Universidad de Valladolid

E.T.S. Ingeniería Informática

TRABAJO FIN DE GRADO

ANÁLISIS DE SENTIMIENTO EN TWITTER CON HERRAMIENTAS DE BIG DATA

Máster en Ingeniería Informática – Especialidad Big Data

Autor:
D. Mario Matesanz Niño
Tutor:
Dr. Carlos J. Alonso González

Resumen

Este TFM tiene objetivo realizar una propuesta de arquitectura Big Data con la que se pueda desarrollar un análisis de sentimiento orientado a textos extraídos de la plataforma Twitter mediante diversas técnicas de clasificación. Se expondrán estas técnicas, su fundamento y su funcionamiento y se pondrán en uso. Para esto se creará una herramienta que permita el uso y configuración de estas técnicas y se presentarán los resultados experimentales que muestran la viabilidad de la propuesta.

Abstract

The aim of this TFM is to make a proposal of Big Data architecture in order to develop a sentiment analysis can be developed focused to texts extracted from the Twitter platform through various classification techniques. These techniques, their foundation and their operation will be exposed and will be implemented. For this purpose, a tool which allows the use and configuration of these techniques will be created and the experiment results which show the viability of the proposal will be finally presented.

Índice de Acrónimos

AS: Análisis de sentimiento
BOW: *Bag of Words* (Bolsa de palabras).
HO: *Hold Out*
IDF: Frecuencia de documento inversa
IoT: *Internet of things*
K-NN: *K-Nearest Neightbors*
NB: Naïve Bayes
MLP: *Multi-layer perceptron*
SVM: Support vector machine
TF: Frecuencia de términos
VC: Validación cruzada

Índice de Figuras

Índice de Tablas

Tabla de contenido

1 Introducción

1.1 Introducción al análisis de sentimientos: interés y auge con Big Data

El análisis de sentimiento es una disciplina dentro de la minería de datos asociada a la minería de textos. Esta disciplina ha tenido un gran crecimiento en la última década con la aparición de las redes sociales y el auge de estas. Debido a estos medios sociales se ha producido una proliferación de críticas, calificaciones, recomendaciones y otras formas de opinión. Esta información es utilizada por empresas que buscan vender sus productos identificando nuevas oportunidades, por ello es importante filtrar toda esta gran cantidad de datos identificando el contenido relevante. Gracias a la aplicación de técnicas y herramientas de Big Data se ha permitido el procesamiento de la gigantesca cantidad de datos que producen miles de millones de personas. Estas están conectadas a la red generando datos, a la que se pueden conectar desde prácticamente cualquier sitio con la introducción de las tecnologías móviles.

1 Usuarios de redes sociales en el mundo

1.2 Contexto y Objetivos

El objetivo de este TFM es presentar una arquitectura Big Data para realizar minería de datos, centrando el estudio en la disciplina de la minería de textos, mediante la aplicación de la misma para el análisis de sentimiento.

El fin del desarrollo del proyecto será el de construir un sistema capaz de realizar una clasificación de ciertos textos extraídos de la plataforma *Twitter*. Los textos podrán ser clasificados entre grado de sentimiento positivo, negativo o neutros. Esta clasificación será realizada principalmente con herramientas de Big Data y aplicando, entre otras, técnicas de aprendizaje automático.

Una vez que hemos definido el ámbito de trabajo del proyecto se van a establecer los objetivos del mismo que serán:

- Presentar una arquitectura de trabajo para un sistema de Big Data.
- Realizar una extracción de datos, en este caso semi-estructurados de la plataforma *Twitter* con los que trabajar.
- Almacenamiento de los distintos estados por los que pasarán los datos extraídos.
- Realizar un análisis y preparación de los datos.
- Realizar una clasificación de los datos mediante técnicas de aprendizaje automático y técnicas de clasificación basada en el léxico.
- Crear un interfaz para que el usuario pueda utilizar los algoritmos implementados además de utilizar los algoritmos de clasificación propuestos.

1.3 Organización de la memoria

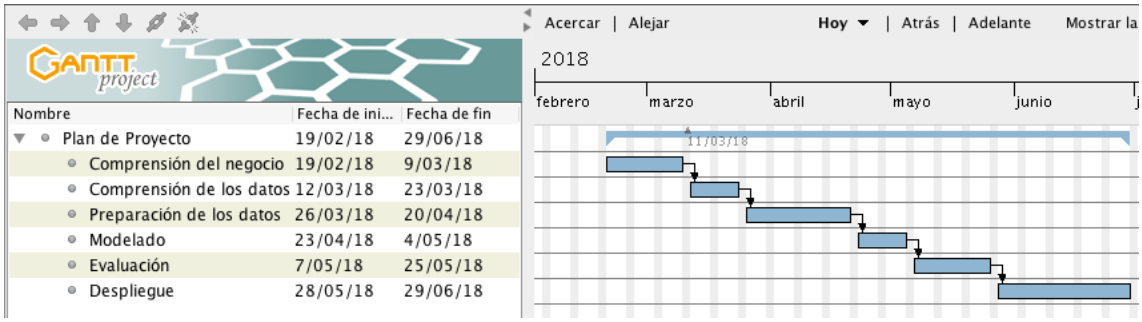
La memoria será dividida en ocho partes fundamentales:

1. Una introducción, con una apuesta en contexto del ámbito y objetivos del proyecto.
2. Un desarrollo de los temas principales, como son la minería de textos y el análisis de sentimientos.
3. Una introducción al Big Data con una propuesta de arquitectura.
4. El desarrollo de la arquitectura propuesta.
5. El desarrollo de un entorno generado para el análisis de sentimiento.
6. Las pruebas desarrolladas sobre la herramienta generada.
7. Las conclusiones sobre el proyecto realizado.
8. Finalmente estará la bibliografía con el conjunto de fuentes consultadas.

1.4 Planificación

1.4.1 Planificación Inicial

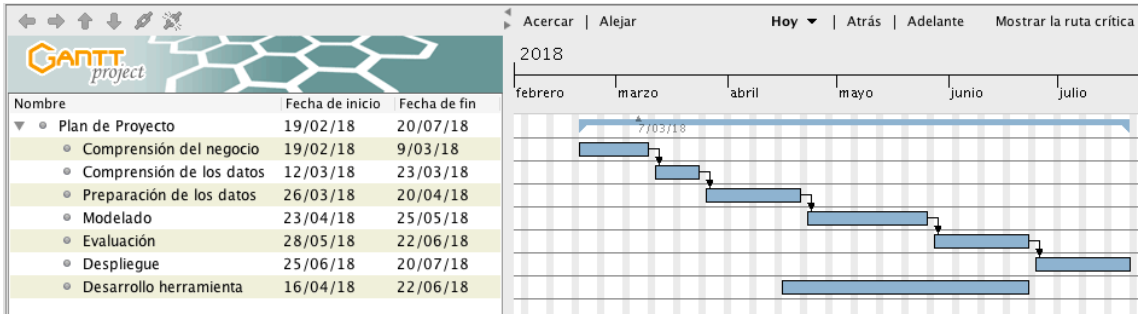
El proyecto se comenzó la semana del 19 de febrero con previsión de entrega de la semana del 20 de Julio. Debido a temas de trabajo el tiempo que se le dedicará será de lunes a viernes en torno a tres o cuatro horas por la tarde. Esto da lugar a una media de 18 horas semanales de trabajo. Intentando perder el menor número de días posibles por temas externos al proyecto, daría lugar a unos 95 días de trabajo. Considerando por festivos u otros temas que se pueden trabajar el 90% de los días el periodo tendría en torno a las 300 horas.



2 Planificación inicial

1.4.2 Planificación Final

La generación de la interfaz para la herramienta de clasificación dio lugar a un incremento de los plazos iniciales, al tener que realizar esta tarea de forma paralela a la preparación de datos, el modelado y la evaluación. Esta tarea, en un principio no prevista, complicaron la planificación inicial y alargaron el plazo hasta la fecha límite de entrega.



3 Planificación final

Finalmente se alargó la duración a 110 días, que, respetando las horas de trabajo iniciales, dieron lugar a unas 350 horas de trabajo.

2 Minería de Textos y Análisis de Sentimientos

2.1 Minería de textos

La minería de textos se refiere al proceso de extracción, de manera automática o semiautomática, de nueva información a partir de textos. Este proceso está englobado dentro de la minería de datos, y se puede definir como el proceso de descubrimiento de patrones en grandes volúmenes de datos. Para este proceso se utilizan métodos de inteligencia artificial, aprendizaje automático, estadística y sistemas de bases de datos. Definida minería de datos, la minería de textos, en una primera aproximación se puede definir como el uso de técnicas de minería de datos para la búsqueda de conocimiento en grandes colecciones de textos.

2.1.1 Evolución histórica

Inicialmente y hasta hace poco, las bases de datos solo trataban datos estructurados, relaciones y tablas. Estas podían responder preguntas sobre la información que se almacenaba en su estructura. Los métodos para el análisis de información no respondían a la necesidad actual de eficiencia y rapidez siendo en su mayoría manuales. Ante el gran aumento de la cantidad de información disponible para procesar, nace la necesidad de automatizar el proceso de análisis de la misma para obtener conocimiento. Este sería el comienzo de la Minería de Datos, nos ofrece la posibilidad de encontrar patrones y tendencias escondidas en bases y conjuntos de datos trabajando con grandes cantidades de información.

Posteriormente, ya habiendo reconocido la labor de la Minería de Datos, se aprecia que dentro de las empresas no solo hay información en las bases de datos, sino que también existen grandes cantidades de documentos que convenientemente estudiados pueden aportar información extra. A esto es a lo que llamamos Minería de Textos.

A lo largo de la historia la Minería de Textos ha sido llamada de diversas formas: Minería de la Información (*Information Mining*), Arqueología de la Información (*Information Arqueology*), Gestión del Conocimiento (*Knowledge Management*), Minería de Datos (*Data Mining*), Destilación de Conocimiento (*Knowledge Distillation*), Minería de Documentos (*Document Mining*), Minería de Datos Textuales (Text Data Mining), Extracción de Conocimiento en Texto (*Knowledge Discovery in Text*) o Minería de Documentos Web (*Web Document Mining*). Utilizamos así el término de Minería de Textos como el proceso por el cual extraemos información que no está presente de forma explícita en los documentos de estudio.

La Minería de Textos nace ante la necesidad de obtener conocimiento de la información documental presente en las empresas, es decir, de obtener un valor adicional. Estos textos pueden ser de naturaleza heterogénea y distribuida pudiendo proceder de diversas fuentes. Son almacenados en bases de datos documentales o en servidores de correo electrónico pudiendo estar en diferentes formatos e idiomas, lo que hace necesitar un trato específico adecuado a su condición.

Esta información documental, los textos, tienen una estructura demasiada compleja para ser tratada sin un procesamiento previo. Esta es una de las mayores diferencias con la Minería de Datos, la necesidad de obtener una forma intermedia de los documentos para que pueda ser aplicado sobre ellos un algoritmo.

2.1.2 Fases

La minería de textos comprende tres actividades fundamentales para el proceso de obtención de información:

1. Recuperación de la información: consiste en la selección, mediante el método pertinente, de los textos a estudiar.
2. Extracción de la información: consiste en la obtención de datos contenidos en los textos mediante el procesamiento de lenguaje natural.
3. Minería de los datos: consiste en encontrar relaciones entre los datos extraídos de forma previa de los textos.

A su vez estas actividades para realizar la minería se dividen en 4 etapas fundamentales [Gupta2009]:

- Selección del corpus: El primer paso de la minería de textos es definir el conjunto de documentos (corpus). La selección de estos documentos debe ser aleatoria siendo estos documentos representativos. Además, no se deben permitir duplicados en el corpus.

- Etapa de preprocesamiento: esta es una etapa de transformación en la cual los textos seleccionados se transforman en algún tipo de representación estructurada o semi-estructurada facilitando así su posterior análisis.

Con el corpus ya seleccionado y estructurado, es el momento de reconocer los *tokens* (unidades gramaticales más pequeñas de texto), esto implica representar el texto como una lista de palabras.

- Etapa de descubrimiento: en esta etapa se analizan las relaciones internas con el fin de descubrir en ellas patrones de interés o nueva información.

- Etapa de visualización: en esta se analizan los resultados por los usuarios mediante la observación y exploración.

2.1.3 Aplicaciones

Siendo una variante de la minería de datos, la minería de textos adoptara técnicas de aprendizaje automático para la obtención de información y el reconocimiento de patrones. Las aplicaciones de la Minería de Textos son variadas destinadas principalmente a la extracción, clasificación y análisis de la información. Dicho esto, de forma general, el fin de la Minería de Textos es:

- La extracción de información.
- El análisis de sentimientos o minería de opiniones.
- La clasificación documental.
- La elaboración de resúmenes.
- La extracción de conocimiento.

Para este proyecto, cabe destacar la aplicación de la minería de textos para el análisis de sentimiento y opinión (*Sentiment Analysis y Opinion Mining*), que consiste en el análisis de los comentarios generados por usuarios generalmente en redes sociales. Los dos aspectos que comprende son:

a. Análisis de sentimiento: es un problema de clasificación que consiste en la asociación de un listado de mensajes a una categoría u otra, mediante la búsqueda de palabras o expresiones, pudiendo ser esta positiva y negativa, o incluso neutra. A partir de aquí se pueden desarrollar sistemas de recomendación, de detección de contenido inapropiado…
b. Análisis de opinión: muy similar al análisis de sentimiento, la diferencia radica en que el análisis de opinión extrae y analiza la opinión de la gente sobre una entidad en concreto.

La minería de textos es ampliamente utilizada en multitud de sectores económicos. Aquellos sectores en los cuales la minería de textos es muy frecuentemente utilizada son [Gupta09]:

- Publicidad y medios de comunicación.
- Telecomunicaciones, energía y otros servicios industriales.
- Sectores de la tecnología de la información e internet.
- Bancos, seguros y mercados financieros.
- Instituciones políticas, analistas políticos, administración pública y en documentos legales.
- Empresas farmacéuticas y de investigación y de salud.

2.1.4 Importancia

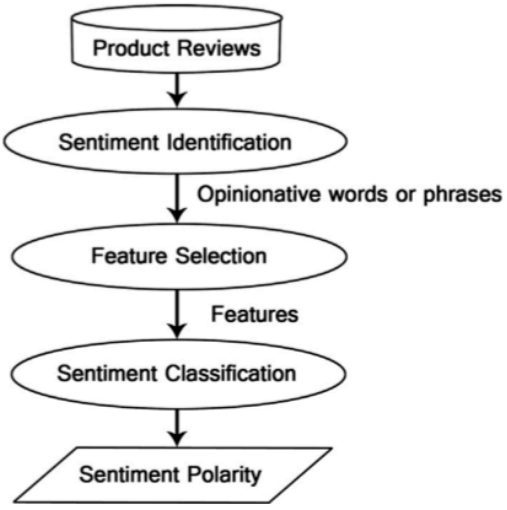
En general, la minería de textos dentro del ámbito de la minería de datos tiene una gran importancia dentro del entorno competitivo empresarial. Actualmente, el análisis de datos dentro del ámbito de la inteligencia de negocios es ampliamente utilizado por las empresas para mantenerse delante de sus competidores en el mercado.

La inteligencia de negocio (BI) es capaz de proporcionar el conocimiento de la información y como debe ser utilizada la misma, para el análisis de la competencia, investigación en el mercado, tendencias económicas, comportamiento del consumo, investigación de la industria, análisis de la información geográfica…. Lo que hace la minería en este ámbito empresarial es la ayuda en la toma de decisiones.

Debido a la aparición de los servicios en la nube, las redes sociales, las posibilidades de interconexión de dispositivos e IoT, la cantidad de datos ha llegado a tener unas dimensiones inmanejables. Los formatos y estructuras no relacionales de estos datos no solo complican el almacenamiento sino, obviamente, el análisis. A esta explosión de datos y su gestión es a lo que llamamos Big Data que lo que ha hecho es ampliar el BI siendo este una nueva fuente de datos para él.

2.2 Análisis de sentimientos

El Análisis de Sentimientos (AS) es un campo en proceso de investigación en el ámbito de la minería de textos. Consiste en el tratamiento de forma computacional de opiniones, sentimientos y texto subjetivo, realizando un estudio de las opiniones de las personas, actitudes y emociones hacia una entidad. Su objetivo final es encontrar las opiniones, identificar los sentimientos expresados en ellas y luego clasificar su polaridad.



4 Proceso de extracción del sentimiento [Medhat2014]

Hay tres niveles principales de clasificación en el AS:

- A nivel de documento: tiene como objetivo clasificar un documento de opinión, según exprese una opinión o sentimiento, en positivo o negativo. Considera el documento completo como una unidad básica de información.
- A nivel de oración: el objetivo es expresar el sentimiento de cada oración. Primero ha de determinar si la oración es subjetiva u objetiva. Si es objetiva determinará si el sentimiento es positivo o negativo.

No existe una gran diferencia entre el AS a nivel de oración y documento ya que los documentos son conjuntos de oraciones. Estas dos clasificaciones no proporcionan los detalles necesarios de los cuales hay que proveer a ciertas aplicaciones por lo que para obtenerlos hay que pasar al tercer nivel.

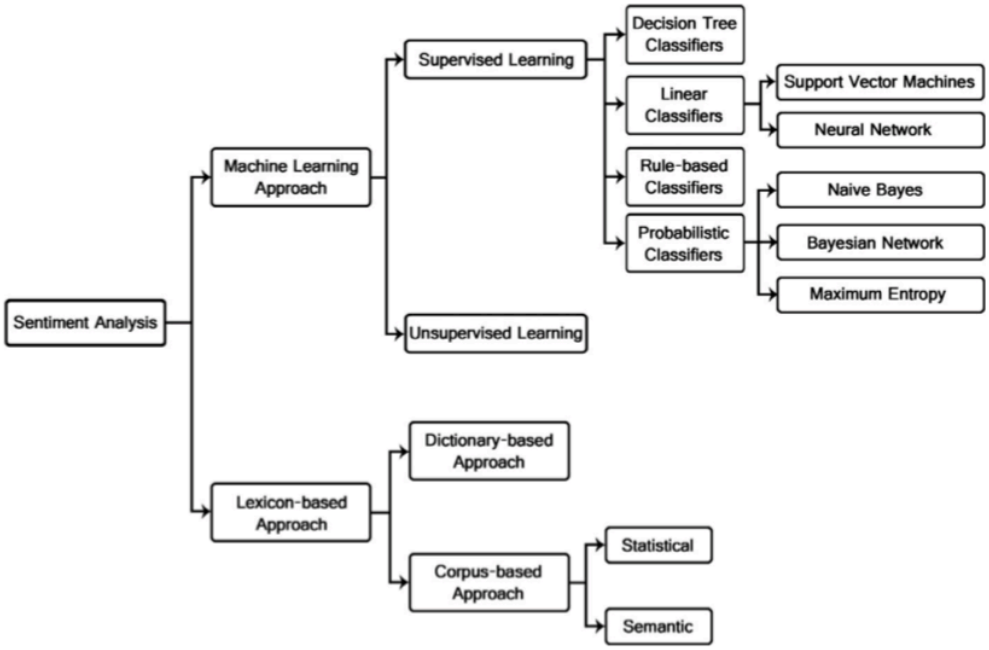
- A nivel de aspecto: tiene como objetivo clasificar el sentimiento con respecto a los aspectos específicos de las entidades. El primer paso es identificar las entidades y sus aspectos. Los proveedores de opinión pueden dar diversas opiniones para diversos aspectos de la misma entidad, como podría ser: “No me gusta la pantalla del iPhone, sin embargo, su batería es muy buena”.

En resumen, el AS en una primera aproximación, consiste en asignar a cada mensaje un valor que esté relacionado con la carga emocional que transmite el mismo. En relación a esa carga emocional, se pueden distinguir 3 tipos de variables:

- Polaridad: indica si el mensaje tiene un sentimiento positivo o negativo. Se puede dar el caso de que aparezca una tercera categoría para la clasificación de mensajes neutros.
- Intensidad: esta proporciona un valor numérico en relación con la intensidad del sentimiento, pudiendo distinguir entre positiva y negativa.
- Emoción: clasifica el texto según los distintos tipos de emociones como serían la alegría, la tristeza y la ira.

La dificultad de realizar un análisis de sentimiento aplicado a grandes volúmenes de datos radica en que el hecho de evaluar el sentimiento la polaridad o el tipo de emoción no resulta siempre unívoco. Incluso cuando el análisis de sentimiento es realizado manualmente puede variar en el resultado según quien sea el codificador.

Las técnicas de clasificación del análisis de sentimiento son:



5 Técnicas para el análisis de sentimiento [Medhat2014]

- Aproximaciones basadas en aprendizaje automático: estas aproximaciones utilizan los algoritmos de aprendizaje automático (Naïve Bayes, Árboles de decisión, regresión logística…) para resolver el problema del análisis de sentimiento como un problema regular de clasificación de textos. Este problema hace uso de las características sintácticas y/o lingüísticas. El problema de la clasificación de textos parte de un conjunto de registros de entrenamiento, a partir de estos registros se genera un modelo que se utiliza para predecir la clase de instancias sin preasignar a una categoría. Dentro de este concepto existen dos aproximaciones:

    - Aprendizaje supervisado: este aprendizaje se realiza cuando existen documentos de entrenamiento ya etiquetados. A partir de esta información se crea el modelo para clasificación.

    - Aprendizaje no supervisado: en este aprendizaje se superan las dificultades que conllevan encontrar un conjunto de entrenamiento correctamente etiquetado. En estos métodos se crea un modelo para clasificación utilizando características extraídas del texto sin etiquetar.

- Aproximaciones basadas en el léxico: en estas aproximaciones se realiza un análisis del texto en los que el método principal es utilizar un conjunto de palabras o frases de opinión para realizar el etiquetado. Se subdividen en dos métodos:

- Aproximación basada en diccionarios: en esta aproximación se utiliza un conjunto de palabras de opinión recogidos de forma manual. Consiste en encontrar estas palabras en el texto para realizar la clasificación mediante las mismas.

- Aproximación basada en un corpus: con este método se pretende esquivar el problema de buscar palabras de opinión en un contexto específico. Estos métodos buscan patrones sintácticos junto con un listado de palabras clave de opinión, para realizar la clasificación tras encontrar coincidencias en un corpus más grande.

2.3 Aproximaciones seleccionadas

Se van a utilizar dos de las aproximaciones mencionadas en el punto anterior que son: de las aproximaciones basadas en el léxico, la basada en diccionarios y de las aproximaciones basadas en aprendizaje automático, el aprendizaje supervisado.

2.3.1 Aproximación basada en diccionarios

Como ya hemos mencionado, esta es una aproximación simple cuyo fin, como el resto de las aproximaciones, es obtener una clasificación de un texto. De manera general, el primer paso es el de obtener un diccionario o conjunto de palabras que expresen una opinión positiva y otro para las que expresen una negativa. Teniendo el diccionario bien definido se realiza una *tokenización* (división del texto en palabras) del texto que puede ir complementada de una *lematización* (extracción de lexemas de las palabras) en el caso de utilizar las raíces de las palabras, en vez de las palabras completas en el diccionario positivo y negativo. Hecho esto se compara la lista obtenida del texto con el diccionario buscando coincidencias. Hay diccionarios en los cuales las palabras llevan asignada un grado de positividad o negatividad que se utiliza para hacer la clasificación. Si no existe este valor simplemente se realiza un recuento de palabras positivas y negativas en cada texto.

2.3.1.1 Etapas

Como ya hemos visto esta aproximación está basada en la búsqueda léxica por lo cual se basa en tres etapas:

1. Lo primero que se hace es obtener los diccionarios del léxico que vamos a buscar en el texto.
2. Después se procesa el texto para transformarlo en el conjunto de léxico que vamos a buscar en los diccionarios, transformando todo aquello que se vea conveniente para mejorar los resultados.
3. Finalmente se buscan las palabras obtenidas en los diccionarios y según los resultados se asigna el texto a una clase.

2.3.2 Aprendizaje supervisado

2.3.2.1 Introducción

El aprendizaje supervisado es una de las formas de realizar aprendizaje automático de las cuales disponemos. La mayoría de las técnicas de aprendizaje automático usa aprendizaje supervisado.

El aprendizaje supervisado se realiza cuando teniendo unas variables de entrada (x) y unas variables de salida (y) usamos un algoritmo para aprender el mapeado de la función de la entrada a la salida.

$$y=f(x)$$

El objetivo es aproximar suficientemente bien la función de mapeado para que, teniendo unas nuevas entradas, podamos predecir las variables de salida para los datos.

Los problemas de aprendizaje supervisado pueden agruparse dentro de regresión y clasificación:

- Clasificación: los problemas de clasificación son aquellos cuyas variables de salida pueden agruparse en categorías como “positivo” o “negativo”.
- Regresión: los problemas de regresión son aquellos cuyas variables de salida corresponden con un valor real, como “dólares” o “peso”.

2.3.2.2 Modelos de texto

Existen diferentes posibilidades a la hora de obtener modelos de texto. El modelo de texto es el conjunto de información que utiliza el algoritmo de clasificación para crear el modelo general con el que realizar predicciones. Algunos de los modelos más utilizados son:

- Bolsa de palabras (*Bag of Words*, BOW): Esta técnica consiste en crear un conjunto con todas las palabras del texto, pudiendo crearse unigramas, bigramas o *n-gramas* [Go2009]. A cada palabra de la bolsa se la asigna un peso que será con el que el algoritmo realice los cálculos.

- Redes de Palabras: una red de palabras es la transformación de un texto, en un grafo G(N,E), donde G es el grafo, o red, compuesto por N palabras (o términos) distintas y E enlaces que las relacionan en un texto [Cárdenas2014]

- Ontologías: Las ontologías es una especificación explicita y leíble por una máquina, de una conceptualización compartida. Se utilizan para modelar términos en un dominio de interés, así como las relaciones entre estos términos. Es la más compleja ya que se necesita un ingeniero en ontologías que genere de forma manual la ontología [Kontopoulos2013].

El método en el que nos centraremos es el de la bolsa de palabras. Se ha decidido por este método por ser capaz de obtener buenos resultados a pesar de su menor complejidad, ya que no tiene en cuenta las relaciones gramaticales entre las palabras del texto, además de por ser el método de entrada de los métodos pertenecientes a las bibliotecas que utilizaremos para clasificación.

2.3.2.3 Etapas

En el caso que nos vamos a centrar, es en el de la clasificación. Para obtener un buen clasificador se realiza un proceso dividido en:

1. Preprocesado de texto: Lo primero que se debe de hacer es procesar el texto, eliminando o transformando todo aquello que se vea conveniente para mejorar los resultados.
2. Selección de características: Con el texto procesado el siguiente paso es realizar una transformación del texto para obtener las características que utilizaran los distintos algoritmos para generar los modelos.
3. Clasificadores: El paso final sería el de obtener un buen clasificador. En este paso se deben realizar varias pruebas para obtener el mejor modelo consiguiendo el mejor ajuste posible de los parámetros para el algoritmo.

2.3.2.4 Preprocesado de texto

Este es el paso previo que ha de realizarse a la obtención de la bolsa de palabras y cuyo objetivo principal es obtener un texto "limpio" y reducir el espacio de características. Las principales trasformaciones a las que es sometido el texto son:

- Igualar todo el texto, es decir, transformar todo a minúsculas para evitar diferenciaciones de palabras iguales por el hecho de tener mayúsculas o minúsculas.
- Transformar los símbolos que puedan identificarse como palabras en su correspondiente palabra.
- Realizar correcciones ortográficas, como eliminar repeticiones de letras.
- Eliminación del resto de los símbolos que quedasen en el texto.

2.3.2.4.1 Tokenizacion, Stemming y StopWords

Una vez que el texto está limpio el siguiente paso es el de dividir el texto en palabras. Este paso también es conocido como *tokenizacion*, con el que se genera la bolsa, en algunos casos la bolsa definitiva.

En la mayoría de los casos mejora los resultados la eliminación de las palabras de parada. Las palabras de parada o *stopwords*, son consideradas palabras vacías, lo que significa no tienen significado en sí mismas. Las palabras de parada son los pronombres, preposiciones, artículos…

La *tokenizacion* puede ser el paso definitivo para obtener la bolsa de palabras, pero hay casos en que se realiza, si con ello se obtienen mejores resultados, una lematización o *stemming*. Esto consiste en obtener de cada una de las palabras la raíz de la misma haciendo que se reduzca el espacio de características.

2.3.2.5 Selección de características

Algunas de las posibilidades que tenemos para realizar la selección de las características de textos son [Medhat2014]:

- Frecuencia y presencia de términos: usar como características las palabras individuales o como *n-gramas* asociando a cada una el valor de su recuento de frecuencia. También podemos asignarlas un peso binario o bien usar su peso de frecuencia.
- *Parts of speech* (POS): encontrar adjetivos, ya que pueden ser indicadores de opinión.
- Frases y palabras de opinión: encontrar palabras que se usan comúnmente para expresar opiniones como “like”, “hate”….
- Negaciones: buscar la aparición de negaciones ya que estas pueden cambiar la orientación de una opinión.

Como se ha indicado, el modelo de texto más común es la bolsa de palabras, BOW. El proceso de extracción de características trabaja sobre el modelo de documento para obtener los vectores de características (matriz de términos) con los que alimentar el proceso de aprendizaje con el paso previo del preprocesado.

2.3.2.6 Matriz de términos

Una vez que tenemos la bolsa de palabras generada, a cada palabra tenemos que asignarle un valor para que lo use el algoritmo de clasificación. Con estos valores se crea la matriz (conjunto de vectores generados) que usará el algoritmo. Estas matrices pueden tener varias representaciones.

2.3.2.6.1 Matriz TF (Term Frecuency) - TF-IDF (Term Frequency – Inverse Document Frequency)

El paso final es obtener a partir de la bolsa una matriz de características TF-IDF o TF. TF-IDF es la abreviatura en ingles de frecuencia de término - frecuencia inversa de documento. Es un estadístico numérico, que lo que hace es reflejar la importancia de una palabra para un documento en una colección o corpus. El valor asignado a cada palabra aumenta proporcionalmente al número de apariciones de la palabra en el documento, pero se compensa con la frecuencia de esta palabra en el corpus. Esto ayuda a ajustar el hecho de que algunas palabras aparecen más frecuénteme que otras de manera general.

El cálculo de cada valor TF-IDF se realiza mediante el producto de dos medidas, la frecuencia de término y la frecuencia inversa de documento.

$$tf - idf(term, document) = tf(t, d)\ x\ idf(t)$$

La frecuencia de término, o número de veces que aparece la palabra en el documento, se multiplica por la componente idf() calculada como:

$$idf(t) = log\frac{1 + n_d}{1 + df(d, t)} + 1$$

n_d = número de documentos.
df(d,t) = número de documentos que contienen el término.

Los vectores resultantes se normalizan por la norma euclídea (norma l2).

$$v_{norm} = \frac{v}{\|v\|^2} = \frac{v}{\sqrt{{v_1}^2 + {v_2}^2 + \cdots .. + {v_n}^2}}$$

Dependiendo de los resultados obtenidos en la clasificación de los textos, se usará la matriz de frecuencia inversa o directamente solo la matriz de frecuencia, eliminando de la ecuación la componente inversa.

2.3.2.6.2 N-gramas y umbrales de frecuencias de términos

Otros factores que se pueden añadir para mejorar el algoritmo son los de añadir a la bolsa *n-gramas* y eliminar palabras dependiendo de las frecuencias.

Los *n-gramas* son agrupaciones de palabras que se realizan en el texto de forma ordenada que pueden tener diversos tamaños. Por ejemplo, un *n-grama* de orden 3 en la oración “El perro marrón” cogerá todos los posibles *n-gramas* siguiendo un orden. Estos serían: el, perro, marrón, el perro, perro marrón, el perro marrón.

La frecuencia de las palabras se utiliza para eliminar aquellas con una frecuencia de aparición demasiado alta o demasiado baja. Por ejemplo, puede darse el caso de querer eliminar las palabras que aparezcan en menos de 10 documentos (frecuencia mínima) o aquellas que aparezcan en más del 80% de los documentos (frecuencia máxima).

En el caso de las frecuencias reducimos el espacio de características, pero en el caso de los *n-gramas* lo aumentamos por lo que en ocasiones se obtienen mejores resultados sin utilizar *n-gramas*. Por esto es muy importante realizar selección de parámetros óptimos, para el algoritmo a utilizar.

2.3.2.7 Principales algoritmos de aprendizaje automático

Vamos a describir brevemente algunos de los algoritmos de clasificación que usaremos en el desarrollo del proyecto. Se han elegido los siguientes para ver como se comportan diversos algoritmos con distinto fundamento de entre los más populares para la clasificación de textos.

2.3.2.7.1 Naïve Bayes

Los clasificadores Naïve Bayes son un tipo de clasificadores probabilísticos simples, basados en el teorema de Bayes, con suposiciones de independencia fuerte entre las características. El clasificador de Bayes asume que la presencia o ausencia de una característica no se relaciona con la presencia o ausencia de otra característica, dada la variable clase.

Se basan en el teorema de Bayes:

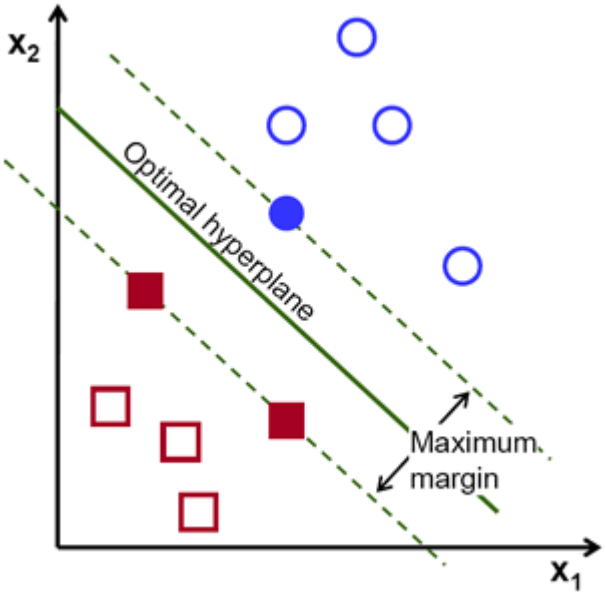
$$P(etiqueta|características) = \frac{P(c|e)P(e)}{P(c)}$$

P(etiqueta) es la probabilidad a priori de una etiqueta. P(características|etiqueta) es la probabilidad a posteriori de que un documento presente la característica c conocida su etiqueta e. *P(características)* es la probabilidad a priori de que se den un conjunto determinado de características. Dada la suposición Naïve que asume la independencia de todas las características dada la clase, la ecuación podría reescribirse:

$$P(etiqueta|características) = \frac{P(e)P(c_1|e) \ldots . P(c_n|e)}{P(c)}$$

2.3.2.7.2 SVM

SVM son las siglas de *Support Vector Machine*, que es un algoritmo de clasificación binario, sin embargo, también permite la clasificación multiclase mediante el método uno contra el resto [Mahesh2005]. Se basa en que dado un conjunto de puntos en el lugar N dimensional, el algoritmo SVM genera un hiperplano (N-1) que realiza una división de todos los puntos en dos grupos, maximizando la distancia del hiperplano a los puntos más cercanos de cada clase (caso linealmente separable). La siguiente figura ilustra el problema de dimensión 2:



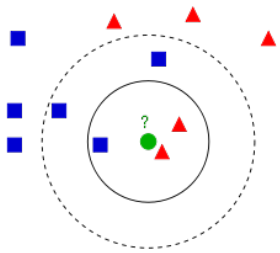
6 Ilustración del problema en SVM para dimensión 2

Los datos de texto son ideales para la clasificación SVM debido a la naturaleza dispersa de los mismos, en el cual algunas características son irrelevantes, pero tienden a relacionarse unas con otras entre sí, organizándose generalmente en categorías separables. SVM es capaz de construir una superficie de decisión no lineal en el espacio de características original, asignando las instancias de datos de forma no lineal a un espacio mediante una transformación implícita, donde las clases se puedan separar linealmente con un hiperplano.

2.3.2.7.3 K-Nearest Neightbors

El algoritmo K-NN, el de los k vecinos más cercanos, a diferencia de los anteriores es un algoritmo del grupo *lazy learning methods.* Los algoritmos de este grupo no generan un modelo fruto del aprendizaje con datos de entrenamiento, sino que el aprendizaje sucede en el momento que se prueban los datos de test.

El fin del algoritmo es clasificar cada nuevo dato en el grupo que le corresponda, según tenga k vecinos más cerca de un grupo u otro según la clase mayoritaria de sus *K* vecinos más próximos. Es decir, el algoritmo calcula la distancia del elemento nuevo a cada uno de los existentes, ordenando dichas distancias de menor a mayor para ir seleccionando al grupo al cual pertenece.

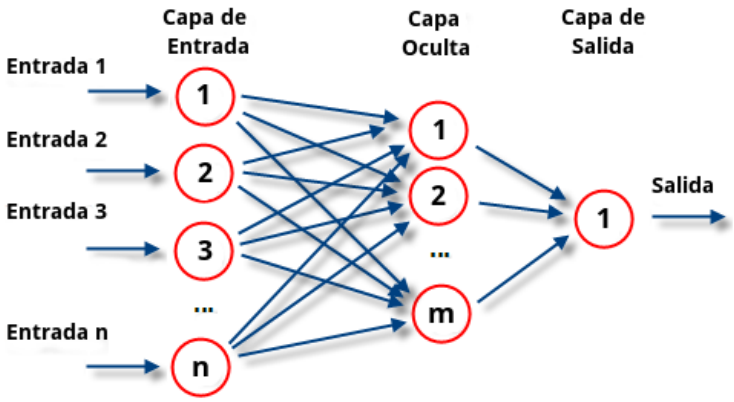


7 Ilustración del algoritmo KNN para K=3 y K=5

Dada la ilustración, la primera circunferencia representaría el caso K=3 donde el punto verde quedaría asociado en el grupo de los triángulos que son mayoría. Sin embargo, en el caso K=5 cambiara de agrupación a los cuadrados al tener 3 vecinos más cercanos sobre 2 de los triángulos.

2.3.2.7.4 MLP (Multi-Layer Perceptron)

Este algoritmo pertenece al grupo de las redes neuronales. El MLP o perceptrón multicapa es una red neuronal artificial formada por múltiples capas que es capaz de resolver problemas que no son linealmente separables y que consiste en la estimación de sus parámetros libres, es decir, los pesos de la red [Monroy-de-Jesús2015]. Es un algoritmo de *backpropagation* (propagación hacia atrás), que está basado en una técnica de descenso por gradiente que utiliza la minimización del error cuadrático medio, mediante un proceso iterativo.



8 Ilustración de las capas del algoritmo MLP

El conjunto de capas formadas por neuronas (elementos de procesado) que utiliza el algoritmo se puede configurar agrupándose en tres tipos distintos:

- Capa de entrada: es esta capa no hay procesamiento. Lo forman las neuronas las cuales introducen los patrones de entrada a la red. Corresponde al número de características.
- Capa oculta: es una capa intermedia de procesamiento de información. La mayoría de los problemas suelen encontrar la solución con una sola capa oculta. También es importante decidir el número de neuronas ocultas óptimo, que es un proceso más complicado. Cada neurona en esta capa transforma los valores

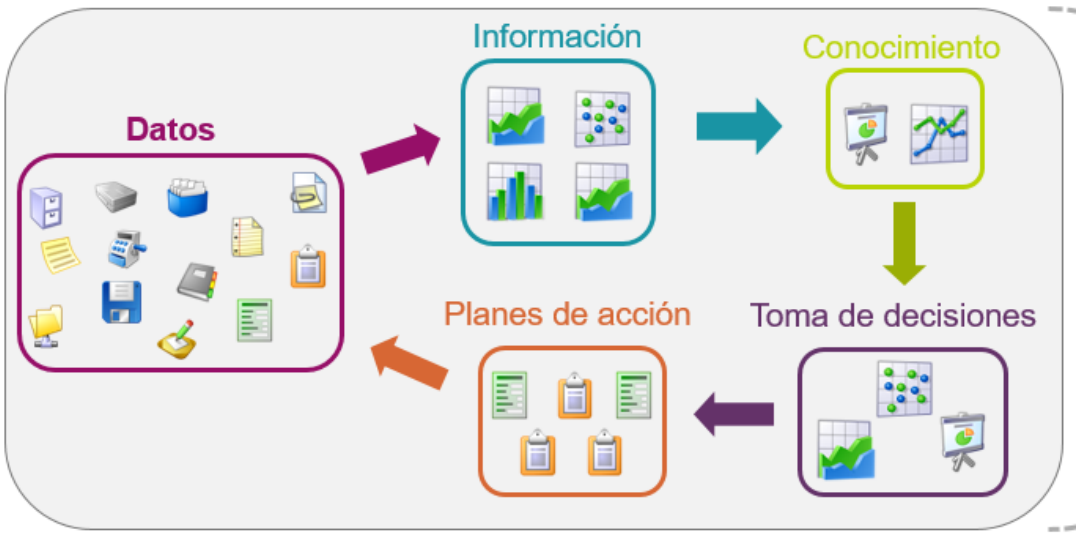
de las capas anteriores en una suma de pesos lineal seguido por una función de activación no-lineal.

- Capa de salida: recibe los valores de la última capa oculta y los transforma en los valores de salida

3 Arquitectura Big Data para el análisis de sentimiento

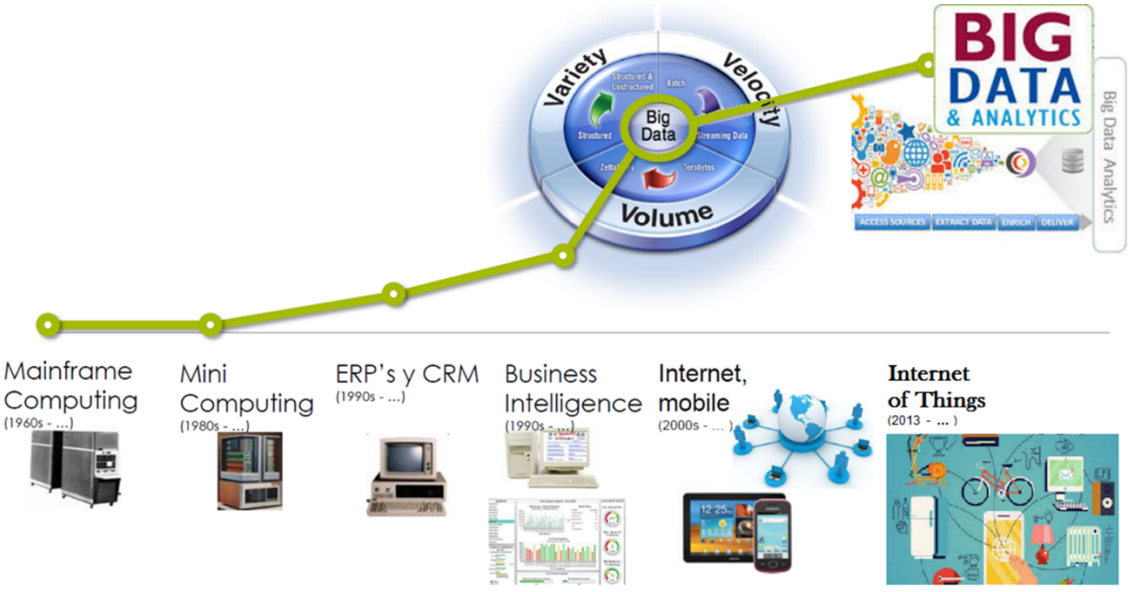
3.1 Introducción al Big Data

El *Big Data* nace a partir de la inteligencia de negocio (BI – *Business Intelligence*). El objetivo del BI es disponer de información con la que se pueda conocer el negocio, tomar decisiones y crear planes de acción con los que generar nuevos datos de interés para la organización. Para lo cual es necesario analizar todos los datos disponibles de la organización y transformar los mismos siguiendo un proceso básico que pasará por las fases de:



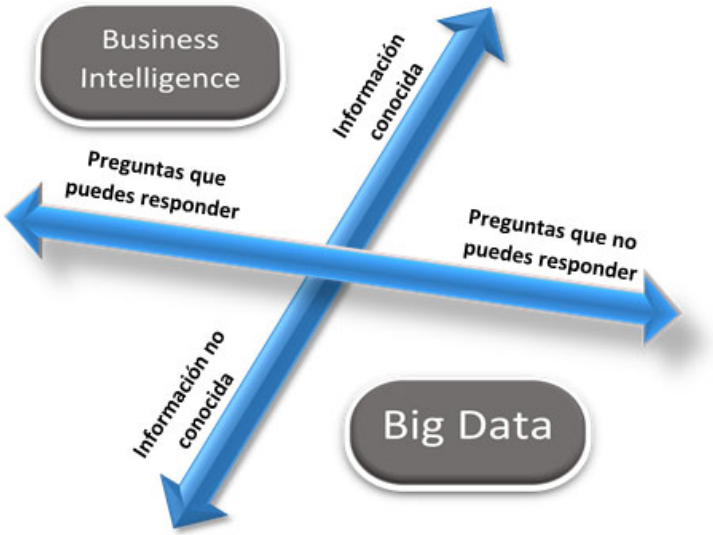
9 Fases de análisis del dato

Con la aparición de internet y los dispositivos móviles la cantidad y diversidad de datos generados ha llegado al punto en el que se ha necesitado desarrollar nuevas técnicas para procesar toda esta gran cantidad de datos. Ahí es donde aparece el *Big Data*.



10 Evolución y diversidad de dato

Entonces, las diferencias entre el Big Data y el BI surgen según el tratamiento y los datos utilizados.



11 Diferencias entre el BI y el Big Data

En los entornos *Big Data*, los datos se almacenan en un sistema de ficheros distribuido, en lugar de un servidor central. Por ello, al estar dispersos, se crea un entorno mucho más seguro y también más flexible consiguiendo:

- Las soluciones *Big Data*, llevan las funciones de proceso a los datos, en lugar de los datos a las funciones. Al estar el análisis centrado en torno al dato, esto permite manejar cantidades más grandes de información de forma más ágil.
- *Big Data* puede analizar datos en diferentes formatos, tanto estructurados como no estructurados. El volumen de datos no estructurados está creciendo a niveles muy superiores.
- *Big Data* permite procesar fuentes tanto históricas como en tiempo real. BI se limita a datos históricos y por ello, los resultados pueden no ser 100% actuales.
- La tecnología de Big Data emplea conceptos de procesamiento paralelo masivo (MPP), in *memory*, lo cual mejora la velocidad en el análisis

Según el tipo de datos a procesar, podremos optar por utilizar una solución de BI o *Big Data*:

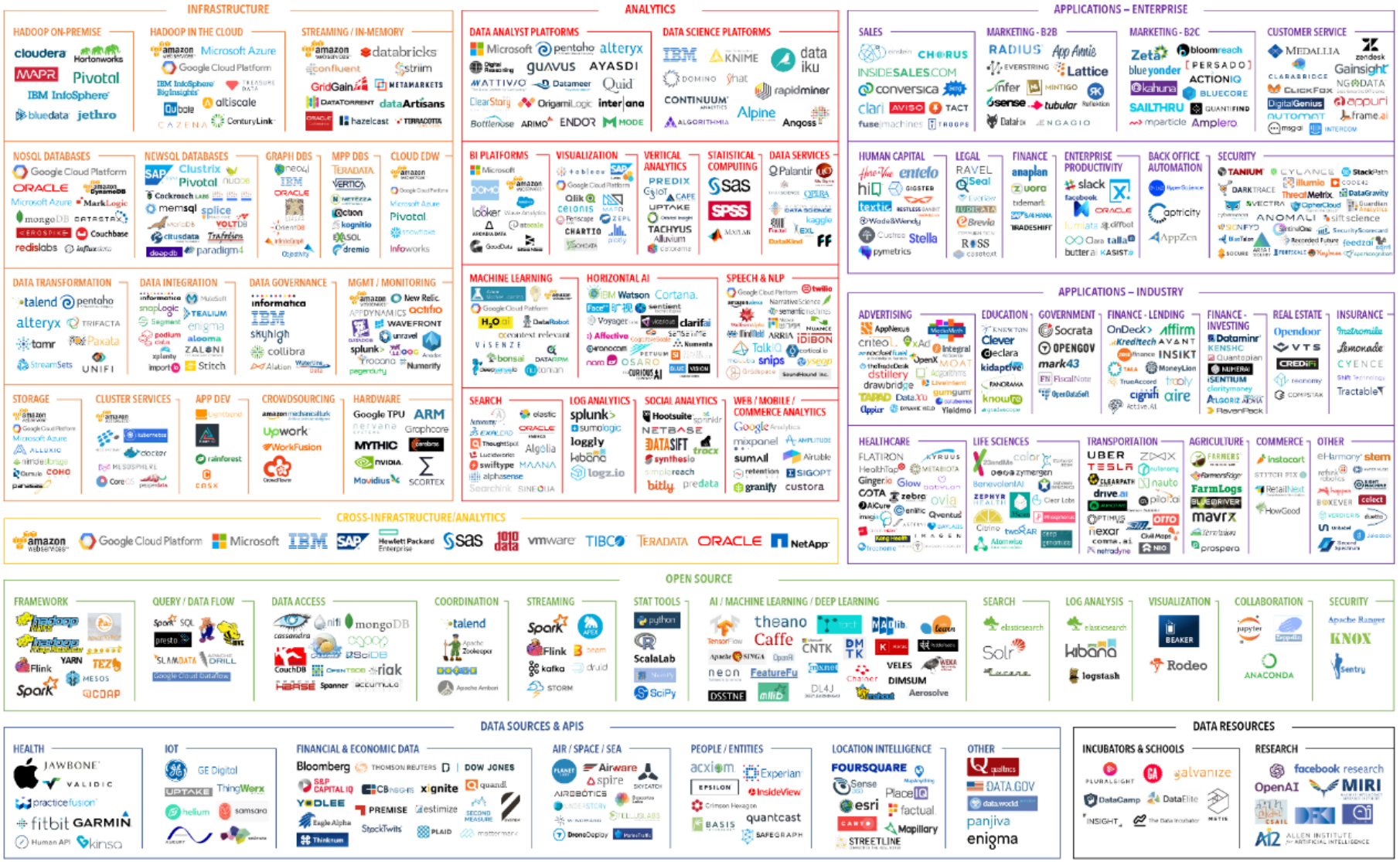
- Estructurados: Datos que contienen un tipo de dato definido, un formato definido, una estructura definida, …
- Semi-estructurados: Ficheros de texto sin estructura formal, que permiten habilitar la lectura de datos con etiquetas/marcadores.
- No-estructurados: Datos sin una estructura inherente, normalmente son almacenados como diferentes tipos de ficheros.

12 Tipos de datos

El BI se fundamenta en estructurar información y presentarla para la toma de decisiones. El Big Data se fundamenta en la gran capacidad que tiene para realizar diferentes tratamientos sobre datos no-estructurados.

Así el ecosistema de herramientas de Big Data va aumentando cada año llegando en 2017 a existir cientos de herramientas para el estudio de los datos en sus distintos ámbitos.

BIG DATA LANDSCAPE 2017



V2 – Last updated 5/3/2017

 mattturck.com/bigdata2017

FIRSTMARK
EARLY STAGE VENTURE CAPITAL

3.2 Arquitectura propuesta

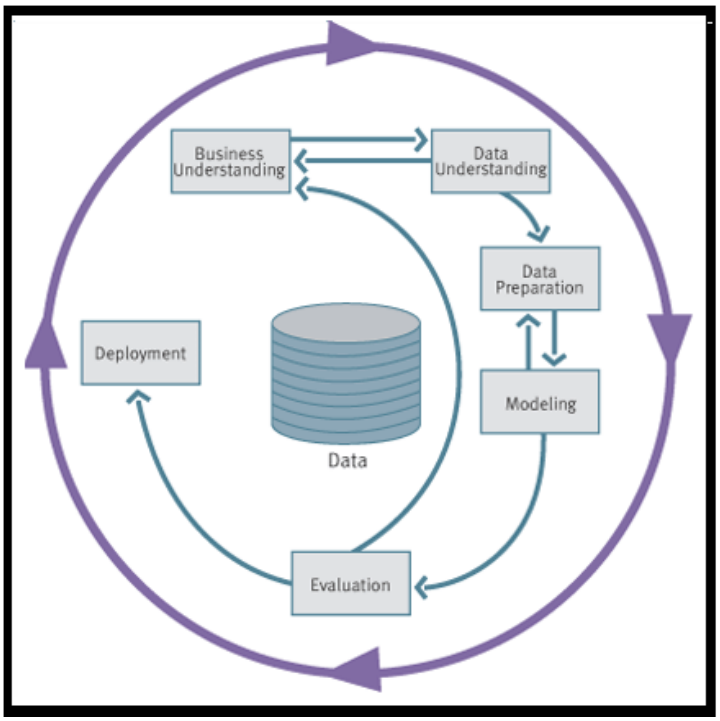
3.2.1 Introducción

El sistema que proponemos consistirá en una extracción de datos recopilados de la plataforma *Twitter*. Estos datos contendrán información sobre los *tweets* publicados en la plataforma, tanto el propio texto del tweet como sus metadatos. Posteriormente se realizará un procesado de estos datos y finalmente se realizará una clasificación de los mismos con los métodos ya mencionados.

3.2.2 Metodología de trabajo

Para implementar una tecnología ya sea en un negocio o simplemente para su desarrollo es necesario una metodología. La mayor parte de las consultoras que están especializadas en una tecnología cuentan, según los proyectos que aborden, con al menos una metodología. Contar con una metodología en un desarrollo se ha convertido en un requisito tan importante como la carta de presentación de las empresas.

En los proyectos de minería de datos, la metodología más utilizada y que vamos a adoptar es la CRISP-DM (*Cross Industry Standard Process for Data Mining*). CRISP-DM es la metodología que se utiliza de facto, de una forma u otra, en los proyectos de análisis de datos que se pretendan abordar con seriedad y asegurando la calidad de los resultados. CRISP-DM proporciona una descripción normalizada del ciclo de vida de un proyecto estándar de análisis de datos, de forma análoga a como se hace en la ingeniería del software con los modelos de ciclo de vida de desarrollo de software.



14 Ciclo metodología CRISP-DM

Este estándar incluye un modelo y una guía, que está estructurado en seis fases siendo algunas de ellas bidireccionales. Esto quiere decir que en ciertas fases será posible revisar de forma parcial o totalmente las fases anteriores.

- Comprensión del negocio (Objetivos y requerimientos desde una perspectiva no técnica)
    - Establecimiento de los objetivos
    - Evaluación de la situación
    - Establecimiento de los objetivos de la minería de textos.
    - Generación del plan de proyecto

- Comprensión de los datos (Familiarizarse con los datos teniendo en cuenta los objetivos)
    - Recopilación inicial de los datos
    - Descripción de los datos
    - Exploración de los datos.
    - Verificación de calidad de los datos

- Preparación de los datos (obtención del *dataset*)
    - Selección de los datos
    - Limpieza de datos
    - Construcción de los datos
    - Integración de los datos
    - Formateo de los datos

- Modelado (Aplicación de técnicas de minería al *dataset*)
    - Selección de la técnica de modelado
    - Diseño de la evaluación
    - Construcción del modelo
    - Evaluación del modelo

- Evaluación (Determinar si los modelos desarrollados son útiles)
    - Evaluación de los resultados
    - Revisar el proceso
    - Establecimiento de los siguientes pasos o acciones

- Despliegue (Exportar la utilidad del modelo)
    - Planificación del despliegue
    - Planificación de la monitorización y del mantenimiento
    - Generación del informe final
    - Revisión del proyecto

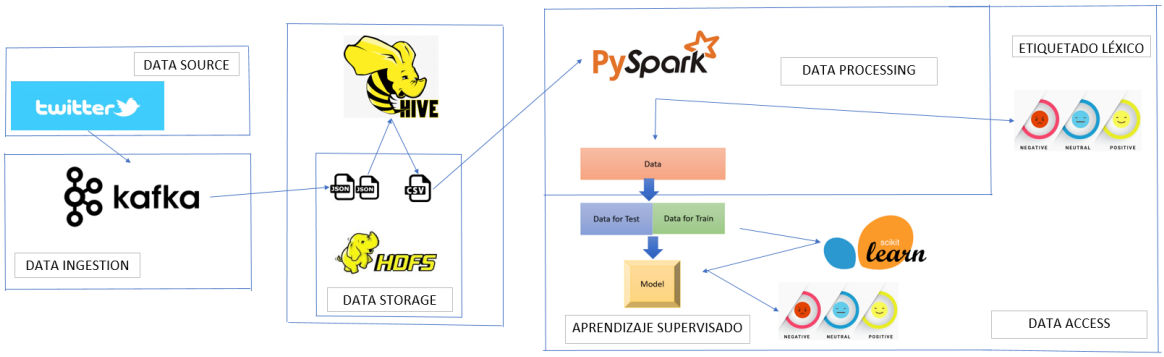
3.2.3 Diseño propuesto

Debido a las diferentes fases de la gestión y las diferentes formas en las cuales se encontrarán los datos, se va a utilizar una arquitectura *data lake. El data lake* es un repositorio central donde se almacena toda la información de una organización, en este caso nuestro sistema, sin importar su futuro u origen.

Las capas más comunes de las cuales dispone esta arquitectura son las siguientes:

- Ingesta de datos (*Data Ingestion*): esta es la capa en la cual se extrae el *raw data* del origen y se vuelca en la capa de almacenamiento.
- Almacenamiento de datos (*Data Storage*): esta es la capa en los cual los datos, volcados por la ingestión son almacenados. Estos datos pueden ser tanto estructurados como no estructurados.
- Procesamiento de datos (*Data Processing*): una vez que se han analizado los datos, se puede ver la necesidad de procesar determinados sets de datos para alojarlos en una capa de uso recurrente. En la misma pueden tener lugar procesos de calidad de datos, integridad y otras adaptaciones para que tengan acceso otros usuarios.
- Acceso a los datos (*Data Access*): esta es la capa más avanzada donde finalmente los datos se ponen a disposición de los analistas de negocio.

En nuestro sistema vamos a utilizar las cuatro capas, en cada una de las cuales se expondrá las herramientas utilizadas y el funcionamiento.



15 Arquitectura propuesta

Se ha diseñado un sistema que utilizará datos que se descargaran directamente de la plataforma *Twitter* para posteriormente procesar y realizar una clasificación de los mismos, analizando su sentimiento.

Nuestro origen de datos será la plataforma Twitter. Twitter es una red que permite enviar mensajes de texto plano de hasta 280 caracteres (140 inicialmente) que se muestran en la página principal del usuario, denominados *tweets*. Los usuarios conectados entre sí pueden ver los *tweets* que publiquen. Estos *tweets* son la información que vamos a extraer de la plataforma ya que en su mayor parte llevan implícito un sentimiento.

La extracción de los *tweets* se va a realizar utilizando la herramienta Apache Kafka [Kafka]. Kafka es un proyecto de intermediación de mensajes de Apache. Con él, extraeremos los *tweets* en formato json y los volcaremos en HDFS.

En HDFS almacenaremos el *raw data*. Mediante Apache Hive [Hive], una infraestructura de almacenamiento de datos construida sobre Hadoop (framework de software que soporta aplicaciones distribuidas) para proporcionar agrupación, consulta, y análisis de datos, crearemos las bases de datos necesarias para estudiar en el grado que veamos conveniente todo el *metadata* que acompaña el *tweet*. Con el también generaremos los ficheros csv previos a la siguiente fase.

El preprocesado de los *tweets* se realizará con *Spark*, con el cual se generarán los datos de entrada a los algoritmos que crearán los clasificadores y el etiquetado para hacer el análisis de sentimiento.

Finalmente, para realizar la clasificación se utilizarán los algoritmos de *Scikit-learn* que es una biblioteca de aprendizaje automático muy potente de software libre desarrollada en Python.

Para la parte de procesamiento de y acceso a los datos se desarrollará una herramienta en Python con PyQt5 [PyQt5] para dar una gran facilidad de uso, que todo pueda ser utilizado con otros conjuntos de datos y que sea fácilmente escalable.

Todos estos aspectos serán desarrollados en mayor medida en el punto 4.

4 Desarrollo de la arquitectura

4.1 Ingesta de datos

La ingesta de datos se va a realizar desde el origen de datos que hemos elegido y que será la plataforma Twitter. Para la extracción se va a utilizar la plataforma Apache Kafka, con ella lo que haremos será generar ficheros en formato json, que contendrán datos del tweet junto con sus metadatos de forma diaria.

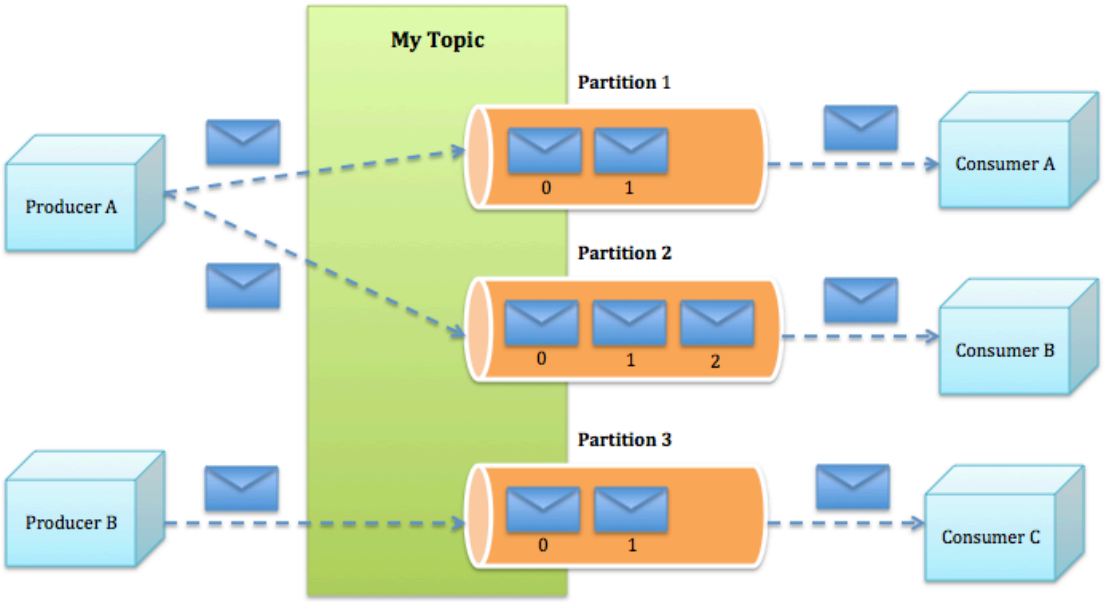
4.1.1 Tecnología utilizada: Apache Kafka

Apache Kafka es un sistema de mensajería distribuido con publicación-suscripción con una cola robusta con lo que puede manejar una gran cantidad de datos y pasar mensajes de un punto a otro. Kafka es adecuado tanto para consumo en línea como fuera de línea. Los mensajes son conservados en el disco y replicados en el clúster. Este sistema se construye sobre el servicio de sincronización Zookeeper.

Los aspectos por destacar en el sistema serían:

1. *Topics*: son una corriente de mensajes perteneciente a una categoría en particular. Es a través de ellos por donde se envían o reciben los mensajes.
2. *Brokers*: son los encargados del mantenimiento de los mensajes publicados.
3. Productores: son los encargados de la publicación de los mensajes en Kafka que se los envían a los *brokers*.
4. Consumidores: leen los datos de los *brokers* y están suscritos a los *topics*.

La arquitectura del sistema sería la siguiente:



16 Esquema de funcionamiento de Apache Kafka

Existen dos posibilidades de conexión que son conexión *source* y conexión *sink*:

- Conexión *source*: esta conexión se crea para obtener mensajes desde una fuente. Esta es la conexión que utilizaremos, con la cual extraeremos los *tweets*.
- Conexión *sink*: esta conexión se crea para enviar mensajes a un destino.

4.1.2 Extracción de los datos

Para realizar la extracción de los datos vamos a usar la API de *Twitter*, un conector de *Twitter* para Kafka llamado *Twitter* Kafka *Connect* y la plataforma Confluence. Confluence es una plataforma de transmisión creada por los fundadores de Apache Kafka. Permite el acceso sencillo a datos como transmisiones en *stream*.
Los servicios de Confluence que usaremos son Zookeeper, Kafka y Schema-Registry (proporciona una capa de servicio para los metadatos).
Como solo necesitamos extraer datos, no enviar, vamos a crear una conexión *source* (de *Twitter* a Kafka). Los pasos de configuración que se han realizado son:

1. Descarga de la versión *open source* de Confluence.
    a. Descargar del paquete de la web oficial.
    b. Descompresión del archivo con tar -xzvf

2. Arranque de los servicios necesarios, que son los tres servicios que hemos mencionado (Zookeeper, Kafka y Schema-Registry), utilizando el comando siguiente comando desde el directorio bin de Confluence.

    bin/confluent start schema-registry

3. Descarga del conector de *Twitter*, para lo cual nos descargamos el proyecto kafka-connect-twitter. Es un proyecto libre que permite la conexión con la API de *Twitter*. Lo trasladamos al directorio de Confluence y lo extraemos.

4. Descarga e instalación de maven, en este caso lo usaremos para generar el .jar a partir del proyecto de kafka-connect-twitter.

5. Mediante maven generamos el .jar desde la ruta del proyecto kafka-connect-*Twitter* con el comando:

    mvn clean package

    Posteriormente al crear la conexión aparecerá un error debido a una de las versiones especificadas en el pom.xml. La versión que hay que modificar es la de Guava, cambiamos la del fichero por la última que en nuestro caso es la “23-jre”.

6. Una vez creado el .jar hay que añadirlo al CLASSPATH exportándolo, para lo que usamos el comando:

export CLASSPATH=/opt/confluent/kafka-connect-twitter-master/target/kafka-connect-twitter-0.1-jar-with-dependencies.jar

7. Para crear una conexión a *Twitter* necesitamos conectarnos con la API para lo cual en https://apps.twitter.com/app/ , usando nuestra cuenta personal de *Twitter* creamos una nueva APP y extraemos los parámetros que necesitamos para crear nuestra conexión *source* que son claves de conexión :
    a. twitter.consumerkey
    b. twitter.consumersecret
    c. twitter.token
    d. twitter.secret

8. Una vez que tenemos todo lo necesario configuramos los ficheros que necesitamos para crear la conexión. Como ya hemos indicado vamos a crear una conexión *source* porque vamos a extraer datos. Para crear esta conexión es necesario configurar el fichero *twitter-source.properties* con los parámetros:

```
name=twitter-source
connector.class=com.eneco.trading.kafka.connect.twitter.TwitterSource
Connector
tasks.max=1
topic=twitter
twitter.consumerkey=****
twitter.consumersecret=****
twitter.token=****
twitter.secret=****
track.terms=test
language=en
```

   Se crea un nuevo tópico llamado *Twitter* al cual conectaremos el consumidor para la extracción de los *tweets* en formato Json. El filtrado que se hemos hecho es que el idioma venga marcado a inglés.

9. Finalmente, como es una conexión *source* necesitamos crear dos aspectos:

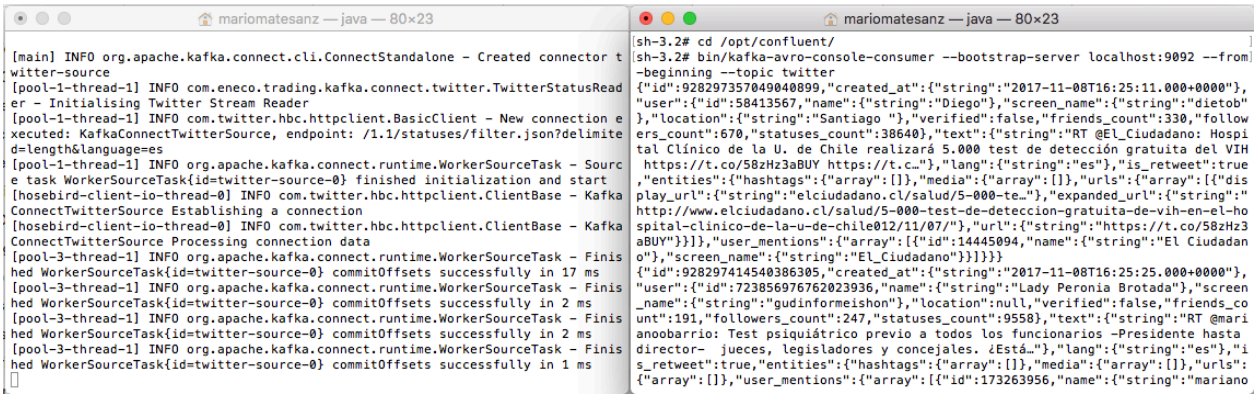
    1. La conexión con los parámetros que hemos definido para conectar con la API de *Twitter* que será el productor. Para crearla usamos el comando:

       bin/connect-standalone kafka-connect-twitter-master/connect-source-standalone.properties kafka-connect-twitter-master/twitter-source.properties

    2. El consumidor al que se le pasarán los mensajes del productor a través del tópico “twitter”. Para crear el consumidor en el tópico twitter usamos el comando:

bin/kafka-avro-console-consumer --bootstrap-server localhost:9092 --from-beginning --topic twitter

Lo que hacemos como paso final, es generar un fichero .json con el volcado de esta conexión extrayendo así todos los *tweets* en formato .json con sus correspondientes metadatos. Estos ficheros son los que volcaremos al sistema HDFS y con los que trabajaremos con Hive.



17 Extracción de los tweets

4.2 Almacenamiento de datos

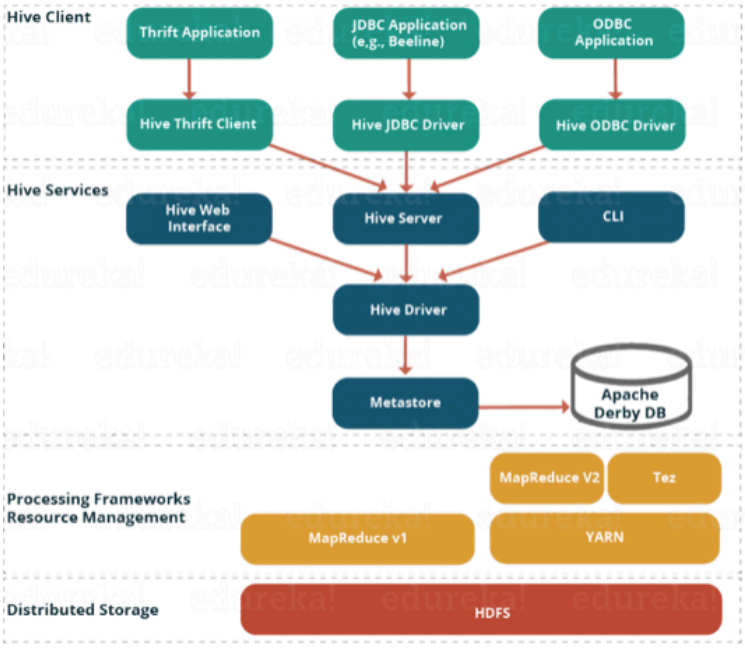
Para gestionar los datos obtenidos y extraer la información necesaria de los mismos para realizar una clasificación vamos a utilizar Hive. Hive es una infraestructura de almacenamiento de datos que está construida sobre Hadoop y que proporciona agrupación, consulta y análisis de datos. Hive funciona sobre HDFS que es la plataforma que almacena los datos y necesaria para su funcionamiento.

4.2.1 Tecnología utilizada: Apache Hive y Apache HDFS

Hive y HDFS son dos plataformas de Apache que mediante el uso de ambas podemos

almacenar y gestionar grandes volúmenes de datos. HDFS es simplemente el sistema de archivos, escalable y distribuido de Hadoop. Hive ofrece una tecnología del tipo data *warehousing* (almacén de datos) y proporciona un lenguaje de consulta declarativo (HiveSQL) muy similar a SQL además de una interfaz de consulta SQL sobre el sistema HDFS.

Podemos considerar Hive como una abstracción de alto nivel de *map reduce*. Lo que hace su motor de ejecución es transformar las sentencias HQl en Jobs Map Reduce que se ejecutan directamente sobre la plataforma Hadoop sobre la que va montado



18 Arquitectura Hive

En cuanto a la arquitectura de Hive podemos decir:

- Mediante las interfaces de usuario se habilitan el acceso a los recursos de Hive.
- El *metastore* utiliza una base de datos para almacenar los esquemas de tablas de Hive. Actúa como diccionario de datos y almacena metadatos sobre las bases de datos, tablas, columnas…
- El motor de HiveQL recibe la consulta HiveQL, que con la ayuda de la información del *metastore*, transforma esta consulta en una configuración *map reduce.* Es el componente *map reduce* el que compila esta consulta y la optimiza de cara a su posterior ejecución.
- En el último paso el motor de ejecución recibe la consulta y obtiene los correspondientes resultados.

4.2.2 Preparación de los datos

Como ya hemos dicho, lo que hemos hecho para almacenar los datos ha sido alojar los ficheros con información de *tweets* que hemos descargado mediante la plataforma Kafka. Todos los ficheros han sido alojados sobre un directorio en HDFS y serán procesados mediante Hive para finalmente generar nuevos ficheros csv con la información que necesitamos para la clasificación. Los pasos que hemos realizado han sido:

1. Colocar los ficheros .json descargados en el directorio de HDFS:
   /user/collector/dataKafka/tweets_raw

2. Generar una nueva base de datos en nuestro sistema:

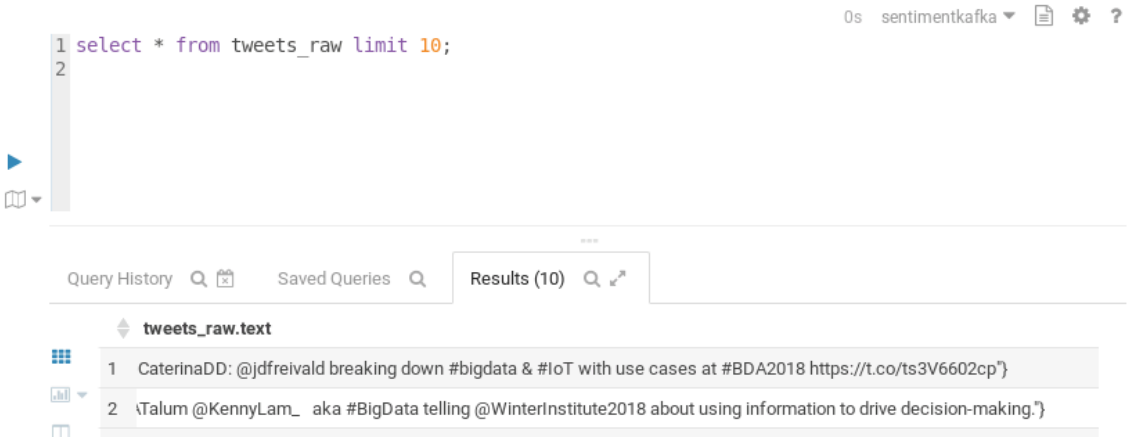
   CREATE DABASE SENTIMENTKAFKA;

3. Tenemos que añadir el fichero .jar para la lectura de los .json descargados ya que tienen un formato especial. Este fichero ha sido colocado en un directorio de HDFS para que pueda ser accedido desde Hive.

   ADD JAR hdfs://quickstart.cloudera:8020/user/collector/hive-serdes-1.0-SNAPSHOT.jar;

4. Crear la nueva tabla que accede a los ficheros descargados desde donde se accederá a la información.

```
CREATE EXTERNAL TABLE tweets_raw (
  id BIGINT,
  created_at STRUCT<string:STRING>,
  text STRUCT<string:STRING>,
  user STRUCT<
        location : STRUCT<string:STRING>
        >
)
ROW FORMAT SERDE 'com.cloudera.hive.serde.JSONSerDe'
LOCATION '/user/collector/dataKafka/tweets_raw';
```



19 *Selección de datos en Hive*

Para este desarrollo se han escogido solo algunos de los campos del fichero ya que no todos son necesarios, en este caso, si quisiéramos acceder a todos los datos crearíamos una tabla nueva como la siguiente:

```
CREATE EXTERNAL TABLE tweets_rawAll (
  id BIGINT,
  created_at STRUCT<string:STRING>,
  text STRUCT<string:STRING>,
  user STRUCT<
        id : BIGINT ,
        name : STRUCT<string:STRING>,
        screen_name : STRUCT<string:STRING>,
        location : STRUCT<string:STRING>,
        verified : boolean,
        friends_count : int,
```

```
        followers_count : int,
        statuses_count :int

        >,
        lang STRUCT<string:STRING>,
        is_retweet boolean,
        entities STRUCT<
                hashtags                                                :
STRUCT<array:ARRAY<STRUCT<text:STRUCT<string:STRING>>>>,
                media : STRUCT<array:ARRAY<STRUCT<

        display_url:STRUCT<string:STRING>,

        expanded_url:STRUCT<string:STRING>,
                                                id : BIGINT,

        type:STRUCT<string:STRING>,
                                                url:STRUCT<string:STRING>
                >>>,
                urls : STRUCT<array:ARRAY<STRUCT<

        display_url:STRUCT<string:STRING>,

        expanded_url:STRUCT<string:STRING>,
                                                url:STRUCT<string:STRING>
                >>>,
                user_mentions : STRUCT<array:ARRAY<STRUCT<
                                                id : BIGINT,
                                                name
:STRUCT<string:STRING>,
                                                screen_name
:STRUCT<string:STRING>
                >>>
        >

)
ROW FORMAT SERDE 'com.cloudera.hive.serde.JSONSerDe'
LOCATION '/user/collector/dataKafka/tweets_raw';
```

5. Podemos crear una vista como paso intermedio para acceder solamente al texto final con el que nos queremos quedar:

```
CREATE VIEW solotexto AS SELECT text FROM tweets_raw;
```

6. Finalmente generamos el fichero con todos los datos de la tabla para lo que usamos el terminal desde donde con el comando hive podemos lanzar consultas en Hive y volcar el resultado sobre un fichero.

hive -e 'ADD JAR hdfs://quickstart.cloudera:8020/user/collector/hive-serdes-1.0-SNAPSHOT.jar;
;select text sentimentkafka.solotexto;' > /home/cloudera/TFM/tweets.csv

```
[root@quickstart cloudera]# hive -e 'ADD JAR hdfs://quickstart.cloudera:8020/user/collector/hive-serdes-1.0-SNAPSHOT.jar;sele
ct text from sentimentkafka.solotexto;' > /home/cloudera/TFM/tweetsTexto.csv

Logging initialized using configuration in jar:file:/usr/lib/hive/lib/hive-common-1.1.0-cdh5.10.0.jar!/hive-log4j.properties
converting to local hdfs://quickstart.cloudera:8020/user/collector/hive-serdes-1.0-SNAPSHOT.jar
Added [/tmp/e5d3b63f-d1e8-45df-8ed8-10d9d118ae8b_resources/hive-serdes-1.0-SNAPSHOT.jar] to class path
Added resources: [hdfs://quickstart.cloudera:8020/user/collector/hive-serdes-1.0-SNAPSHOT.jar]
OK
Time taken: 2.957 seconds, Fetched: 41247 row(s)
```

20 Consulta Hive desde terminal

Este proceso que hemos simplificado para proponer un almacenamiento utilizando una infraestructura Big Data se puede complicar introduciendo en el proceso de creación de las tablas particiones dinámicas creadas por fecha. Esto mejoraría el almacenamiento de grandes cantidades de datos pudiendo acceder a ellos por fechas facilitando el estudio de estos. Hive es una herramienta muy potente que incluso podría realizar una aproximación al etiquetado basado en diccionarios. En la actualidad la mayoría de las empresas que se dedican al Big Data almacenan sus datos sobre HDFS y utilizan plataformas como Hive o Impala para el acceso a los mismos.

4.3 Procesamiento y acceso a los datos

4.3.1 Introducción

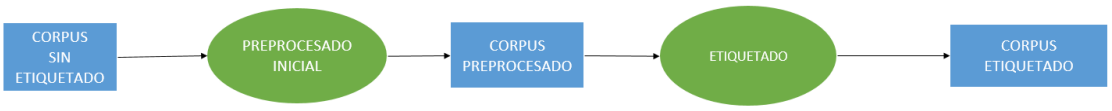
En este apartado se va a explicar cómo se ha realizado la parte de procesamiento y acceso a los datos para lo cual se ha utilizado el lenguaje de programación Python. Se expondrán los procesos utilizados, así como los algoritmos (se presentarán en este apartado y se explicará su configuración en el apartado 5 donde se desarrolla la interfaz) que posteriormente serán introducidos en la herramienta de clasificación.

4.3.2 Etiquetado basado en diccionarios

4.3.2.1 Introducción

Como ya hemos indicado el etiquetado es un proceso por el cual utilizando diccionarios de palabras se realiza una clasificación de un texto, en nuestro caso de un *tweet* en positivo, negativo o neutro. Este proceso de etiquetado se divide en dos partes, la primera en la que se preprocesa el texto para limpiarlo y dividir en palabras y la segunda para comparar las palabras obtenidas con las de unos diccionarios para realizar el etiquetado. Así el proceso de etiquetado se dividiría en dos fases:

1. Preprocesado: limpieza del texto para obtener un conjunto de palabras por cada tweet para buscar.
2. Etiquetado: proceso de comparación con los diccionarios para obtener la categoría del tweet.



21 Proceso del etiquetado

4.3.2.2 Preprocesado

Este procesado se realiza para limpiar el texto obtenido del tweet, para extraer un listado de palabras, pudiendo así ser comparado con las de los diccionarios.

1. Lo primero que hacemos es transformar todo en minúsculas, así evitamos que dos palabras que sean igual sean distinguidas por el hecho de tener una letra de distinto tipo.

2. Emoticonos: los emoticonos son una fuente muy importante para la determinación del sentimiento, son caracteres que representan una emoción. La mayoría de los iconos se pueden agrupar en dos clases, por ello, se ha distinguido entre emoticonos que aportan un sentimiento positivo y lo que aportan uno negativo. Los positivos se sustituyen por "emotpos" y los negativos por "emotneg". Esto se hace para evitar ruido en los algoritmos de clasificación ya no vamos a considerar diferentes grados de positividad o negatividad en los emoticonos. Posteriormente a la hora de buscar las palabras en el diccionario estas serán consideradas como una palabra positiva o negativa según el caso.

3. Transformación de simbologías de *Twitter*. El modelo del lenguaje de *Twitter* tiene unas características únicas que transformamos para intentar buscar palabras escondidas en ellas. Estas características únicas son:

    - *Hashtags*: son palabras o frases sin espacios seguidas con el prefijo "#". Reemplazamos el *hashtag* por la palabra que contiene el *hashtag*.

    - Menciones: son los nombres de los usuarios que pueden aparecer en el tweet y con el fin de dirigir el mensaje. Van prefijados con el símbolo "@". Todos estos los eliminamos ya que son palabras que no estarán en los diccionarios.

    - Letras repetidas: puede darse el caso de encontrar una palabra en la que una de sus letras está repetida cierto número de veces. Lo que hacemos en este caso será sustituir esa cantidad de letras por una sola. Existen palabras, sobre todo en inglés que es nuestro caso, con dos letras seguidas iguales por lo que en el diccionario se buscara la palabra con y sin las repeticiones.

- *Links*: muy usualmente en *Twitter* se incluyen *urls*. Todas estas *urls* las eliminamos ya que no son palabras que puedan encontrase en el diccionario.

4. Finalmente, el último paso consiste en eliminar todos los símbolos que no aportan información y pueden ir acompañando en el tweet a alguna palabra modificándola, como pudieran ser signos de admiración, de exclamación, de puntuación…

Una vez que tenemos un texto limpio con solamente palabras, ya podemos proceder a realizar el etiquetado.

4.3.2.3 Etiquetado de los tweets

El etiquetado del texto se realiza mediante asociación léxica. Tenemos un diccionario con palabras de carácter positivo y otro con palabras de carácter negativo. Por comparación de cada palabra del tweet con las de los diccionarios obtenemos el sentimiento. Calculamos la suma de palabras positivas menos negativas, en caso de ser 0 el sentimiento es neutro.

Los diccionarios que hemos utilizado han sido buscados en internet [liub/FBS], en nuestro caso son unas listas que han sido compiladas por Bing Liu y Minqing Hu desde la publicación de su primer artículo [Hu2004] relacionado con el análisis de sentimiento. Ambos son profesores distinguidos de la Universidad de Illinois en Chicago con numerosos artículos en su haber sobre análisis de sentimiento y aprendizaje automático por esto se han considerado sus recopilaciones muy apropiadas para nuestro uso.

Por otro lado, había otras recopilaciones muy interesantes como las realizadas por la Universidad de Pittsburgh [mpqa] que ha creado numerosos tipos de léxicos no solo para el ámbito del análisis de sentimiento. En nuestro caso la versión del léxico que también se estudió fue la utilizada en [Wilson2005] pero nos decantamos por la primera debido a su mayor amplitud de términos.

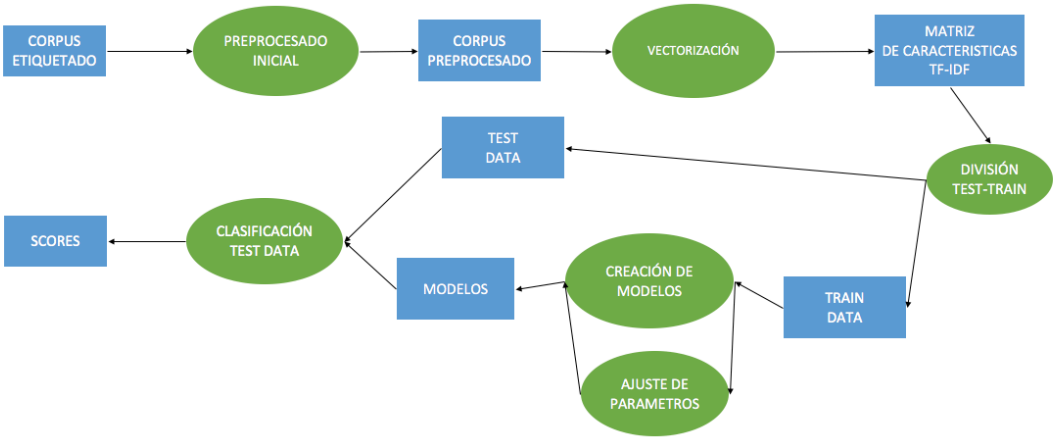
El mecanismo de etiquetado ha consistido en extraer como ya hemos visto cada palabra del tweet y posteriormente buscar cada una de ellas en ambos diccionarios. Si una palabra era encontrada en uno de los dos diccionarios se aumentaba en recuento o se disminuía considerándose finalmente positivo o negativo si era mayor o menor que 0. En caso de que el recuento fuera 0 se consideraría neutro. Como ya hemos visto en el preprocesado hay un caso especial, que es el de los emoticonos, si las palabras que hemos sustituido (emotpos, emotneg) se encontrasen se consideraría al caso como una palabra positiva o negativa y se modificaría el recuento de igual modo.

Dependiendo del valor dado en la interfaz para las categorías (binario, ternario) el programa se quedará con los tres tipos de etiquetado o descartará los neutros.

4.3.3 Aprendizaje supervisado

4.3.3.1 Introducción

Como ya se explicó, el aprendizaje supervisado es un proceso por el cual, a partir de un conjunto etiquetado, podemos crear un modelo que nos permita realizar predicciones. En nuestro caso el proceso final que hemos creado para realizar el aprendizaje ha sido el siguiente:



22 Proceso del aprendizaje supervisado

4.3.3.2 Preprocesado de textos

El fin de realizar el pre-procesado es el de obtener texto lo más limpio posible y el de reducir el espacio de características. Este preprocesado es equivalente al de la sección 4.3.2.2 con la diferencia de que en la sección anterior queremos obtener las palabras para realizar una comparación y en este caso para generar una bolsa con todas ellas.

Los dos pasos principales que hemos realizado son:

1. Limpieza del texto: este paso se realizará en dos partes, la primera antes de la reducción del espacio de características y la segunda tras ella (Sección 4.3.2.2 Puntos 1 y 4)

2. Reducción del espacio de características: El modelo del lenguaje de *Twitter* tiene unas características únicas que transformamos para reducir el espacio de características. (Sección 4.3.2.2 Puntos 1 y 4)

4.3.3.3 Extracción de características (Matriz de términos)

Una vez que se ha realizado el preprocesado comienza el proceso para construir la matriz de términos, es decir, de generar la bolsa de palabras con las asignaciones de pesos a cada una de las palabras de la bolsa. Para construir esta matriz hemos utilizado una biblioteca de Python llamada *Sklearn*. En este caso, que pertenece a esta biblioteca, hemos utilizado la clase TfidfVectorizer que realizará el proceso de vectorización. Lo que

hace está clase es transformar el conjunto que preprocesaríamos, como hemos explicado en el punto 4.3.3.2, y mediante una serie de parámetros de configuración generaría la matriz de términos.

Los parámetros que se han seleccionado para configuración han sido:

- *tokenizer*= parámetro que permite pasar como argumento una función que realice la *tokenizacion* y el *stemming*.
- *min_df*= umbral que define el número de apariciones mínimo o frecuencia de una palabra.
- *max_df* = umbral que define el número de apariciones máximo o frecuencia de una palabra.
- *sublinear_tf*= parámetro que permite sustituir en la ecuación del cálculo, el parámetro tf por 1+log(tf)
- *use_idf* = parámetro que permite decidir si se usa o no la frecuencia inversa para el cálculo.
- *stop_words*= parámetro que permite definir el conjunto de palabras de parada utilizado.
- *analyzer*= parámetro que con la opción '*word*' permite el uso del parámetro anterior denominado *tokenizer*.
- *ngram_range*= parámetro para definir el tamaño de los *n-gramas*.
- *max_features*= parámetro para definir el número máximo de características usadas.

Estos parámetros son suficientes para poder manejar la matriz resultante y permitir una parametrización en busca de generar el mejor vector posible para la clasificación posterior.

4.3.3.4 Creación de clasificadores

Al igual que para la creación de la matriz de términos, vamos a utilizar las clases asociadas a cada algoritmo de los propuestos, de la biblioteca Sklearn. Para generar los modelos de los distintos algoritmos se utilizada la matriz de términos generada en el paso anterior. Para cada algoritmo se generará un modelo con el que se realizarán las diversas predicciones. En el apartado 5 se expondrá cada una de las opciones de generación de modelos que hemos introducido, así como las distintas formas de predicción. Los parámetros que se han seleccionado de cada algoritmo son:

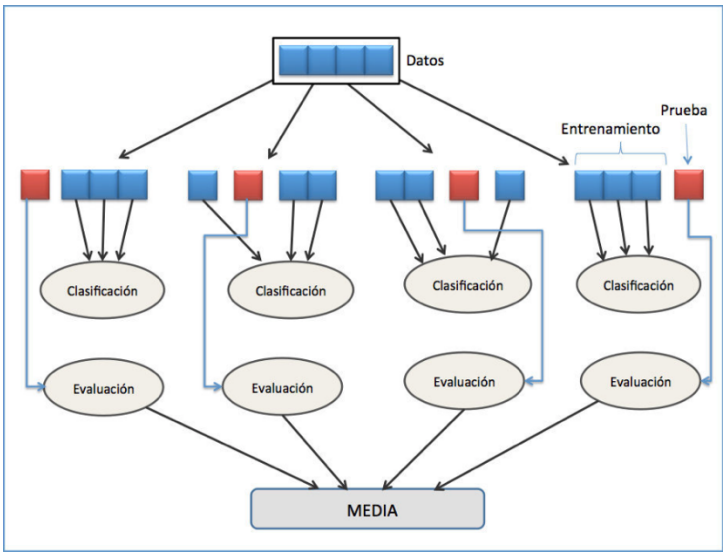
- Naïve Bayes:
  - *alpha*: parámetro para suavizar el ruido.
  - *fit_prior*: parámetro para definir si se aprende las probabilidades previas de la clase o no.
- SVM:
  - *dual*: parámetro que define el tipo de algoritmo usado.
  - C: parámetro de penalización del término del error

- KNeighbors:
    - *n_neighbors*: parámetro para definir el número de vecinos.
    - *algorithm*: define el tipo de algoritmo usado para el computo de vecinos más cercanos. Usamos '*brute*' ya que es el que mejor se adapta a nuestro análisis.
    - *metric*: se ha utilizado la métrica '*cosine*' ya que se ha determinado que se obtiene mejores resultados que el resto en el caso de la clasificación de textos [Huang2008].

- MLP:
    - *hidden_layer_sizes*: parámetro para definir el número de capas ocultas y de neuronas en esa capa
    - *max_iter*: número máximo de iteraciones del algoritmo.
    - *alpha*: parámetro para regulación de la penalización.
    - *learning_rate*: parámetro que define el tipo de tasa de aprendizaje para la actualización de los pesos, se podrá elegir entre sus 3 tasas: *'constant'*, '*invscaling'*, '*adaptive'*.
    - *learning_rate_init*: parámetro para definir la tasa inicial de aprendizaje.

Los métodos de predicción que vamos a utilizar son:

- Evaluación sobre el *test data*: con este método lo que haremos será predecir sobre los datos de test que hemos obtenido en la división inicial del *dataset,* con el modelo generado sobre el *train data*. Se pude dar el caso de utilizar el test data original o de usar uno distinto que habría que volver a cargar.
- *Hold Out*: en este método solo utiliza el *train data* que hemos definido inicialmente. Lo que se hace es realizar una nueva división sobre el *train data* generando otro *train data* que usará para crear un modelo y otro *test data* para predecir sobre él.
- Validación cruzada: este método, al igual que el anterior, solo utiliza el *train data* definido inicialmente. Lo que hace es dividir el *train data* en varias partes que combina entre sí de formas distintas. Genera las diversas combinaciones de las partes generadas para crear un nuevo *train data* con el que crea un nuevo modelo dejando siempre una de las partes como *test data* para predecir.



23 Validación Cruzada

Para realizar las predicciones, se pueden utilizar distintas métricas de rendimiento, en nuestro caso las tres métricas que vamos a usar son el área bajo la curva (AUC) y la *accuracy* (tasa de acierto).

- *Accuracy*: es el porcentaje de elementos clasificados de forma correcta.
- AUC: representa la probabilidad de que el clasificador proporcione a los ejemplos del conjunto de prueba *scores* correctamente ordenados que permitirían generar clasificadores con precisión 1 (con tasa nula de falsos positivos y tasa de ciertos positivos igual a uno).

5 Desarrollo de un entorno para el análisis de sentimiento

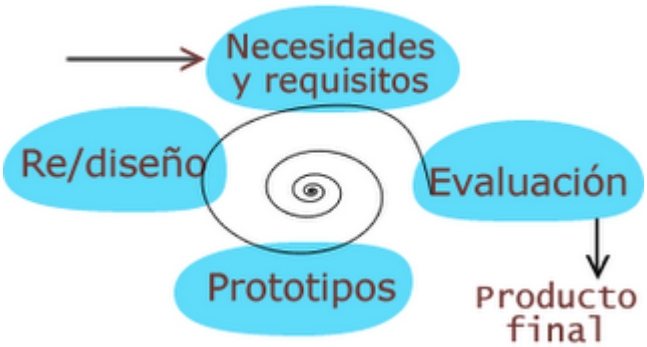
Para ampliar el alcance del proyecto se decidió realizar una interfaz que agrupase toda la funcionalidad de la parte de la arquitectura de procesamiento y acceso a los datos. Partiendo de esto se ha creado una interfaz desde la cual se puede configurar los dos procesos en los cuales hemos centrado el proyecto como son el etiquetado basado en diccionarios y el aprendizaje supervisado. Estas dos funcionalidades son:

1. Etiquetado basado en diccionarios: para esta funcionalidad se ha creado una pestaña en la interfaz en la cual se podrán ajustar ciertos parámetros de entrada.

2. Clasificación mediante aprendizaje supervisado: para este desarrollo se han creado 4 pestañas en cada una de las cuales se realizará una parte de la preparación del clasificador. Serán una para preprocesado, una para parametrización de la matriz de términos, una para selección de modelos y otra para predicciones.

5.1 Metodología de trabajo

Para el diseño de la interfaz se ha utilizado una metodología de diseño interactiva. Esta metodología está basada en un ciclo de vida en el que iteración a iteración se va perfeccionando el diseño. El primer paso, de necesidades y requisitos, llegado a este punto ya lo conocíamos, ya que se ha definido en apartados anteriores describiendo todo aquello que iba a estar incluido en esta herramienta. Lo que necesitábamos principalmente era que realizase dos funcionalidades, el etiquetado y la clasificación con sus respectivos pasos intermedios.



24 Proceso de diseño y desarrollo de interfaces

Conociendo las necesidades, se propuso un diseño inicial y un prototipo para una primera evaluación. Partiendo de esto, el método de evaluación empleado fue utilizando como usuario al tutor del proyecto. Se realizaron reuniones semanales en las que el tutor como usuario realizaba una evaluación del proyecto en la que se contemplaban posibles mejoras o cambios. Estos cambios se implementaban durante la semana y se realizaba una nueva evaluación. Este proceso continuó hasta que se obtuvo la herramienta final.

5.2 Tecnología utilizada

Ante necesidad de seleccionar las herramientas de desarrollo, se consideraron las tareas a realizar y el entorno *Big Data*. Las alternativas que observamos más factibles para nuestro caso fueron las de realizar el código en Scala o Python. Ambas habían sido usadas a lo largo del Máster en alguna asignatura. En el caso de Python su uso era más reciente y además este lenguaje ya se conocía en gran medida antes de comenzar el Máster.

Lo primero que teníamos que tener en cuenta era que había que realizar una interfaz para la herramienta, para poder ser capaz de configurar todo el proceso del etiquetado y clasificación. Ante el conocimiento de la gran cantidad de bibliotecas que dispone Python tanto para clasificación, como para trabajo con texto, como para crear interfaces, además de su uso más reciente, nos decantamos por Python. También Python dispone de la posibilidad de crear entornos virtuales lo que nos permite instalar en un directorio todo lo necesario para ejecutar la herramienta y trasladarlo de un ordenador a otro sin mucha dificultad.

Las principales bibliotecas de *Python* que se han utilizado han sido:

- *Sklearn (Scikit Learn):* para la obtención de los algoritmos de clasificación y de vectorización.
- *Numpy*: para el guardado de ficheros de datos csv.
- *PyQT5:* para la creación de la interfaz.
- *Nltk*: para la obtención de herramientas de lenguaje natural como el *stemming* y la *tokenización*.
- *Pyspark*: para el procesamiento paralelo de ciertos aspectos del preprocesado y el etiquetado.

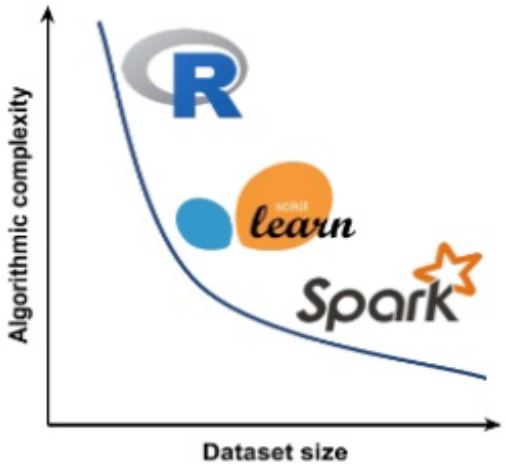
Para realizar la clasificación debíamos decidir si utilizar los algoritmos de clasificación de *Sklearn* o los de *Spark* de *Mllib/Ml*. Finalmente teniendo en cuenta las siguientes ventajas nos decantamos por *Scikit.*

Ventajas de Mllib/Ml:

- Es más escalable en un entorno *Big Data*, ya que explota las propiedades de paralelismo y distribución de *Spark*, además de estar programada en *Scala*, el lenguaje nativo de *Spark.*

Ventajas de *Sklearn*:

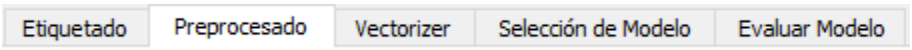
- No es tan *escalable* como *Mllib/Ml* pero es capaz de trabajar en el ámbito del Big Data. Funciona muy bien cuando los datos caben en la RAM, a partir de ahí sería más eficiente *Mlib/Ml.*
- Permite una mayor cantidad de opciones de parametrización de los distintitos algoritmos (sobre todo en la vectorización).
- Mayor diversidad de algoritmos.
- Mayor facilidad de programación.
- Los 4 algoritmos que hemos escogido soportan clasificación de 3 categorías.



25 Relación en velocidad de computación de herramientas de clasificación

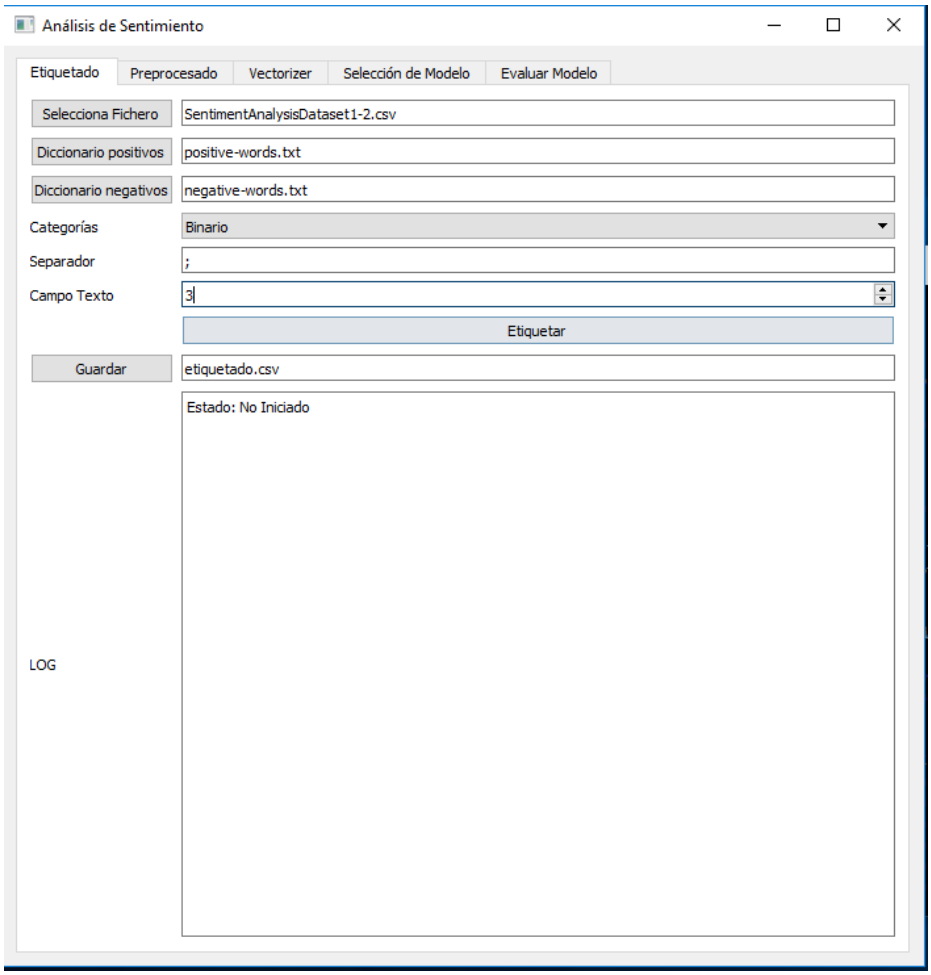
5.3 Desarrollo de la interfaz

La interfaz desarrollada dispondrá de cinco pestañas cada una de las cuales albergará una parte de la funcionalidad. Estas serán:



26 Menú de la aplicación

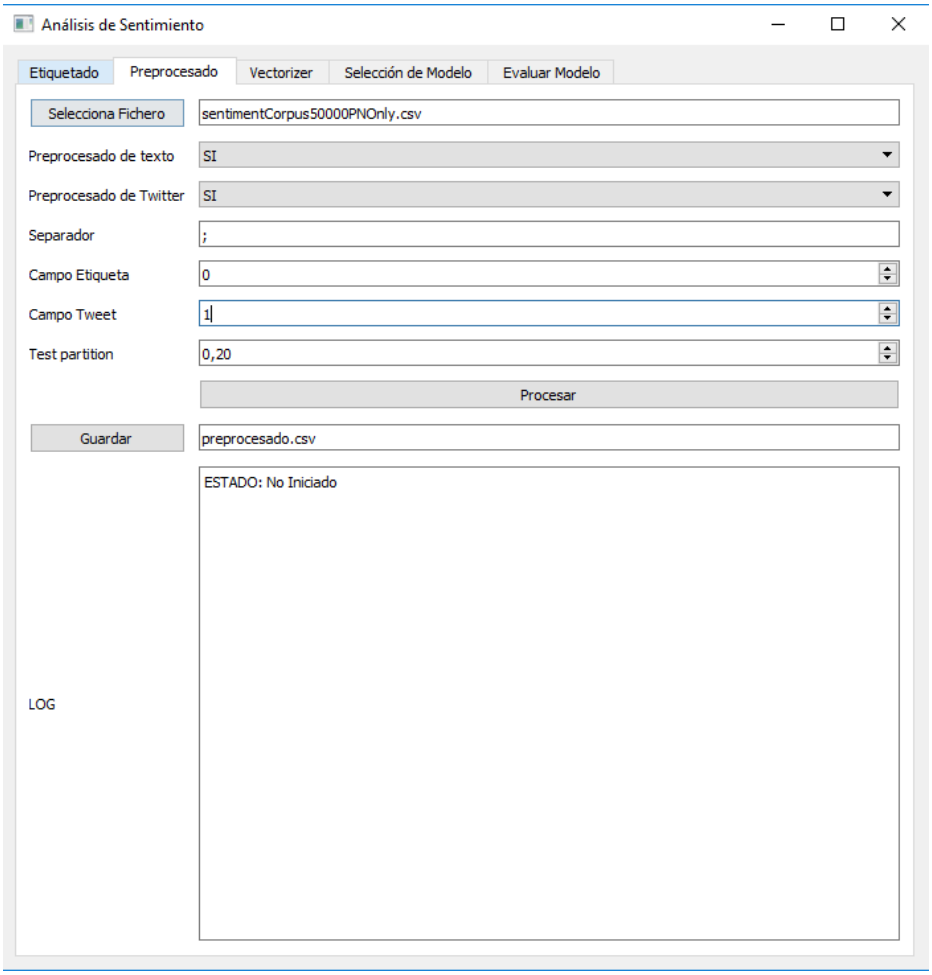
1. Etiquetado: en este apartado permitimos realizar un etiquetado de texto tanto en categorías binarias como ternarias, pudiendo elegir diccionarios de búsqueda propios. Las partes parametrizables que tiene la pestaña son:



27 Pestaña de etiquetado de la interfaz

a. Selección de fichero: permite elegir el fichero de entrada para realizar el etiquetado.
b. Diccionario positivo: permite elegir el fichero de entrada con el diccionario de palabras que se consideran de grado positivo.
c. Diccionario negativo: permite elegir el fichero de entrada con el diccionario de palabras que se consideran de grado negativo.
d. Categorías: permite decidir si se van a utilizar solo aquellos *tweets* etiquetados como positivos y negativos o si se quieren también aquellos de sentimiento neutro.
e. Separador: en el caso de que en el fichero hubiera más campos además del *tweet* permite elegir el separador que utiliza el fichero.
f. Campo texto: campo que defina la posición en la cual se encuentra el *tweet* si en el fichero hubiera más información además del texto de este. Si solo hubiera un campo en el fichero se dejaría el 0.
g. Etiquetar: para ejecutar el proceso de etiquetado.
h. Guardar: para guardar los resultados generados por el etiquetado. En el directorio de ejecución del proceso debe de haber un directorio llamado etiquetados. En él, se guardará un fichero con el nombre que le demos en el campo de texto.
i. LOG: campo en el cual se mostrarán resultados o estados de ejecución.

2. Preprocesado: este apartado permite realizar un procesado de texto tal como se ha explicado las secciones anteriores. Las partes parametrizables que tiene la pestaña son:

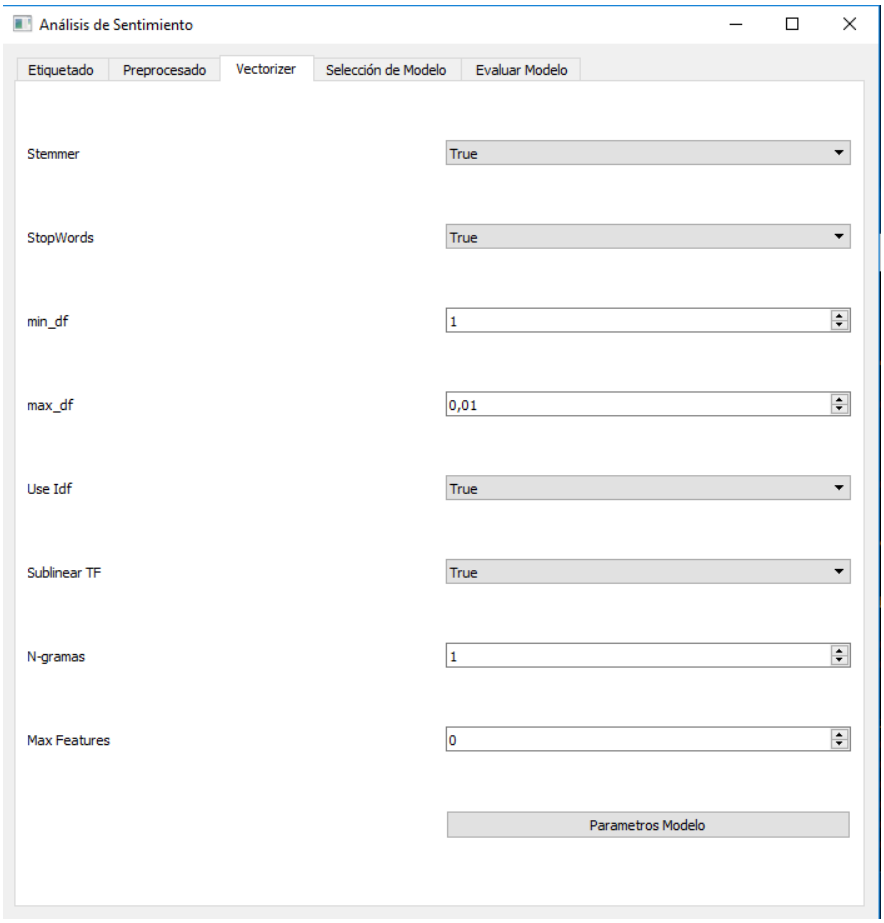


28 Pestaña de preprocesado de la interfaz

   a. Selección de fichero: permite elegir el fichero de entrada para realizar el etiquetado.
   b. Procesado de texto: permite decidir si se realiza la limpieza de texto, correspondiente al apartado 4.3.3.2 punto primero.
   c. Procesado de *Twitter*: permite decidir si se realiza el procesado elementos típicos de *Twitter* para reducir el espacio de características, correspondiente al apartado 4.3.3.2 punto segundo.
   d. Separador: permite elegir el separador que utiliza el fichero.
   e. Campo etiqueta: campo que defina la posición en la cual se encuentra la etiqueta que define la categoría del *tweet*. Si solo hubiera un campo en el fichero se dejaría el 0.
   f. Campo tweet: campo que defina la posición en la cual se encuentra el tweet. Si solo hubiera un campo en el fichero se dejaría el 0.
   g. *Test partition*: permite decir el tamaño que se dará a la partición que dejará para usar como test.
   h. Procesar: botón para ejecutar el procesado.

i. Guardar: para guardar los resultados generados por el procesado. En el directorio de ejecución del proceso debe de haber un directorio llamado preprocesados. En él, se guardará un fichero con el nombre que le demos en el campo de texto.
j. *LOG*: campo en el cual se mostrarán resultados o estados de ejecución.

3. Vectorizer: este apartado permite realizar una parametrización de cómo se va a generar el conjunto de vectores. Correspondiente al apartado 4.3.3.3. Los parámetros que configuramos son:



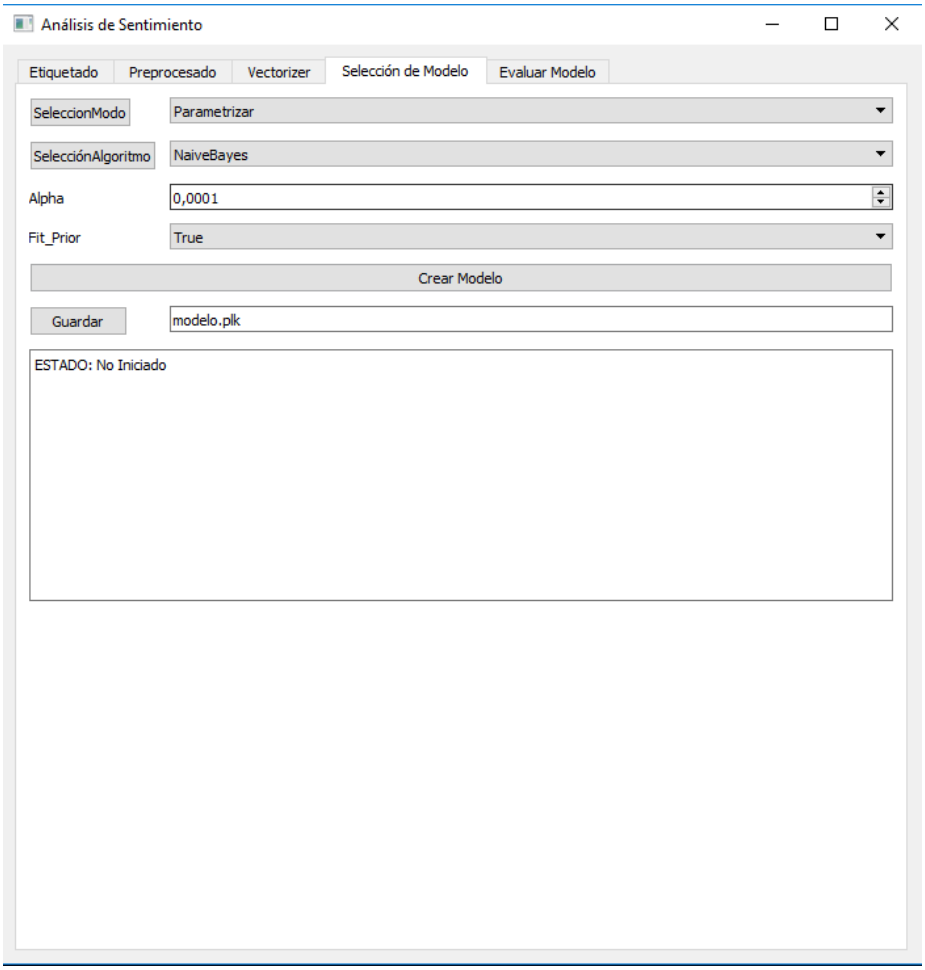
29 Pestaña de parametrización del vectorizer

a. *Stemmer*: para decidir si se usa o no la transformación en lexemas. *(True/False).*
b. *StopWords*: Para decidir si se eliminan o no el conjunto de *StopWords.(True/False).*
c. *min_df*: un entero para definir el número mínimo de apariciones de una palabra.
d. *max_df*: un *float* para definir el porcentaje máximo de textos en los que aparece una palabra.
e. *Use Idf*: para decidir si se usa la frecuencia inversa para el cálculo *(True/False).*
f. *Sublinear Tf*: para decidir si se usa el valor 1+log(tf) en vez de tf en el cálculo *(True/False).*
g. *N-gramas*: entero para definir el tamaño de los *n-gramas*.

h. *Max Features*: entero para definir el número máximo de características, a 0 si no se quiere usar un máximo.

4. Selección de modelo: para la selección de modelos hemos introducido 3 funcionalidades, cada una de ellas dentro de la misma pestaña nos permitirá modificar unos parámetros u otros. Estas opciones son la parametrización, la optimización y el importado. Permitirá elegir entre los algoritmos definidos en el proyecto y sus parámetros (apartado 4.3.3.4). Cada opción la hemos definido:

    a. Parametrizar modelo: esta función permite crear un modelo eligiendo el algoritmo y sus parámetros. Sus opciones serán:

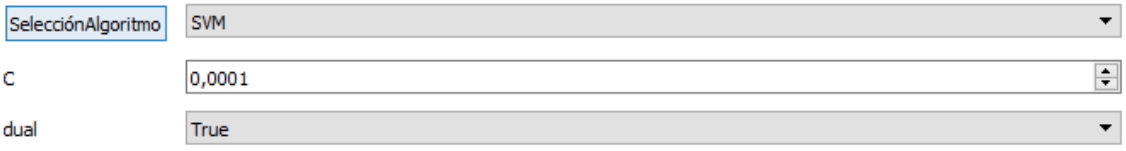


30 Pestaña de Selección de Modelos: Parametrización

        i. Selección modo: parte común, con la que podemos cambiar entre las 3 funcionalidades.
        ii. Selección de algoritmo: parte de la opción parametrización en la que podemos cambiar el algoritmo entre los 4 disponibles que nos ofrecerá sus parámetros configurables. Para cada caso tenemos:
            1. Naïve Bayes: definimos el parámetro *alpha* como un *float* y el para metro *Fir_Prior* como un *Boolean*.

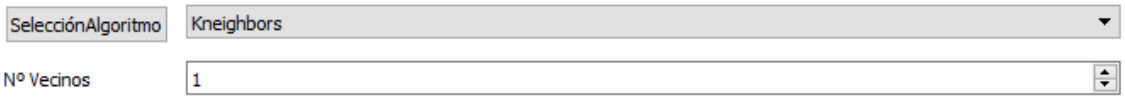
31 Parametrización NaïveBayes

2. SVM: definimos el parámetro C como un *float* y el parámetro *dual* como un *Boolean*.



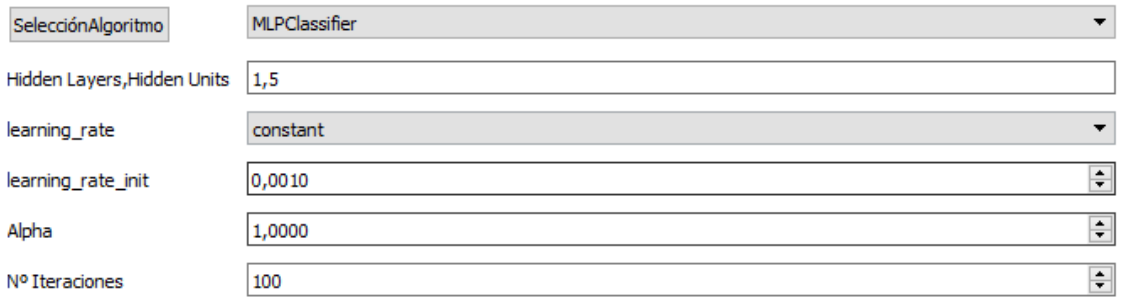
32 Parametrización SVM

3. KNeightbors: definimos el parámetro número de vecinos como un entero.



33 Parametrización KNeightbors

1. MLP: definimos el *alpha* como un *float* y el número de iteraciones como otro entero. También utilizamos el *learning_rate* con sus tres opciones disponibles, así como la *learning_rate_init* que es un *float*. Finalmente, el parámetro más complejo es el que define la capa oculta en el cual los valores van separados por comas. El primer número define el número de capas ocultas y posteriormente se añadirán tantos tamaños como capas ocultas definimos. Por ejemplo, en la imagen inferior definimos el caso para una capa oculta con 5 neuronas.

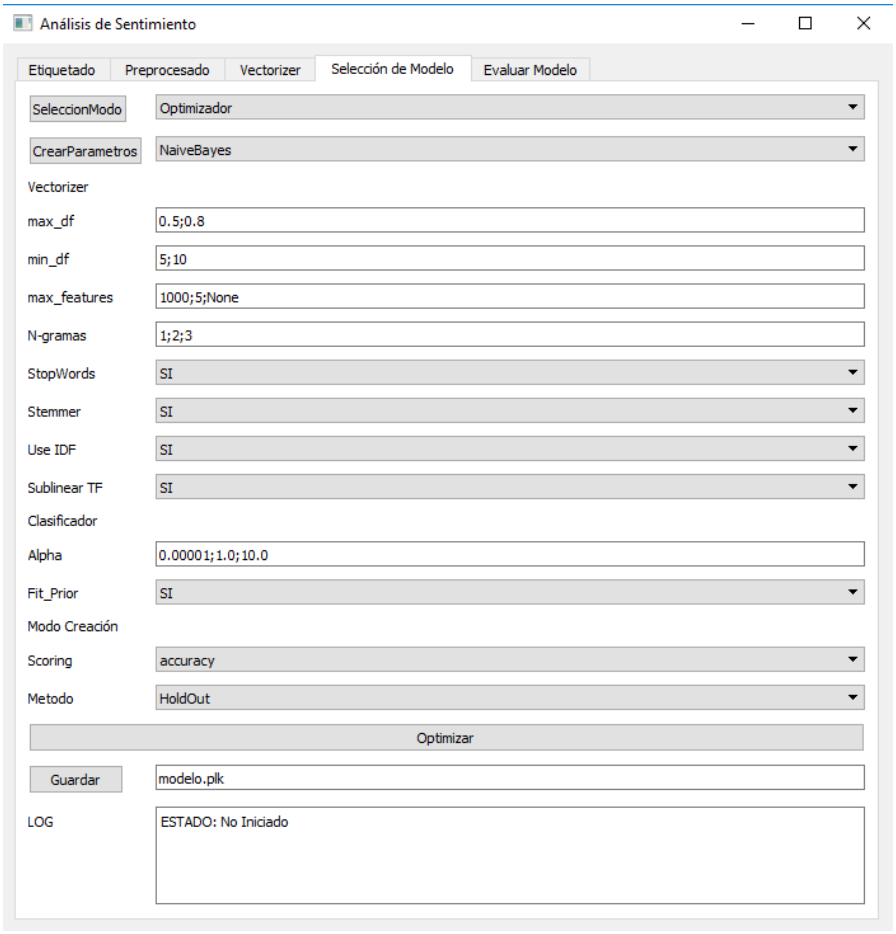


34 Parametrización MLPClassifier

iii. Crear Modelo: una vez parametrizado pulsando esta opción crearemos el modelo.
iv. Guardar: permite guardar el modelo creado con el nombre que le demos.

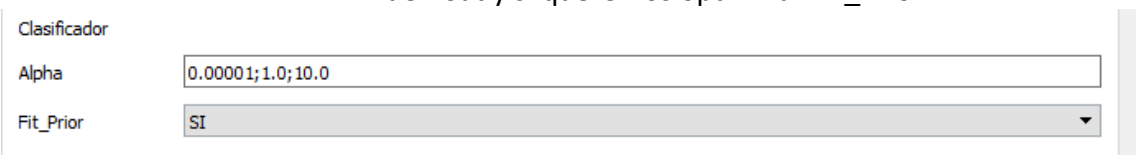
v. LOG: campo en el cual se mostrarán resultados o estados de ejecución.

b. Optimizar modelo: esta función permite crear un modelo óptimo eligiendo el algoritmo, sus posibles parámetros, el método de optimización y la métrica para la optimización. Cada lista de parámetros usará el separador ";". Sus opciones serán:



35 Pestaña de Selección de Modelos: Optimización

i. Selección modo: parte común, con la que podemos cambiar entre las 3 funcionalidades.
ii. Selección de algoritmo: parte de la opción parametrización en la que podemos cambiar el algoritmo entre los 4 disponibles que nos ofrecerá sus parámetros configurables. Para cada caso tenemos:
    1. Naïve Bayes: definimos el parámetro alpha como una lista de float y si queremos optimizar Fir_Prior.

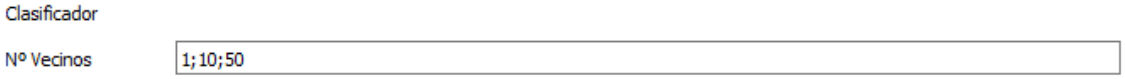


36 Optimización Naïve Bayes

2. SVM: definimos el parámetro C como una lista de float y si queremos optimizar el parámetro dual.

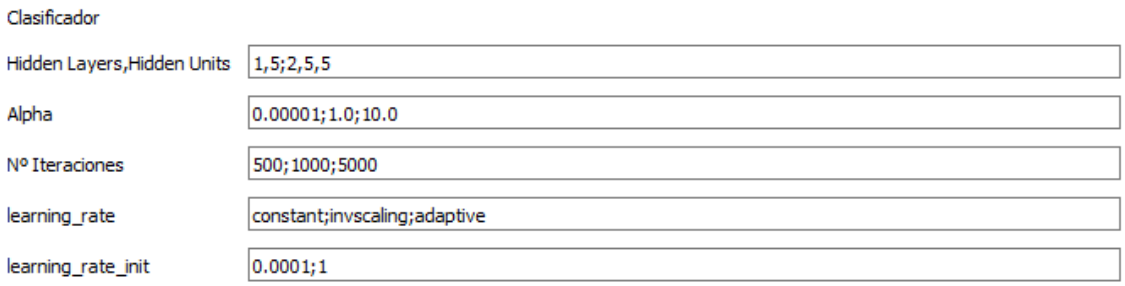
37 Optimización SVM

3. KNeightbors: definimos el parámetro número de vecinos como una lista de enteros.



38 Optimización Kneightbors

4. MLP: definimos el alpha como una lista de float y el número de iteraciones como una lista de enteros. También utilizamos el *learning_rate,* donde podemos introducir sus tres opciones disponibles, así como la *learning_rate_init* que es una lista de *float*. Finalmente, el parámetro más complejo es el que define la capa oculta en el cual cada conjunto de valores va separado por comas. El primer número define el número de capas ocultas y posteriormente se añadirán tantos tamaños como capas ocultas definimos. Por ejemplo, en la imagen inferior definimos dos casos, uno con una capa con 5 neuronas y otro con dos capas y 5 neuronas en cada capa.



39 Optimización MLPClassifier

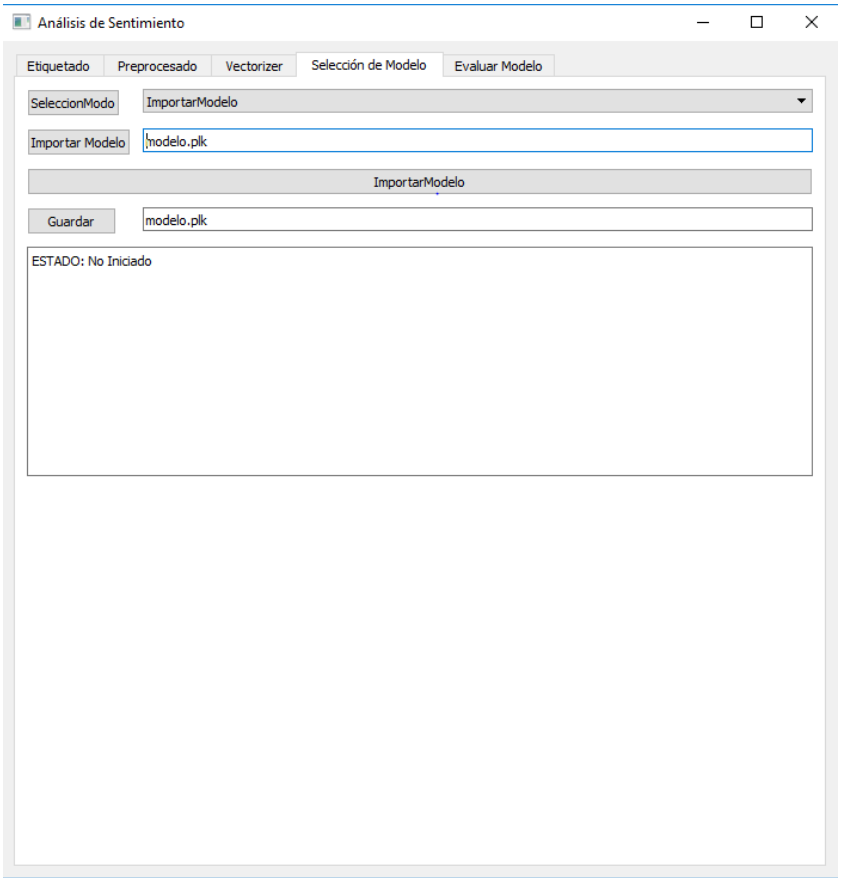
iii. Parámetros del vectorizer comunes a los 4 algoritmos:
   1. max_df: listado de floats.
   2. min_df: listado de enteros.
   3. max_features: listado de enteros, para usar la opción "None" podemos usar el valor 0 o escribir tal cual esa opción.
   4. *n-gramas*: listado de enteros.
   5. StopWords: para decidir si usamos o no esta optimización.
   6. Stemmer: para decidir si usamos o no esta optimización.

7. Use IDF: para decidir si usamos o no esta optimización.
8. Sublinear TF: para decidir si usamos o no esta optimización.

iv. Parámetros de Modo de creación del optimizador:
    a. *Scoring*: permite decidir si crear la optimización usando los valores de tasa de error (*accuracy*) o el área bajo la curva (roc_auc).
    b. Método: permite decidir si crear la optimización usando validación cruzada o *Hold Out*.

v. Optimizar: una vez parametrizado pulsando esta opción crearemos el modelo óptimo.
vi. Guardar: permite guardar el modelo creado con el nombre que le demos.
vii. *LOG*: campo en el cual se mostrarán resultados o estados de ejecución.

c. Importar modelo: esta opción nos permite cargar un modelo guardado, que hubiera sido creado en los dos casos anteriores. Sus opciones son:

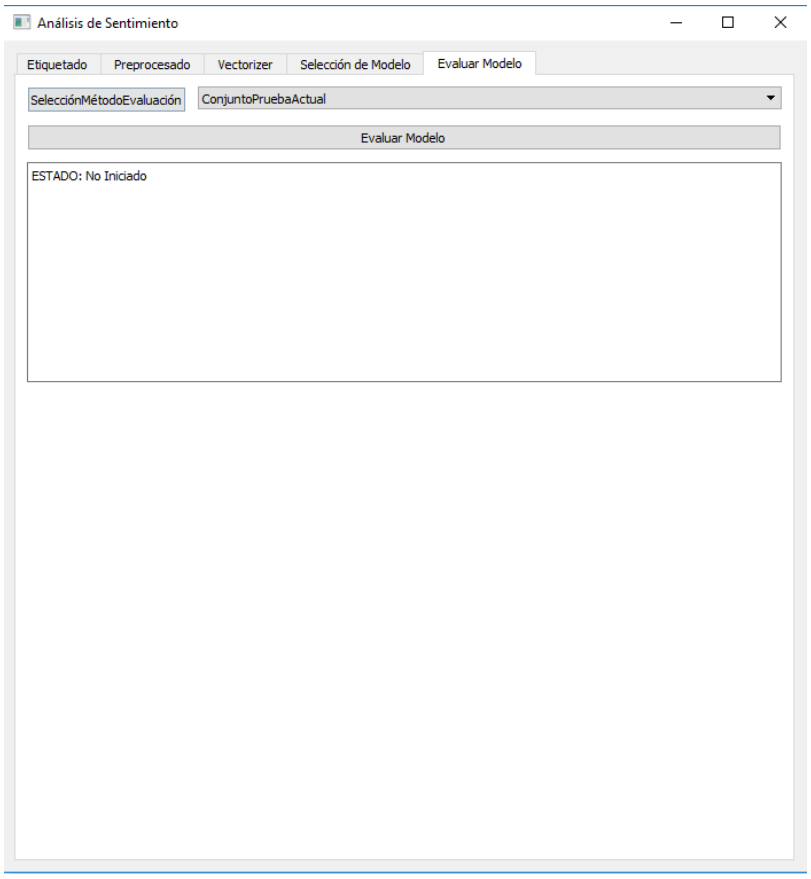


40 Importar modelo

i. Selección modo: parte común, con la que podemos cambiar entre las 3 funcionalidades.

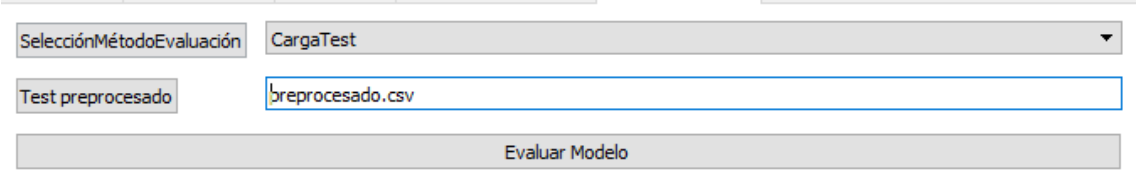
ii. Importar Modelo: Selección de la ruta del algoritmo.
iii. Importar Modelo: una vez seleccionada la ruta, pulsando esta opción se cargará el modelo.
iv. Guardar: permite guardar el modelo importado de nuevo con el nombre que le demos.
v. *LOG*: campo en el cual se mostrarán resultados o estados de ejecución.

5. Evaluación de Modelo: esta pestaña permite realizar la evaluación de los modelos de cuatro formas posibles. Como métricas mostrará la tasa de error, la varianza y el área bajo la curva, esta última si fuera posible. Las cuatro formas de evaluación son:



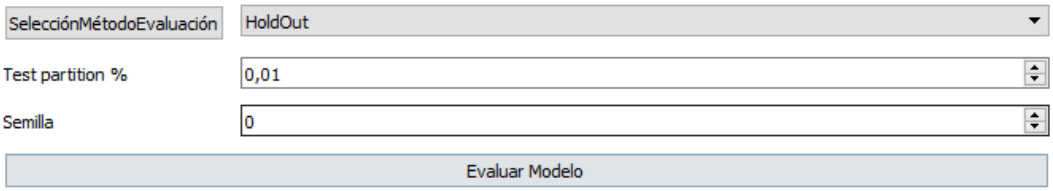
41 Pestaña Evaluar Modelo

a. ConjuntoPruebaActual: esta es la opción más simple, solo dispone de un botón para realizar la evaluación del modelo generado en el paso anterior sobre el conjunto de *test* preprocesado.
b. *CargaTest*: esta opción permite la selección de un conjunto de *test* distinto al ya preprocesado para la evaluación del modelo.



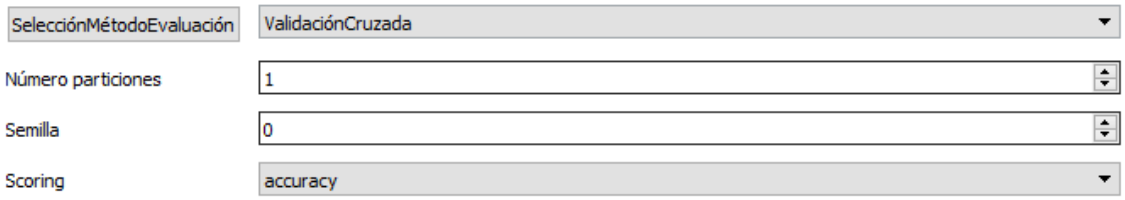
42 Evaluación por el método CargaTest

c. *HoldOut*: esta opción permite utilizar la técnica de *hold out* sobre el *train* data para generar una evaluación. No disponible para la opción importar modelo ya que no podemos obtener los parámetros con los que se creó ese modelo del fichero importado. Permite configurar:
   i. *Test partition*: para selección el tamaño de la partición de test:
   ii. Semilla: para seleccionar un valor y poder obtener resultados distintos.



43 Evaluación por el método Hold Out

d. Validación Cruzada: esta opción permite utilizar la técnica de validación cruzada sobre el *train* data para generar una evaluación. No disponible para la opción importar modelo ya que no podemos obtener los parámetros con los que se creó ese modelo del fichero importado. Los parámetros configurables son:



44 Evaluación por el método validación cruzada

   i. Número de particiones: para decidir cuantas evaluaciones queremos realizar.
   ii. Semilla: para seleccionar un valor y poder obtener resultados distintos.
   iii. Scoring: al tener que generar nuevos modelos y ser un proceso más costoso, en este caso podemos decir si usar la tasa de error (accuracy), la varianza o el área bajo la curva (roc_auc) como métrica.

6 Pruebas experimentales

6.1 Descripción de los experimentos

Se van a realizar distintas pruebas sobre los algoritmos propuestos. Para estas pruebas se van a utilizar dos conjuntos que se han obtenido en internet y que se explicarán de forma más detallada posteriormente. Las razones que nos llevan a utilizar *dataset* externos en vez de los recopilados por la herramienta son dos:

- La primera es evaluar la propuesta realizada en el proyecto de etiquetado, frente a otras propuestas. Además, también queremos observar la diferencia de resultados sobre un etiquetado binario contra uno ternario.
- La segunda es que nuestras capacidades de descarga con Kafka han sido limitadas. *Twitter* solo nos permite descargar una pequeña cantidad de *tweets*, teniendo conectada durante una semana Kafka en un portátil no dedicado exclusivamente a esa tarea, se recopilaron 40000 tweets. De los cuales una gran mayoría eran repetidos al ser RTs. Por ello solo teniendo unos 10000 tweets lo consideramos insuficiente.

Para cada uno de los cuatro algoritmos, se van a generar cuatro casos:

- Caso 1: aprendizaje supervisado utilizando el etiquetado binario propuesto en este proyecto.
- Caso 2: aprendizaje supervisado utilizando el etiquetado ternario (positivo-negativo-neutro) propuesto en este proyecto.
- Caso 3: aprendizaje supervisado para el etiquetado propuesto en el *dataset* que usaremos de la herramienta Sentiment140 en ambos casos. Será clasificación binaria.
- Caso 4: se usará el modelo binario generado en el caso 1 frente al etiquetado usado en el caso 3.

Estos cuatro casos van a realizarse sobre ambos *dataset*. Sin embargo, sobre el primer *dataset* la evaluación será distinta que sobre el segundo. Las evaluaciones que se van a realizar sobre el primero serán:

1. El primer paso será optimizar los parámetros mediante las funcionalidades de optimización. Los parámetros seleccionados se utilizarán para crear los modelos finales que se evaluaran con las distintas métricas.

Una vez obtenido los parámetros óptimos se utilizarán para realizar evaluación de los modelos:

2. Evaluación sobre el *test* cargado
3. *Hold out*: con test 20% y semilla 3, sobre el conjunto de entrenamiento.
4. Validación cruzada: 3 particiones y semilla 0, sobre el conjunto de entrenamiento.

La evaluación con Hold Out y Validación cruzada sobre el conjunto de entrenamiento se ha incluido para comprobar si se produce sobreajuste debido al proceso de optimización. Podemos adelantar, en base a los resultados obtenidos, que no es así.

El caso 4 es una excepción, ya que solo se realizará una evaluación, en vez de las 4 de los 3 primeros. Para cada uno de los algoritmos se van a optimizar los parámetros del *vectorizer* y los del algoritmo de clasificación en cada caso.

La optimización es muy costosa por lo cual para cada parámetro del *vectorizer* no se introducirán más de dos valores. Y la optimización que se va a realizar va a ser con el modo *hold out* y con la métrica *accuracy*.

El *vectorizer* es común en todos los casos, los parámetros con los valores que se van a optimizar son:

- *max_df*: 0.5, 0.8. Este parámetro utiliza valores float, que equivalen a porcentajes. Para no exceder la carga de computación se han elegido dos valores iguales o superiores al 50%.
- *min_df*: 5,10. Este parámetro utilizan valores enteros, que equivalen a número de ocurrencias. Para no exceder la carga de computación se han elegido dos valores entre 5 y 10 ocurrencias, que como se verá posteriormente en la mayoría de los casos es 5 el valor adecuado lo que indica que los valores elegidos son adecuados.
- *ngram_range*: (1, 1), (1,2). Sin n-gramas o con n-gramas de grupos de dos palabras.
- *stop_words*: ['i', 'me', ……..'ve', 'y', 'nt'] , None. Eliminando el conjunto de *stopwords* o sin eliminarle.
- *tokenizer*: stemming_tokenizer , None. Utilizando nuestra función de *stemming* con tokenización o sin ella.
- *use_idf*: True, False. Usando la opción de frecuencia inversa o sin ella.
- *sublinear_tf*: True, False. Usando la opción de frecuencia sublinear o sin ella.
- *max_features*: no se optimiza ya que funcionará mejor con el mayor número de características ya que son el resto de los parámetros los que eliminan aquellas palabras que crearían características innecesarias.

Esto se realizará para el primer *dataset*, sin embargo, para el segundo será algo diferente. En este *dataset* aplicaremos los cuatro casos propuestos, pero no usaremos los cuatro algoritmos disponibles sino solamente aquel que en cada caso ofreció mejores resultados. Como parámetros usaremos los que se obtuvieron en la optimización de cada algoritmo que seleccionemos.

6.2 Experimentos primer *dataset*

El primer *dataset [dataset]* es de Sander's Lab donde se recopilan 1.5 millones de *tweets* que han sido etiquetados en categorías positivas o negativas por una herramienta de etiquetado denominada Sentiment140 [Sentiment140]. Los *tweets* son aleatorios, es decir, que no son de ningún tema en especial. Además, corresponden al tamaño antiguo de mensaje de *Twitter* por lo que tienen como máximo 140 caracteres. Sentiment140

es una herramienta gratuita que permite realizar el análisis de sentimiento en *Twitter*. Esta herramienta fue creada por Alec Go, Richa Bhayani, and Lei Huang, Graduados en Informática por la Universidad de Standford, y es una de las más populares para el análisis de sentimiento. Las pruebas se van a realizar sobre una parte del *dataset* de 50000 *tweets.* El tamaño de los conjuntos será 80% para *train* y 20% para *test*.

6.2.1 Selección de modelos

1. Algoritmo Naïve Bayes:

   Los valores del algoritmo a optimizar son:
   alpha: 0.0001, 1.0 , 10.0
   fit_prior: True, False

- Etiquetado manual con descarte de *tweets* neutros

Best score	0.9608
Best parameters set	
classifier	
alpha	1.0
fit_prior	False
vectorizer	
max_df	0.5
min_df	5
ngram_range	(1, 1)
stop_words	True
tokenizer	True
use_idf	False
sublinear_tf	True

Tabla 1 Best parameters prueba 1.1

- Prueba test:
  - Accuracy: 0.9601±0.002
  - AUC: 0.9935

- Validación cruzada (3):
  - Accuracy: 0.9571±0.0014
  - AUC: 0.9911

- Hold Out(20%):
  - Accuracy: 0.9518±0.003
  - AUC: 0.9900

- Etiquetado manual sin descarte de *tweets* neutros

Best score	0.7724
Best parameters set	
classifier	
alpha	0.0001
fit_prior	True
vectorizer	
max_df	0.5
min_df	5
ngram_range	(1, 1)
stop_words	True
tokenizer	False
use_idf	False
sublinear_tf	True

Tabla 2 Best parameters prueba 2.1

  - Prueba test:
    - Accuracy: 0.7705±0.0041
  - Validación cruzada (3):
    - Accuracy: 0.7691±0.0025
  - Hold Out(20%):
    - Accuracy: 0.7659±0.0047

- Etiquetado sentiment140 positivos negativos

Best score	0.7526
Best parameters set	
classifier	
alpha	1.0
fit_prior	True
vectorizer	
max_df	0.5
min_df	5
ngram_range	(1, 2)
stop_words	False
tokenizer	False
use_idf	False
sublinear_tf	True

Tabla 3 Best parameters prueba 3.1

- Prueba test:
  - Accuracy: 0.7559±0.0042
  - AUC: 0.8382
  - Error ± Desviación estandar: 0.2441

- Validación cruzada (3):
  - Accuracy: 0.7588±0.0092
  - AUC: 0.8368

- Hold Out(20%):
  - Accuracy: 0.7621±0.0047
  - AUC: 0.8465

2. Algoritmo SVM:

Los valores del algoritmo a optimizar son:
C: 0.0001, 1.0, 10.0
dual: True, False

- Etiquetado manual con descarte de *tweets* neutros

Best score	0.964
Best parameters set	
classifier	
C	1.0
dual	True
vectorizer	
max_df	0.5
min_df	5
ngram_range	(1, 2)
stop_words	True
tokenizer	True
use_idf	True
sublinear_tf	True

Tabla 4 Best parameters prueba 1.2

- Prueba test:
  - Accuracy: 0.9733±0.0022
  - AUC: 0.9964

- Validación cruzada (3):
  - Accuracy: 0.9734±0.0021
  - AUC: 0.9953

- Hold Out(20%):
  - Accuracy: 0.9640±0.0028
  - AUC: 0.9955

- Etiquetado manual sin descarte de *tweets* neutros

Best score	0.9001
Best parameters set	
classifier	
C	1.0
dual	True
vectorizer	
max_df	0.5
min_df	5
ngram_range	(1, 1)
stop_words	True
tokenizer	False
use_idf	True
sublinear_tf	True

Tabla 5 Best parameters prueba 2.2

  - Prueba test:
    - Accuracy: 0.9031±0.0029
  - Validación cruzada (3):
    - Accuracy: 0.8928±0.0021
  - Hold Out(20%):
    - Accuracy: 0.8873±0.0035

- Etiquetado sentiment140 positivos negativos

Best score	0.7502
Best parameters set	
classifier	
C	1.0
dual	True
vectorizer	
max_df	0.5
min_df	5
ngram_range	(1, 2)
stop_words	True
tokenizer	True
use_idf	False
sublinear_tf	True

Tabla 6 Best parameters prueba 3.2

- Prueba test:
  - Accuracy: 0.7430±0.0043
  - AUC: 0.8209

- Validación cruzada (3):
  - Accuracy: 0.7408±0.0089
  - AUC: 0.8170

- Hold Out(20%):
  - Accuracy: 0.7455±0.0048
  - AUC: 0.8276

3. Algoritmo K-Neightbors
   Los valores del algoritmo a optimizar son:
   k: 5;10;50

- Etiquetado manual con descarte de *tweets* neutros

Best score	0.8945
Best parameters set	
classifier	
n_neightbors	10
vectorizer	
max_df	0.5
min_df	5
ngram_range	(1, 1)
stop_words	True
tokenizer	False
use_idf	True
sublinear_tf	True

Tabla 7 Best parameters prueba 1.3

- Prueba test:
  - Accuracy: 0.8942±0.0042
  - AUC: 0.9525

- Validación cruzada (3):
  - Accuracy: 0.8957±0.0105
  - AUC: 0.9477

- Hold Out(20%):
  - Accuracy: 0.8906±0.0048
  - AUC: 0.9484

- Etiquetado manual sin descarte de *tweets* neutros

Best score	0.6798
Best parameters set	
classifier	
n_neightbors	50
vectorizer	
max_df	0.5
min_df	5
ngram_range	(1, 1)
stop_words	True
tokenizer	False
use_idf	False
sublinear_tf	True

Tabla 8 Best parameters prueba 2.3

  - Prueba test:
    - Accuracy: 0.6880±0.0046
  - Validación cruzada (3):
    - Accuracy: 0.6777±0.0063
  - Hold Out(20%):
    - Accuracy: 0.6767±0.0052

- Etiquetado sentiment140 positivos negativos

Best score	0. 6987
Best parameters set	
classifier	
n_neightbors	50
vectorizer	
max_df	0.5
min_df	5
ngram_range	(1, 1)
stop_words	True
tokenizer	False
use_idf	False
sublinear_tf	True

Tabla 9 Best parameters prueba 3.3

- Prueba test:
  - Accuracy: 0.6965±0.0045
  - AUC: 0.7519

- Validación cruzada (3):
  - Accuracy: 0.6960±0.0061
  - AUC: 0.7662±0.0061

- Hold Out (20%):
  - Accuracy: 0.7023±0.0050
  - AUC: 0.7737

4. Algoritmo MLPClassifier

Los valores del algoritmo a optimizar son:
hidden_layer_sizes: (2,5,5) (1,5)
max_iter: 1000,500
alpha: 0.0001,1.0
learning_rate: invscaling, constant ,adaptive
learning_rate_init: 0.001, 1.0

- Etiquetado manual con descarte de *tweets* neutros

Best score	0.9754
Best parameters set	
classifier	
hidden_layer_sizes	(1,5)
max_iter	1000
alpha	1.0
learning_rate	invscaling
learning_rate_init	0.001
vectorizer	
max_df	0.5
min_df	5
ngram_range	(1, 1)
stop_words	True
tokenizer	True
use_idf	True
sublinear_tf	True

Tabla 10 Best parameters prueba 1.4

- Prueba test:
  - Accuracy: 0.9668±0.0024
  - AUC: 0.9960

- Validación cruzada (3):
  - Accuracy: 0.9672±0.0038
  - AUC: 0.9951

- Hold Out (20%):
  - Accuracy: 0.9609±0.0030
  - AUC: 0.9938

- Etiquetado manual sin descarte de *tweets* neutros

Best score	0.9312
Best parameters set	
classifier	
hidden_layer_sizes	(1,5)
max_iter	1000
alpha	1.0
learning_rate	invscaling
learning_rate_init	0.001
vectorizer	
max_df	0.5
min_df	5
ngram_range	(1, 1)
stop_words	True
tokenizer	True
use_idf	True
sublinear_tf	True

Tabla 11 Best parameters prueba 2.4

- Prueba test:
  - Accuracy: 0.9222±0.0026

- Validación cruzada (3):
  - Accuracy: 0.9150±0.0010

- Hold Out (20%):
  - Accuracy: 0.9143±0.0031

- Etiquetado sentiment140 positivos negativos

Best score	0.7412
Best parameters set	
classifier	
hidden_layer_sizes	(1,5)
max_iter	1000
alpha	1.0
learning_rate	invscaling
learning_rate_init	0.001
vectorizer	
max_df	0.5
min_df	5
ngram_range	(1, 1)
stop_words	True
tokenizer	True
use_idf	True
sublinear_tf	True

Tabla 12 Best parameters prueba 3.4

  - Prueba test:
    - Accuracy: 0.7368±0.0043
    - AUC: 0.8183

  - Validación cruzada (3):
    - Accuracy: 0.7398±0.0093
    - AUC: 0.8089

  - Hold Out (20%):
    - Accuracy: 0.7455
    - AUC: 0.8306
    - Error ±Desviación estandar: 0.2545±0.0048

6.2.2 Resultados obtenidos

En este apartado vamos a mostrar los resultados que hemos obtenido en los diferentes casos además de los resultados que obtienen otras herramientas usando técnicas diferentes. El valor Optim, se refiere al valor obtenido en la optimización que no tiene por qué coincidir con el resto. Como se explicó en el apartado anterior, los casos que propusimos fueron:

- Caso 1: aprendizaje supervisado utilizando el etiquetado binario propuesto en este proyecto.

- Caso 2: aprendizaje supervisado utilizando el etiquetado ternario (positivo-negativo-neutro) propuesto en este proyecto.
- Caso 3: aprendizaje supervisado para el etiquetado propuesto en el *dataset* que usaremos de la herramienta Sentiment140 en ambos casos. Será clasificación binaria.
- Caso 4: se usará el modelo binario generado en el caso 1 frente al etiquetado usado en el caso 3.

- Resultados *accuracy*

	Caso 1				Caso 2			
	Optim	Test	HO	VC	Optim	Test	HO	VC
NB	0.961	0.960±0.002	0.952±0.003	0.957±0.001	0.772	0.771±0.004	0.766±0.005	0.769±0.002
SVM	0.964	0.973±0.002	0.964±0.003	0.973±0.002	0.900	0.903±0.003	0.887±0.004	0.893±0.002
KNN	0.895	0.894±0.004	0.891±0.005	0.896±0.010	0.680	0.688±0.005	0.677±0.005	0.678±0.006
MLP	0.975	0.967±0.002	0.961±0.003	±0.004	0.931	0.922±0.003	0.914±0.003	±0.001

Tabla 13 Resultados accuracy 1 dataset 1

	Caso 3				Caso 4
	Optim	Test	HO	VC	CargaTest
NB	0.753	0.756±0.004	0.762±0.005	0.759±0.009	0.672±0.002
SVM	0.750	0.743±0.004	0.746±0.005	0.741±0.009	0.640±0.002
KNN	0.699	0.697±0.005	0.702±0.005	0.696±0.006	0.645±0.002
MLP	0.741	0.737±0.004	0.746±0.005	0.740±0.009	0.643±0.002

Tabla 14 Resultados accuracy 2 dataset 1

- *Resultados AUC*

	Caso 1			Caso 3			Caso 4
	Test	HO	VC	Test	HO	VC	CargaTest
NB	0.994	0.990	0.991	0.838	0.847	0.837	0.737
SVM	0.996	0.996	0.995	0.821	0.828	0.817	0.700
KNN	0.953	0.948	0.948	0.752	0.774	0.766	0.693
MLP	0.996	0.994	0.995	0.818	0.831	0.8306	0.698

Tabla 15 Resultados AUC dataset 1

6.2.3 Discusión de resultados

A la vista de los resultados obtenidos en las pruebas podemos realizar las siguientes conclusiones:

- El etiquetado binario obtiene mejores resultados que el caso del etiquetado ternario, como muestran los resultados de Caso 1 y caso 2

- El etiquetado binario que proponemos obtiene mejores resultados que el etiquetado propuesto en el primer *dataset* de sentiment140. Véanse los resultados de Caso 1 y Caso 3.
- Los resultados que obtenemos al enfrentar el modelo generado con nuestro etiquetado contra el etiquetado propuesto en el *dataset* no son buenos. Esto nos indica que no es suficiente con disponer de datos etiquetados. Es necesario además conocer el proceso de etiquetado de los datos.
- Los resultados obtenidos con los algoritmos utilizados son muy similares salvo en el caso de KNN que son ligeramente inferiores. La diferencia no es significativa en términos de la tasa de error, aunque ligeramente mayor con la métrica AUC
- La comparación de las columnas Optimo, *Test*, *Hold out* y Validación cruzada indica que la selección de parámetros óptimos no parece sobreajustar la hipótesis generada.

6.3 Experimentos segundo *dataset*

Para contrastar los resultados obtenidos se ha buscado otro *dataset* [dataset2], también de *tweets* aleatorios, de 1.6 millones de registros de los cuales se han seleccionado unos 50000. De los estos datos se utilizarán 80% para *train* y 20% para *test*. Este *dataset* también ha sido generado por la misma herramienta. Para cada uno de los cuatro casos de pruebas se ha elegido uno de los clasificadores con mejores resultados sobre el conjunto de *test* y se han creado nuevos modelos para evaluar estos datos.

6.3.1 Selección de modelos

- Caso 1: Se ha elegido el clasificador SVM con los datos siguientes y obtenido los resultados:

Best score dataset 1	0.964
Best parameters set	
classifier	
C	1.0
dual	True
vectorizer	
max_df	0.5
min_df	5
ngram_range	(1, 2)
stop_words	True
tokenizer	True
use_idf	True
sublinear_tf	True

Tabla 16 Datos Caso 1 dataset 2

- Prueba test:
  - Accuracy: 0.9595±0.0026
  - AUC: 0.9939

- Validación cruzada (3):
  - Accuracy: 0.9614±0.0054
  - AUC: 0.9928

- Hold Out (20%):
  - Accuracy: 0.9547±0.0030
  - AUC: 0.9918

- Caso 2: Se ha elegido el clasificador MLP con los datos siguientes y obtenido los resultados:

Best score dataset 1	0.9312
Best parameters set	
classifier	
hidden_layer_sizes	(1,5)
max_iter	1000
alpha	1.0
learning_rate	invscaling
learning_rate_init	0.001
vectorizer	
max_df	0.5
min_df	5
ngram_range	(1, 1)
stop_words	True
tokenizer	True
use_idf	True
sublinear_tf	True

Tabla 17 Datos Caso 2 dataset 2

- Prueba test:
  - Accuracy: 0.9125±0.0026

- Validación cruzada (3):
  - Accuracy: 0.9102±0.0039

- Hold Out (20%):
  - Accuracy: 0.9138±0.0029

- Caso 3: Se ha elegido el clasificador Naïve Bayes con los datos siguientes y obtenido los resultados:

Best score dataset 1	0.7526
Best parameters set	
classifier	
alpha	1.0
fit_prior	True
vectorizer	
max_df	0.5
min_df	5
ngram_range	(1, 2)
stop_words	False
tokenizer	False
use_idf	False
sublinear_tf	True

Tabla 18 Datos Caso 3 dataset 2

  - Prueba test:
    - Accuracy: 0.7600±0.0036
    - AUC: 0.8443

  - Validación cruzada (3):
    - Accuracy: 0.7615±0.0067
    - AUC: 0.8476

  - Hold Out (20%):
    - Accuracy: 0.7599±0.0040
    - AUC: 0.8451

- Caso 4: Se ha elegido el clasificador MLP con los datos siguientes y obtenido los resultados:

Best score dataset 1	0.672
Best parameters set	
classifier	
alpha	1.0
fit_prior	False
vectorizer	
max_df	0.5
min_df	5
ngram_range	(1, 1)
stop_words	True
tokenizer	True
use_idf	False
sublinear_tf	True

Tabla 19 Datos Caso 4 dataset 2

  - CargaTest
    - Accuracy: 0.6611±0.0018
    - AUC: 0.7295

6.3.2 Resultados obtenidos

Como se puede observar en todos los casos, los resultados son muy similares a los obtenidos en las pruebas con el otro *dataset* apenas oscilando en ±1. Con esto podemos deducir que los resultados no son casuales a los datos del primer *dataset* y que por ello con este método se han obtenido buenos resultados.

	Test	HO	VC
Caso1: MLP	0.960±0.003	0.955±0.003	0.961±0.005
Caso2: MLP	0.913±0.003	0.910±0.004	0.914±0.003
Caso 3: NB	0.760±0.004	0.762±0.007	0.760±0.004

Tabla 20 Resultados accuracy dataset 2

	CargaTest
Caso 4: NB	0.661±0.002

Tabla 21 Resultados accuracy 2 dataset 2

	Test	HO	VC
Caso1: MLP	0.9940	0.9912	0.9936
Caso2: MLP	0.8443	0.8476	0.8451

Tabla 22 Resultados AUC dataset 2

	CargaTest
Caso 4: NB	0.7295

Tabla 23 Resultados AUC 2 dataset 2

Se observa que los resultados son similares a los del conjunto de datos anterior. Teniendo en cuenta que ambos conjuntos de datos se han obtenido con la misma metodología, recogida aleatoria de tweets, estos resultados sugieren que los métodos propuestos son competitivos sobre este tipo de conjuntos de datos.

6.4 Otras herramientas y datasets

Se han investigado resultados sobre otros *dataset* y con otras técnicas. Estos resultados vienen descritos en un artículo muy completo sobre *benchmark* en análisis de sentimiento [Ribeiro2016]. La mayoría de estas herramientas utilizan técnicas de etiquetado usando diferentes diccionarios, así como un análisis lingüístico, con su posterior enfoque de *machine learning*. Centrándonos en los conjuntos similares al nuestro, es decir los que utilizan *tweets* aleatorios tenemos dos casos de posibles resultados en términos de *accuracy*:

- Clasificación binaria: en la imagen inferior se muestran algunos de los resultados de otros métodos utilizados. Como vemos son similares a los nuestros pudiendo superar al igual que nosotros el 90% dependiendo del método.

Opinion Finder	78.32
Opinion Lexicon	93.45
PANAS-t	90.71
Pattern.en	91.76
SASA	70.06
Semantria	91.61
SenticNet	73.64
Sentiment140	94.75

45 Clasificación binaria con otras herramientas

- Clasificación ternaria: en la imagen inferior se muestran algunos de los resultados de otros métodos utilizados. Como vemos en este caso son peores

que los nuestros ya que la mayoría no superan el 60% cuando en nuestro caso como se ha visto en apartado 6.1 podemos llegar hasta un 90%.

Opinion Finder	57.63
Opinion Lexicon	60.37
PANAS-t	53.08
Pattern.en	57.99
SASA	50.63
Semantria	61.54
SenticNet	49.68
Sentiment140	60.42

46 Clasificación ternaria con otras herramientas

7 Conclusiones

Tras la finalización de este TFM, podemos concluir que se han cumplido casi todos los objetivos inicialmente fijados, y en base a ellos se puede afirmar que:

- Se ha presentado y realizado cada una de las partes de una arquitectura Big Data
- Se ha realizado una extracción de datos de una plataforma de red social como es Twitter con una herramienta de Big Data. Serían necesarias mejoras adicionales para obtener lo conjuntos de datos de utilidad con menos tiempo de cómputo.
- Se ha realizado un almacenamiento y transformación de los datos extraídos en un medio Big Data.
- Se ha realizado un análisis de sentimiento mediante técnicas de clasificación basadas en aprendizaje automático y en el léxico.
- Se ha creado una herramienta con la que poder configurar el análisis de sentimiento que además puede clasificar cualquier tipo de texto.
- La metodología propuesta es competitiva contra los resultados propuestos por otros autores y otros conjuntos de datos.

Mediante la consecución de dichos objetivos, el alumno ha adquirido cierta soltura con el lenguaje de programación Python y sus diferentes bibliotecas y en especial las destinadas a la clasificación y el estudio del lenguaje natural.

Además, se ha profundizado en el uso de herramientas de Big Data de las que no se tenían muchos conocimientos.

Como trabajo futuro se podría realizar:

- Introducción de nuevos algoritmos de clasificación.
- Introducción de algoritmos de *Spark Mlib/Ml* para poder trabajar con cantidades mucho más grandes de datos.
- Mejoras en la visualización de la interfaz. (añadir estadísticas adicionales, matriz de confusión…)
- Mejora de las descargas de *tweets* con *Kafka*. Por limitaciones del método solo se permitía descargar una cantidad pequeña de datos, por lo que, para recopilar suficientes, era necesaria una cantidad de tiempo excesiva.
- Realizar un etiquetado semisupervisado.
- Implementar otras técnicas como el uso de ontologías.

8 Bibliografía

[Cárdenas2014] Cárdenas, P., Olivares, G. y Alfaro, R. , Clasificación automática de textos usando redes de palabras, Revista Signos, Estudios de Lingüística, 2014 PUCV, Chile.

[Go2009] Go,A. , Bahayani, R. y Huang, L., Twitter Sentiment Classification using Distant Supervision. Technical report, Stanford. United States, 2009.

[Gupta09] Gupta, V. y Gurpreet S. Lehal, A Survey of Text Mining Techniques and Applications. Journal of emerging tehcnologies in web intelligence, vol. 1, no. 1, August 2009, India.

[Huang2008] Huang, A., Similarity Measures for Text Document Clustering, Proceedings of the 6th New Zealand Computer Science Research Student Conference, New Zealand, 2008.

[Hu2004]. Minqing H. y Bing L. Mining and Summarizing Customer Reviews, Proceedings of the ACM SIGKDD International Conference on Knowledge, Discovery and Data Mining (KDD-2004), Aug 22-25, 2004, Seattle. Washington, USA

[Kontopoulos2013] Kontopoulos, E., Berberidis, C., Dergiades, T. y Bassiliades, N., Ontology-based sentiment analysis of twitter posts. Journal Expert Systems with Applications Volume 40, Issue 10, August 2013, Pages 4065-4074, Greece . 2013.

[Mahesh2005] Mahesh P., Multiclass Approaches for Support Vector Machine Based Land Cover Classification, in Proc. 8th Annu. Int. Conf., Map India, 2005.

[Medhat2014] Sentiment analysis algorithms and applications: A survey. Medhat, W., Hassan, A. y Korashy, H., Ain Shams Engineering Journal, Volume 5, Issue 4, December 2014, Pages 1093-1113, 2014

[Monroy-de-Jesús2015] Monroy-de-Jesús, J. Alejo, R., Lòpez-Gonzàlez, E., Antonio-Velázquez, J. y Gil-Antonio, L., Heurísticas para mejorar el rendimiento del algoritmo Backpropagation, Mexico, 2015.

[Ribeiro2016] Ribeiro, F.N., Araújo, M., Gonçalves, P., Gonçalves, M.A. y Benevenuto, F., SentiBench, a benchmark comparison of state of the practice sentiment analysis methods, al, EPJ Data Sci, 5: 23, Brazil, 2016

[Wilson2005] Wilson, T., Wiebe, J. y Hoffmann, P., Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. In HLT ’05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, pages 347– 354, Morristown, NJ, USA. Association for Computational Linguistics.

[dataset] Twitter Sentiment Analiysis Corpus. http://thinknook.com/twitter-sentiment-analysis-training-corpus-dataset-2012-09-22/ .Última visita 17/07/2018

[dataset2] Sentiment analisys with tweets, Kaggle, https://www.kaggle.com/kazanova/sentiment140 , Última visita 17/07/2018

[Hive] Apache Hive, Apache Software Foundation, http://hive.apache.org ,Última visita 17/07/2018

[Kafka] Apache Kafka, Apache Software Foundation, https://kafka.apache.org ,Última visita 17/07/2018

[liub/FBS] Opinion Mining, Sentiment Analysis, and Opinion Spam Detection, Bing Liu and Minqing Hu, https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html .Última visita 17/07/2018

[mpqa] Subjetive Lexicon. Multi perspective question answering, http://mpqa.cs.pitt.edu/lexicons/subj_lexicon/.Última visita 17/07/2018

[PyQt5] Qt5 Documentation, The Qt Company Ltd, http://doc.qt.io/qt-5/qtwidgets-module.html , Última visita 17/07/2018

[Sentiment140] Sentiment140. http://help.sentiment140.com/for-students ,Última visita 17/07/2018

[Sklearn] Scikit-learn, Machine Learning in Python. Open source, commercially usable - BSD license, http://scikit-learn.org/stable/index.html .Última visita 17/07/2018

[Python Env] Virtual Environments and Packages, Python Software Foundation, https://docs.python.org/3/tutorial/venv.html .Última visita 17/07/2018

Apéndice A: Manual de instalación

Para el uso de nuestra aplicación se necesita como único requisito principal tener instalado una versión de Python igual o superior a la 3. Con Python instalado nos dirigimos al terminal y nos colocamos en el directorio que queramos, desde donde vamos a crear un entorno virtual [Python Env] de trabajo para lo cual ejecutamos el comando:

python3 -m venv EntornoPython

Con esto creará un directorio "*EntornoPython"* en el que colocaremos el directorio "data" que se encuentra en el CD. En este directorio se encuentran los ejecutables, asi como una serie de directorios donde encontrar diferentes ficheros para utilizar la herramienta. Para activar el entorno tenemos que lanzar el siguiente comando, distinto en Windows que en Linux/OS X:

Windows: EntornoPython\Scripts\activate.bat

Linux: source EntornoPython/bin/actívate

Una vez activado el entorno nos dirigimos al directorio data que hemos copiado donde tenemos el fichero requirements.txt con el listado de bibliotecas necesarias y las versiones usadas. Para descargarlas desde el directorio data, usamos el comando:

cd EntornoPython/data/
pip install -r requeriments.txt

Tras la realizar la descarga de las bibliotecas en algunos casos en Windows es necesario una descarga adicional, para lo que ejecutamos fichero nltkPunkt.py alojado en el directorio "*EntornoPython*/data", con el siguiente comando:

python3 nltkPunkt.py

Finalmente, para ejecutar la herramienta ya solo nos queda ejecutar la herramienta también alojada en el directorio "*EntornoPython*/data", para lo que usamos el comando:

python3 analisisSentimiento.py