[12] G. SALTON, A. WONG and C. S. YANG. A vector space model for automatic indexing. In Communications of the ACM, 18(11): 613-620, Nov. 1975.

[13] A. HOTHO, A. NÜRNBERGER and G. PAAß. A brief survey of text mining. In LDV Forum GLDV Journal for Computational Linguistics and Language Technology, 20(1):19-62, May 2005.

[14] A. BERGO. Text categorization and prototypes. Technical report, Institute for Logic, Language and Computation, Universiteit van Amsterdam, 2001.

[15] Ludmila I. Kuncheva. Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience, 2004. ISBN 0471210781.

[16] Michael John Collins. A new statistical parser based on bigram lexical dependencies. In Proceedings of the 34th Annual Meeting on Association for Computational Linguistics, ACL ’96, pages 184–191, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics. doi: 10.3115/981863.981888. URL http: //dx.doi.org/10.3115/981863.981888.

[17] Peter D. Turney. Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, ACL ’02, pages 417–424, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083. 1073153. URL http://dx.doi.org/10.3115/1073083.1073153.

[18] F. Benamara, C. Cesarano, A. Picariello, D. Reforgiato, and V.S. Subrahmanian. Sentiment analysis: Adjectives and adverbs are better than adjectives alone. In Proceedings of the International Conference on Weblogs and Social Media (ICWSM), 2007.

[19] Ellen Rilo, Janyce Wiebe, and Theresa Wilson. Learning subjective nouns using extraction pattern bootstrapping. In Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4, CONLL ’03, pages 25–32, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1119176.1119180. URL http://dx.doi.org/10.3115/1119176.1119180.

[20] Peter R. R. White. Appraisal outline. Stroudsburg, PA, USA, 2001. Association for Computational Linguistics. URL www.grammatics.com/appraisal.

[21] V. M. Morales de Jesús. Utilización de expresiones de actitud para el análisis de sentimientos. Puebla, Puebla., 2014.

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

[23] Bing Liu. Sentiment analysis and opinion mining. Synthesis Lectures on Human Language Technologies, 5(1):1–167, 2012. doi: 10.2200/

S00416ED1V01Y201204HLT016. URL http://dx.doi.org/10.2200/S00416ED1V01Y201204HLT016.

[24] Kai Ming Ting and Ian H. Witten. Stacked generalization: when does it work? In in Procs. International Joint Conference on Artificial Intelligence, pages 866–871. Morgan Kaufmann, 1997.

[25] Kagan Tumer and Joydeep Ghosh. Linear and order statistics combiners for pattern classification. CoRR, cs.NE/9905012, 1999. URL http://arxiv.org/abs/cs.NE/9905012.

[26] Richard O. Duda and Peter E. Hart. Pattern Classification and Scene Analysis. A Wiley Interscience Publication. Wiley, 1973.

[27] J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas. On combining classifiers. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 20(3):226–239, Mar 1998. ISSN 0162-8828. doi: 10.1109/34.667881.

[28] Thomas G. Dietterich. Ensemble methods in machine learning. In Proceedings of the First International Workshop on Multiple Classifier Systems, MCS ’00, pages 1–15, London, UK, UK, 2000. Springer-Verlag. ISBN 3-540-67704-6. URL http://dl.acm.org/citation.cfm?id=648054.743935.

[29] João Gama and Pavel Brazdil. Cascade generalization. Machine Learning, 41 (3):315–343, 2000. ISSN 0885-6125. doi: 10.1023/A:1007652114878. URL http: // dx.doi.org/10.1023/A%3A1007652114878.

[30] Uci. machine learning repository. urlhttp://archive.ics.uci.edu/ml/

[31] V. Anitha and R.Leela Velusamy. Iris recognition systems with reduced storage and high accuracy using majority voting and haar transform. Advances in Intelligent Systems and Computing, pages 813–822. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-30110-0. doi: 10.1007/978-3-642-30111-7 78. URL http://dx.doi.org/10.1007/978-3-642-30111-7_78.

[32] Maurizio Aiello, Maurizio Mongelli, and Gianluca Papaleo. Supervised learning approaches with majority voting for dns tunneling detection. In International Joint Conference SOCO’14-CISIS’14-ICEUTE’14, volume 299 of Advances in Intelligent Systems and Computing, pages 463–472. Springer International Publishing, 2014. ISBN 978-3-319-07994-3. doi: 10.1007/978-3-319-07995-0 46. URL http://dx.doi.org/10.1007/978-3-319-07995-0_46.

[33] J. Ross Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. ISBN 1-55860-238-0.

[34] Johannes Fürnkranz. More e cient windowing. Technical Report OEFAI-TR-97-01, Austrian Research Institute for Artificial Intelligence, Wien, Austria, 1997. URL http://www.ke.informatik.tu-darmstadt.de/~juffi/ publications/aaai-97.ps.gz.

[35] S. Milgram, “The small world problem.” http://measure.igpp.ucla.edu/ GK12-SEE-LA/Lesson_Files_09/Tina_Wey/TW_social_networks_Milgram_1967_small_world_problem.pdf, 1967.

[36] GO-Globe,“SocialMediaInChina–Statistics and Trends[Infographic].”http:// www. go-globe.com/blog/social-media-china/, 2013.

[37] Karsten Strauss, “Facebook And The China Problem.” http://www.forbes.com/sites/karstenstrauss/2012/05/18/facebook-and-the-china-problem/, 2012.

[38] Cindy Chiu, Chris Ip, and Ari Silverman, “Understanding social media in China.” http://www.mckinsey.com/insights/marketing_sales/understanding_social_media_in_china, 2012.

[39] L. D’Monte, “Swine Flu’s Tweet Causes Online Flutter.” http://www.business-standard.com/article/technology/swine-flu-s-tweet-tweet-causes-online-flutter-109042900097_1.html, 2009.

[40] D. Sarno, “Twitter creator Jack Dorsey illuminates the site’s founding document. Part I.” http://latimesblogs.latimes.com/technology/2009/02/twitter-creator. html, febrero 2009.

[41] M. Arrington, “Odeo Releases Twttr.” http://techcrunch.com/2006/07/15/ is-twttr-interesting/, julio 2006.

[42] M. L. Congosto, “Definiendo Twitter en pocas palabras.” http://www.barriblog.com/index.php/2009/09/30/definiendo-twitter-en-pocas-palabras/, septiembre 2009.

[43] Twitter Inc., “Changes coming in Version 1.1 of the Twitter API.” https://dev.twitter.com/blog/changes-coming-to-twitter-api, agosto 2012.