Universidad Autónoma Metropolitana Azcapotzalco.
División de Ciencias Básicas e Ingeniería.

Licenciatura en Ingeniería en Computación.

Reporte final del proyecto de Integración.

Sistema para la clasificación de opiniones generadas en twitter usando redes neuronales artificiales.

Proyecto Tecnológico

Alumno:
Rojas Luis Victor Hugo
209300679

Asesor:
Dra. Maricela Claudia Bravo Contreras
Profesor Asociado
Departamento de Sistemas

Co-asesor:
Dr. José Alejandro Reyes Ortiz
Profesor Titular
Departamento de Sistemas

Trimestre 2016 Otoño
Fecha de entrega
04 de enero de 2017

Declaratoria

Yo, Dra. Maricela Claudia Bravo Contreras, declaro que aprobé el contenido del presente reporte de Proyecto de Integración y doy autorización para su publicación en la Biblioteca Digital, así como en el repositorio Institucional de la UAM Azcapotzalco.

Dra. Maricela Claudia Bravo Contreras

Yo, Dr. José Alejandro Reyes Ortiz, declaro que aprobé el contenido del presente reporte de Proyecto de Integración y doy autorización para su publicación en la Biblioteca Digital, así como en el repositorio Institucional de la UAM Azcapotzalco.

Dr. José Alejandro Reyes Ortiz

Yo, Victor Hugo Rojas Luis, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como el Repositorio Institucional de la UAM-Azcapotzalco

Víctor Hugo Rojas Luis

Resumen

La cantidad excesiva de documentos que se hace por usuarios finales, en un lenguaje natural disponibles en formato electrónico hace imposible su análisis, un resultado propuesto a este problema es la extracción de información, las cuales permiten estructurar datos relevantes a un dominio especifico en los documentos. En otras palabras, la extracción de información convierte el problema de analizar una colección de texto en consultar una base de datos.

El dominio de las redes sociales, se generan grandes volúmenes de texto, se estudian y se analizan con algoritmos clásicos de la programación, aprovechando las opiniones que comparten los consumidores sobre un producto o servicio, además, este método puede resultar poco eficiente y en algunos casos inoperable para un determinado problema. Para apoyar esta tarea, se tiene como requisito utilizar un algoritmo computacional y técnicas de procesamiento de lenguaje natural, que facilite dicho análisis.

Como parte del proyecto de integración se propone diseñar una herramienta computacional que logre la clasificación automática de la información contenida en las redes sociales, para lo cual se pretende utilizar el procesamiento de lenguaje natural, extracción de información, identificación de patrones TF/IDF y booleano y finalizando con la clasificación de clases según la polaridad de las emociones que expresan sobre determinados objeto, servicio o producto.

En este proyecto se emplea el algoritmo de una red neuronal multicapa, con la finalidad de clasificar documentos automáticamente según polaridad de emociones (positivo, negativo y neutro), como caso de uso de prueba, el sistema se desarrolló para la red social twitter.

Índice

1. Introducción 1

2. Antecedentes 1

2.1 Referencias Internas 1

2.2 Referencias Externas 2

3. Justificación 3

4. Objetivos 4

4.1 Objetivo General 4

4.2 Objetivo Específicos 4

5. Marco Teórico 4

5.1 Red Neuronal Artificial 4

5.1.1 Aplicaciones de las Redes Neuronales 6

5.2 Perceptrón multicapa 7

5.2.1 Limitaciones de un Perceptrón 8

5.3 Clasificación con WEKA 9

5.3.1 Modos de evaluación del clasificador 10

5.3.2 Configuración de Modos de entrenamiento 11

5.4 Minería de Textos 12

5.4.1 Data Mining 12

5.5 Minería de Opiniones. 13

6. Desarrollo Del Proyecto 13

6.1 Etapa de Entrenamiento. 14

6.1.1 Pre-procesamiento del texto 15

6.1.2 Extracción de características 15

6.1.3 Representación de Datos. 16

6.2 Etapa de pruebas. 17

7. Resultados 23

8. Conclusiones 24

9. Bibliografía 25

10. Anexos 26

10.1 Clase Principal 26

10.2 Clase Abrir Archivo..... 27
10.3 Clase Escritura..... 29
10.4 Clase Vector..... 37

Índice de Figuras

Figura 1. Componente de una neurona. 5
Figura 2. Sistema Global de proceso de una red Neuronal. 6
Figura 3. Topología de un Perceptrón Multicapa. 8
Figura 4. Topología de un Perceptrón Multicapa. 9
Figura 5. Matriz de Confusión WEKA. 11
Figura 6. Etapas de desarrollo del proyecto. 14
Figura 7. Fichero con Texto y Polaridad. 16
Figura 8. Tabla de comparación con la consola de NetBeans IDE 8.1. 17
Figura 9. Tabla de comparación con el número de coincidencias del vocabulario y su polaridad Archivo.csv. 17
Figura 10. Red neuronal MultilayerPerceptrón. 18
Figura 11. Modo Split. 19
Figura 12. Tabla de Resultados. 23

1. Introducción

A lo largo de la historia del hombre en la naturaleza y los distintos procesos complejos que se ven involucrados en ella, estos han servido de inspiración para resolver problemas de la vida cotidiana. Los algoritmos basados en redes neuronales son una rama de la Inteligencia Artificial en la que se emula el comportamiento de los sistemas naturales con el fin de diseñar métodos de aprendizaje, reconocimiento, simulación y caracterización.

Los algoritmos basados en redes neuronales artificiales son sistemas construidos por medio de software, que con ayuda de la inteligencia artificial descubren y describen aspectos de la inteligencia humana que pueden ser simulados mediante maquinas, con la finalidad de procesar información y resolver problemas de clasificación, reconocimiento de patrones y aprendizaje automático.

Este tipo de algoritmos son de utilidad para el análisis de textos, entre otras tareas, la clasificación de textos, la cual se logra mediante el aprendizaje de los pesos de ciertas características a partir de un conjunto de entrenamiento, para que, posteriormente, se logre evaluar con datos que no han sido clasificados.

Por lo tanto, en este trabajo de proyecto de integración describimos el arte de una red neuronal artificial de tipo perceptrón multicapa para la clasificación de opiniones generadas en la red social Twitter sobre productos o servicio. Los textos de las opiniones se convirtieron en un modelo de bolsa de palabra TF/IDF (frecuencia del termino-frecuencia inversa de documento) y el peso booleano, estos datos fueron las entradas para el entrenamiento de la red neuronal multicapa y así hacer el proceso de clasificación.

La clasificación consistió en determinar si una opinión es positiva, negativa o neutra. Las redes sociales nos proporcionaron información actualizada y fresca sobre lo que se está hablando de ciertos productos, además son datos abiertos y libres que pueden ser utilizados sin infringir en la propiedad intelectual.

Trabajos Relacionados

2. Antecedentes

En esta sección se muestran algunos trabajos relacionados con el proyecto que se quiere realizar.

2.1 Referencias Internas

Dentro de la institución se han desarrollados Proyectos de Integración que comprenden temas relacionados con el Análisis de Sentimientos, es decir la identificación y extracción de opiniones emitidas en textos, utilizando el Procesamiento de Lenguaje Natural y clasificación de los mismo.

- **Detectando la prioridad de contenido generados en twitter por medio de n-gramas de palabra. [1]:**Este proyecto es un parte fundamental, radica en cuanto que se aborda el tema de proponer métodos automáticos que apoyen a las labores de los Analistas de Reputación en Línea de las empresas, para determinar cuándo un tuit es importante dentro de una categoría predefinida de mensaje, ya sea artistas, personajes públicos y políticos entre otro, mediante el twitter analizan y clasifican las opiniones (positiva, negativa y neutras), para tomar decisiones estratégicas que ayuden a mejorar la reputación de los mismo.

- **Sistema de recuperación de información semántico. [2]:** aquí se trata de aplicar técnicas para obtener información semántica, y así recolectar, descubrir información relevante sobre un conjunto de documentos, utilizando el Procesamiento del Lenguaje Natural(NLP) analizando la traducción automática, la recuperación y extracción de información con la finalidad de estudiar el significado de las palabras claves(unigramas)

2.2 Referencias Externas

- **Inteligencia Artificial, métodos bio-inpirados: Un enfoque funcional para la ciencia de la computación. [3]:** Este proyecto terminal es una parte fundamental introductoria en saber que tanto nos podemos apoyar de los algoritmos bio-inpirados para solucionar problemas de alta complejidad computacional, dando a conocer los diferentes métodos utilizados para poder encontrar la mejor solución

- **Procesamiento de lenguaje Natural en Sistemas de Análisis de Sentimiento. [4]:** En esta tesis se aplican técnicas de Procesamiento de lenguaje Natural para la extracción, análisis, identificación y evoluciones de opiniones, sentimientos y emociones emitidas en el texto, para el análisis de información subjetiva, clasificando por su polaridad (positivo y negativo).

- **Una Aplicación de Web *OpinionMining* para la Extracción de Tendencias y Tópicos de relevancias a partir de las Opiniones Consignadas en Blog y Sitios de Noticias. [5]:** El objetivo del articulo consiste en la extracción de información para tomar decisiones estratégicas según las opiniones del producto que se encuentran en noticias o blogs, tomando en cuenta que en la web 2.0 donde grandes cantidades de información son creados por los usuarios, es decir, en esta web predominan sistemas de foros, redes sociales, blog y sistema de recomendación todo ello por una alta interacción de usuarios, así mismo las empresas obtienen información para la administración de la mejor manera posible sus recursos y al mismo tiempo adelantan a cada movimiento de la competencia.

- **Wordstat7 [6]:** Es un software para la extracción y el análisis rápido de información de enormes documentos, unas de sus aplicaciones es el análisis competitivo de sitios web que ayuda a la clasificación de las páginas, como una

opinión positiva, negativa y neutra. A diferencia nosotros pretendemos implementar una red neuronal para la clasificación de productos en la red social twitter.

3. Justificación

La cantidad de información que existe en los medios digitales es demasiada, más cuando se utilizan buscadores, los cuales a través de palabras clave devuelven grandes cantidades de páginas web, blogs, documentos digitales, redes sociales, foros, etc. Estas grandes cantidades de resultados devueltos por el buscador ocasionan que sea muy difícil su análisis. Desde hace muchos años, las empresas han fundamentado su éxito en el estudio de la información recuperada de redes abiertas como la Internet, investigan sobre los productos o servicios que ofrecen así como la opinión que se genera sobre sus productos, sus marcas y la empresa. Esta necesidad de análisis de las opiniones fue creciendo y hoy en día existen grupos de analistas que se dedican a estudiar dicha información de los medios digitales con el objetivo de clasificarla de varias maneras.

El análisis de opiniones es importante ya que ayuda a obtener el tipo de opinión, ya sea positiva, negativa o neutra sobre objetos, productos o servicios, para lo cual se utilizan clasificadores, algoritmos computacionales y técnica de Procesamiento del Lenguaje Natural para estudiar el texto.

Esta propuesta sigue al quehacer computacional, en el sentido de que se aplicaron técnicas innovadoras para la extracción automatizada de la información, aplicando el conocimiento y métodos disponibles para la resolución del problema con ayuda de las redes neuronales artificiales y procedimientos de Procesamiento del Lenguaje Natural.

La importancia de estudiar las redes neuronales artificiales para la clasificación de opiniones generadas en Twitter radica en que éstas han probado ser eficientes en la clasificación de textos de dominios como descripciones de servicios web, noticias, entre otros. Además, al ser algoritmos inspirados en el comportamiento de neuronas biológicas, se desea validar su eficacia para esta tarea.

Por ello, se proponer diseñar una herramienta computacional que permita extraer textos cortos de la red social (Twitter) y clasificarlos como opiniones positivas, negativas o neutras acerca de un producto, para lo cual se propone utilizar técnicas de Procesamiento del Lenguaje Natural y algoritmos basados en redes neuronales artificiales. Además, los textos de las opiniones serán representados en un modelo de palabras con un pesado estadístico y binario.

4. Objetivos

4.1 Objetivo General

Diseñar e implementar un sistema para la clasificación los textos de opiniones de productos generadas en la red social Twitter, en base a su polaridad: positivo, negativo o neutro, utilizando una red neuronal artificial de tipo perceptrón multicapa y un modelo de bolsa de palabras para la representación de los textos.

4.2 Objetivo Específicos

- Diseñar e implementar un método para procesar los textos de la red social
- Extraer la bolsa de palabras con sus pesos utilizando métodos estadísticos como la

  frecuencia de aparición y un pesado binario, con la finalidad de obtener el conjunto de datos de entrenamiento y el conjunto de datos para validar la clasificación con la red neuronal.
- Implementar una red neuronal artificial de tipo perceptrón multicapa para la clasificación de opiniones de productos en tres categorías: positivo, negativo o neutro.
- Diseñar una etapa de entrenamiento de la red neuronal y una etapa de clasificación de un conjunto de textos utilizando los dos métodos estadísticos.

5. Marco Teórico

5.1 Red Neuronal Artificial

En 1880 Ramón y Cajal demuestra que el sistema nervioso está compuesto por una red de células individuales, las neuronas, ampliamente interconectadas entres si, la información fluye desde las dendritas hacia el axón atravesando el soma, es decir el cerebro humano es el sistema de cálculo más complejo que se conoce por el hombre. La Figura 1 muestra las partes que constituyen una neurona.

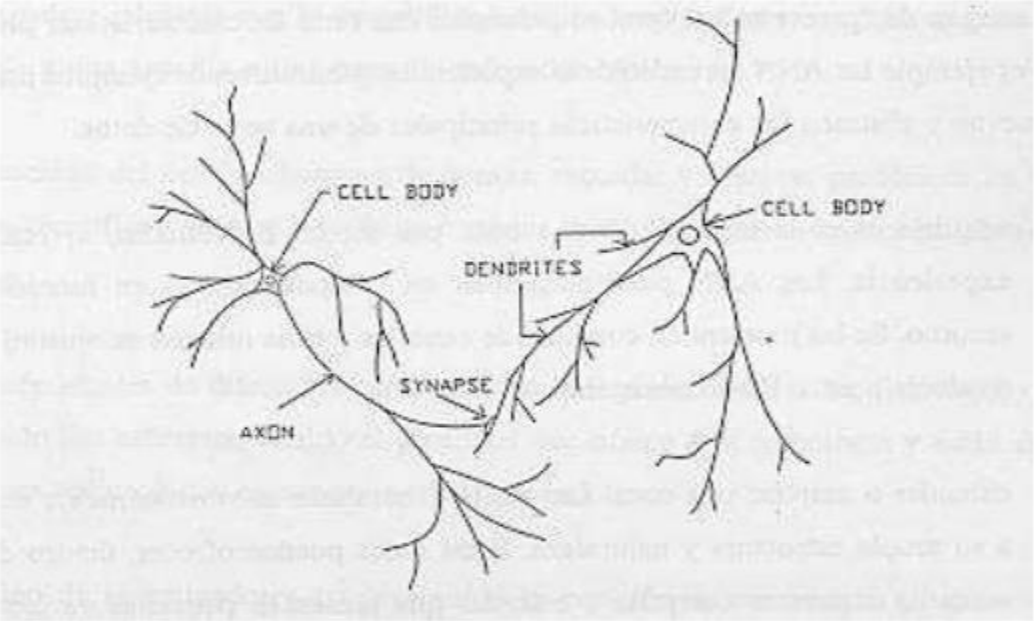

Figura 1. Componente de una neurona.

La capacidad del cerebro humano de pensar, recordar, analizar, interpretar emociones y resolver problemas has inspirado a muchos científicos, dirigiéndolos a la Inteligencia Artificial es un intento de descubrir y describir aspecto de la inteligencia humana que puedan ser simulados mediante maquinas. El resultado ha sido una nueva tecnología llamada Computación Neuronal o Red Neuronal Artificial.

La Redes Neuronales Artificiales, ANN (Artificial Neural Networks), están constituidas por elemento que se comportan de forma similar a la neurona biológica en las funciones más comunes, se organiza de una forma parecida a la que presenta el cerebro humano.

Una red neuronal es “un nuevo sistema para el tratamiento de información, cuya unidad básica de procesamiento está inspirada en la célula fundamental del sistema nervioso humano: la neurona” **[7]**

El ANN aprende de las experiencias, generalizan de ejemplos previos a ejemplos nuevos y abstraen las características principales de un conjunto de datos. **[8]**

Aprender es adquirir conocimientos de un objeto por medio de su estudio o ejercicio logrando experiencia, es decir las ANN puede cambiar su comportamiento en función del entorno, mostrándoles una serie de entradas y ellas mismas se ajustan para producir unas salidas consistentes.

Generalizar es extender o ampliar el conocimiento de algún objeto. Las ANN generalizan automáticamente debido a su propia estructura y naturaleza, ofreciendo dentro de un

margen, respuestas correctas a entradas que presentan pequeñas variaciones debido a los efectos del ruido o distorsión.

Abstraer aislar mentalmente o considerar por separado las cualidades de un objeto, algunas ANN son capaces de abstraer la esencia de un conjunto de entradas que aparentemente no presentan aspectos comunes o relativos.

El modelo estándar de una red neuronal artificial se dio en los principios de Rumelhart y McClelland en 1986, como se muestra en la Figura 2. Siguiendo dichos principios, la i-ésima neurona artificial estándar consiste en:

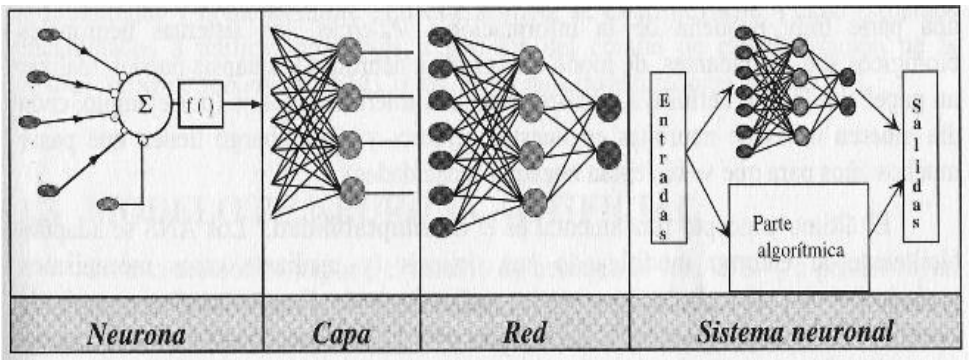

Figura 2. Sistema Global de proceso de una red Neuronal.

5.1.1 Aplicaciones de las Redes Neuronales

Las características especiales de los sistemas de computación neuronal permiten que sea utilizada esta nueva técnica de cálculo de una extensa variedad de aplicaciones.

Las Redes Neuronales Artificiales provee un acercamiento mayor al reconocimiento y percepción humana que los métodos tradicionales de cálculo, presentando resultados razonables en aplicaciones donde las entradas presentan ruidos o las entradas están incompletas.

Algunas de las áreas son:

Análisis y Procesados de señales, Reconocimiento de imágenes, Control de procesos, Filtrado de ruido, Robótica, Procesado del Lenguaje, Diagnostico médicos ya que cada una conlleva diferentes aplicaciones como son:

- Conversión texto a voz.
- Procesado Natural del Lenguaje.
- Comprensión de Imágenes.

- Reconocimiento de Caracteres.
- Reconocimiento de Patrones en Imágenes.
- Problemas de combinatoria.
- Procesado de la Señal, Predicción.
- Modelado del Sistema.
- Filtro del Ruido.
- Modelos Económicos y Financieros.
- ServoControl.

5.2 Perceptrón multicapa

En 1958 Frank Rosenblatt, comenzó en el desarrollo del Perceptrón, esta es la red neuronal más antigua, utilizándose hoy en día para resolver problemas de asociación de patrones, segmentación de imágenes, comprensión de datos, este modelo es capaz de generalizar, es decir después de haber aprendido una serie de patrones podría reconocer otro similares, aunque no lo hubiese presentado en el entrenamiento.

El Perceptrón multicapa es una red de alimentación hacia adelante, compuesta por una capa de unidades de entradas(sensores), otra capa de unidades de salida y un número determinado de capas intermedias de unidades de proceso, también llamadas capas ocultas por que se ven las salidas de dichas neuronas y no tiene conexiones con el exterior, cada sensor de entrada está conectados con las unidades de la segunda capa, y cada unidad de proceso de la segunda capa está conectada con las unidades de la primera capa y con las unidades de la tercera capa oculta, como se muestra en la Figura 3.

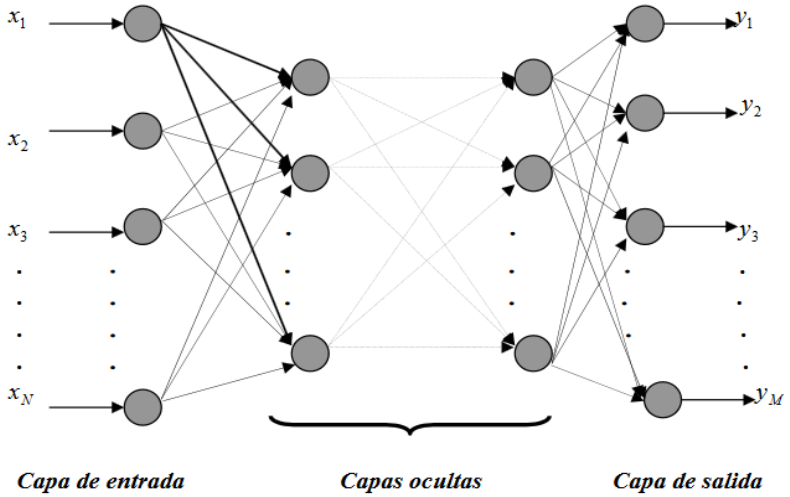

Figura 3. Topología de un Perceptrón Multicapa.

Es decir, con esta red se pretende establecer la correspondencia entre un conjunto de entradas y un conjunto de salidas deseadas de manera que

$$(x_1, x_2, \ldots, x_N) \in R^N \rightarrow (y_1, y_2, \ldots, y_M) \in R^M$$

Para ello se dispone de un conjunto de *p* patrones de entrenamiento, de manera que sabemos perfectamente que el patrón de entrada $(x_1^k, x_2^k, \ldots, x_N^k)$ le corresponde la salida $(y_1, y_2, \ldots, y_M)$, , k=1,2,...,p. Es decir, conocemos dicha correspondencia para p patrones, así nuestro conjunto de entrenamiento será:

$$\left\{(x_1^k, x_2^k, \ldots, x_N^k) \rightarrow (y_1^k, y_2^k, \ldots, y_M^k) : k = 1,2,\ldots, p\right\}$$

Para dicha relación, la primera capa tendrá tanto sensores como componentes tenga el patrón de entrada, es decir, N la capa de salida tendrá tanta unidades de proceso como componentes tenga las salidas deseadas, es decir, M y el número de capas ocultas y su tamaño dependerá de la dificultad de la correspondencia a implementar.

5.2.1 Limitaciones de un Perceptrón

Debemos tener en cuenta que no siempre el algoritmo de entrenamiento del Perceptron podrá converger hacia el error nulo. De hecho, el Perceptrón es incapaz de converger en aquellas funciones que no son linealmente separables, es decir aquellas cuyos elementos pueden ser separados por una línea recta.

Esto se debe a las propiedades inherentes de las unidades básicas del Perceptrón que son las neuronas artificiales, cuya limitación reside en la función de activación, esto es problema cuyas salidas estén clasificadas en dos categorías diferentes y que permitan que su espacio de entrada sea dividido en estas dos regiones por medio de un hiperplano de características similares.

Entendido de otra forma, podríamos decir que el Perceptron divide en dos grupos las entradas por medio de una línea divisora de manear que no es posibles separa elementos que no se encuentren claramente separados de otro elemento, es decir que no se pueden caracterizar elementos no lineales. Esto se puede observar en la Figura 4.

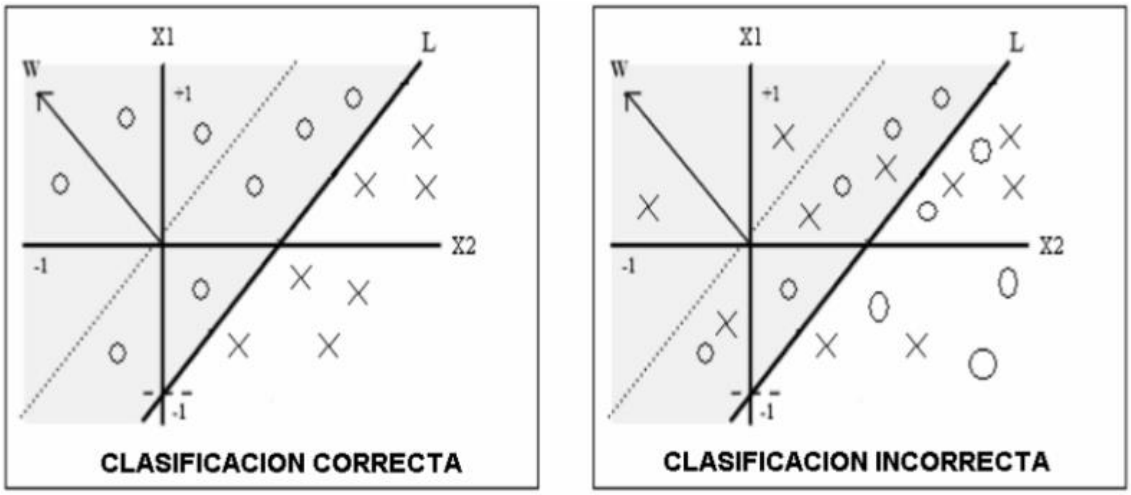

Figura 4. Topología de un Perceptrón Multicapa.

5.3 Clasificación con WEKA

A partir de la generación de los datos finales ya cargados en Weka, se realizó un estudio más a fondo, leyendo de una forma más clara cada atributo y llegar a conclusiones acertadas. En ocasiones, el problema de clasificación se formula como un refinamiento en el análisis, una vez que se han aplicado algoritmos de agrupamiento no supervisados y asociación para descubrir y describir relaciones de interés en los datos.

Se pretenden construir un modelo que permita predecir la categoría de las instancias en función de una serie de atributos de entrada. En el caso de WEKA , podremos aplicar una gran variedad de algoritmos de clasificación a los datos ya cargados, evaluar su rendimiento y seleccionar aquellos que proporcionen un mejor resultado, en este caso ocupamos el algoritmo Multilayer Perceptron basado en funciones, y nos dio como resultado una sentencia ordenada de cada uno de los valores, generado una mejor interpretación de los datos, la clase es simplemente uno de los atributos simbólicos

disponibles, que se convierte en la variable objetivo a predecir. Por defecto, es el último atributo (última columna).

5.3.1 Modos de evaluación del clasificador

El resultado de aplicar el algoritmo de clasificación se efectúa comparando la clase predicha con la clase real de las instancias. Esta evaluación puede realizarse de diferentes modos de entrenamiento.

- **Use training set:** Esta opción evalúa el clasificador sobre el mismo conjunto sobre el que se construye el modelo predictivo para determinar el error, que en este caso denomina “error de resustitución”, es decir WEKA se entrenó el método con todos los datos disponibles y luego se aplica sobre los mismos. Por tanto, esta opción puede proporcionar una estimación demasiado optimista del comportamiento del clasificador, al evaluarlo sobre el mismo conjunto sobre el que se hizo el modelo.

- **Supplied test set:** Evaluación sobre conjunto independiente. Esta opción permite seleccionar un fichero de un conjunto de los nuevos datos distintos a los del entrenamiento. Sobre cada dato se realiza una predicción de clase para contar los errores y se probó el clasificador obtenido con el método de clasificación usado y datos iniciales.

- **Cross – validation [9]:** Evaluación con validación cruzada de k hojas(k por omisión es de 10=). Esta opción es la más elaborada y costosa. Se realizan tantas evaluaciones como se indica en el parámetro **folds**. Dado un número *n* se dividen las instancias en *n* partes y en cada parte se construye el clasificador con las *n*-1 partes restantes y se prueba con esa, y así con cada una de las *n* particiones, los errores calculados son el promedio de todas las ejecuciones.

  Una validación cruzada es estratificada cuando cada una de las partes conserva las propiedades de la muestra original (porcentaje de elementos de cada clase).

- **Percentage Split:** Esta opción divide los datos en dos grupos, de acuerdo con el porcentaje indicado con el que se va a construir el clasificador y con la parte restante para las pruebas. El valor indicado es el porcentaje de instancias para construir el modelo de entrenamiento, que a continuación es evaluado sobre las que se han dejado aparte. Cuando el número de instancias es suficientemente elevado, esta opción es suficiente para estimar con precisión las prestaciones del clasificador en el dominio.

  Al utilizar WEKA desordena aleatoriamente el conjunto inicial, después parte del porcentaje indicado para el entrenamiento y el restante para las pruebas, de esta manera se construye el clasificador 2 veces, obteniendo dos desordenes distintos,

y por lo tanto dos porcentajes, es importante configurar esta opción para que solo la primera parte sea para la construcción del clasificador y la segunda el de pruebas.

5.3.2 Configuración de Modos de entrenamiento

Para obtener una interpretación adecuada en la interfaz es importante hacer las configuraciones correctas en el clasificador seleccionado, permitiendo un análisis prudente y llegar a conclusiones más acertadas (importante si se ha elegido SPLIT). Activamos algunas de estas opciones:

- **Output Model:** Si activamos esta opción una vez construido y probado el clasificador, nos mostro la salida del clasificador el modelo que ha construido (parte media derecha de la ventana)
- **Output per-class stats:** Activada, muestra estadísticas referentes a cada clase
- **Output entropy evalution mesures:** Muestra información de mediciones de la entropía en la clasificación
- **Output confusion matrix:** Muestra la matriz de confusión del clasificador, esta tabla cuyo número de columnas es el número de atributos muestra la clasificación de las instancias. Dando una información muy útil, porque nos refleja los errores producidos sino también informa del tipo de éstos como se muestra en la Figura 5.

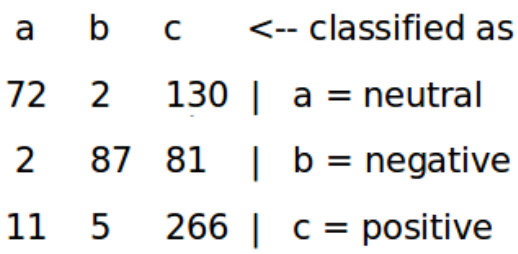

Figura 5. Matriz de Confusión WEKA.

Donde las columnas indican las categorías clasificadas por el clasificador y las filas las categorías reales de los datos, por lo que los elementos en la diagonal principal son los elementos que han acertado y los demás son los errores.

5.4 Minería de Textos

La minería de texto es una disciplina especializada en la obtención de información que no se encuentra de forma explícita en un conjunto de texto, esto es posible a través de la identificación de patrones y correlaciones de los términos contenidos en ellos, es decir, visto de otra manera se define como la extracción y descubrimiento de conocimiento no trivial a partir de textos no estructurados. Esto contempla una gran área de estudio desde *Information Retrieval* (*IR*), clasificación de textos y *Clustering* hasta extracción de entidades, eventos y relaciones de ellos. *Natural Language Processing* (*NLP*) corresponde al intento por extraer una representación más acabada del significado de los textos. NLP usa típicamente conceptos lingüísticos tales como *part-of-speech* (*POS tagging*) y estructuras gramaticales además de su intento por tratar con distintos elementos gramaticales como anáforas y ambigüedades. Para realizar esto se usan distintas representaciones y softwares como *lexicons* de palabras que contienen sus significados y sus propiedades gramaticales, además de conjuntos de reglas gramaticales y usualmente otros recursos tales como la ontología de entidades y diccionarios de sinónimos y abreviaciones.

Las opiniones encapsulan un gran número de técnicas que forman parte del *Text Mining*, la que a su vez también es un sub-campo de la minería de datos. La principal diferencia entre el *text mining* y *data mining* es que en general el tipo de datos utilizados para el primero tipo son documentos compuestos por caracteres alfanuméricos, y datos no estructurados, mientras tanto el segundo son datos estructurados obtenidos de base de datos.

Existen varias aplicaciones para la minería de textos, algunos son los siguientes:

- Extracción de información.
- Análisis de sentimientos.
- Clasificación de documentos.
- Elaboración de resúmenes.
- Extracción de conocimiento.

5.4.1 Data Mining

Como es mencionado en los datos están siendo recolectados a una tasa cada vez mayor. Esta tarea de convertir los datos en conocimiento antes era parte de un proceso manual de análisis e interpretación la cual es muy demandante en tiempo, dinero y es altamente subjetiva. A medida que los volúmenes de data crecen, este tipo de metodologías se vuelven completamente inútiles en muchos campos.

Este hecho tiene como consecuencia la urgencia de una nueva generación de teoría computacional y herramientas que ayuden a las personas a extraer información útil de estos crecientes volúmenes de data digital. Estas herramientas pertenecen a un campo conocido como Descubrimiento de Conocimiento desde las Bases de Datos (KDD) que permite descubrir patrones relaciones y formular modelos de grandes sets de datos utilizando métodos de inteligencia artificial, *machine learning*, estadísticas y sistemas de bases de datos, estas técnicas han adquirido enorme importancia en áreas como marketing, soporte de decisiones, planeamiento financiero, análisis de datos financieros, análisis de datos científicos, bioinformática, análisis de texto y de datos web.

El principal objetivo de este proceso es la extracción y descubrimiento de información desde un conjunto de datos no trivial, desconocida y potencialmente útil en grandes repositorios de datos, transformarlos en una estructura entendible para su uso posterior. Las principales tarea y métodos de este proceso son la clasificación, agrupamiento, estimación, modelado de dependencias y descubrimiento de reglas.

5.5 Minería de Opiniones.

Los estudios y softwares de detección de sentimientos y opiniones han sido testigos de un creciente interés en los últimos años debido a la creciente cantidad de reseñas y opiniones online y la necesidad de organizarlos. En relación a este tema existen 4 problemas predominantes para los investigadores del área: clasificación de la subjetividad de documentos, clasificación de sentimientos de una palabra, clasificación de sentimientos de un documento, y la extracción de opiniones. Es un hecho que existen relaciones entre las tareas anteriormente mencionadas como por ejemplo que un clasificador de subjetividad puede prevenir al clasificador de sentimientos considerar textos irrelevantes para su clasificación del punto de vista objetivo.

6. Desarrollo Del Proyecto

La fase inicial de nuestro sistema consta de 2 etapas, la etapa de entrenamiento y la de pruebas a continuación describiremos cada una de las etapas, como se observa en la Figura 6.

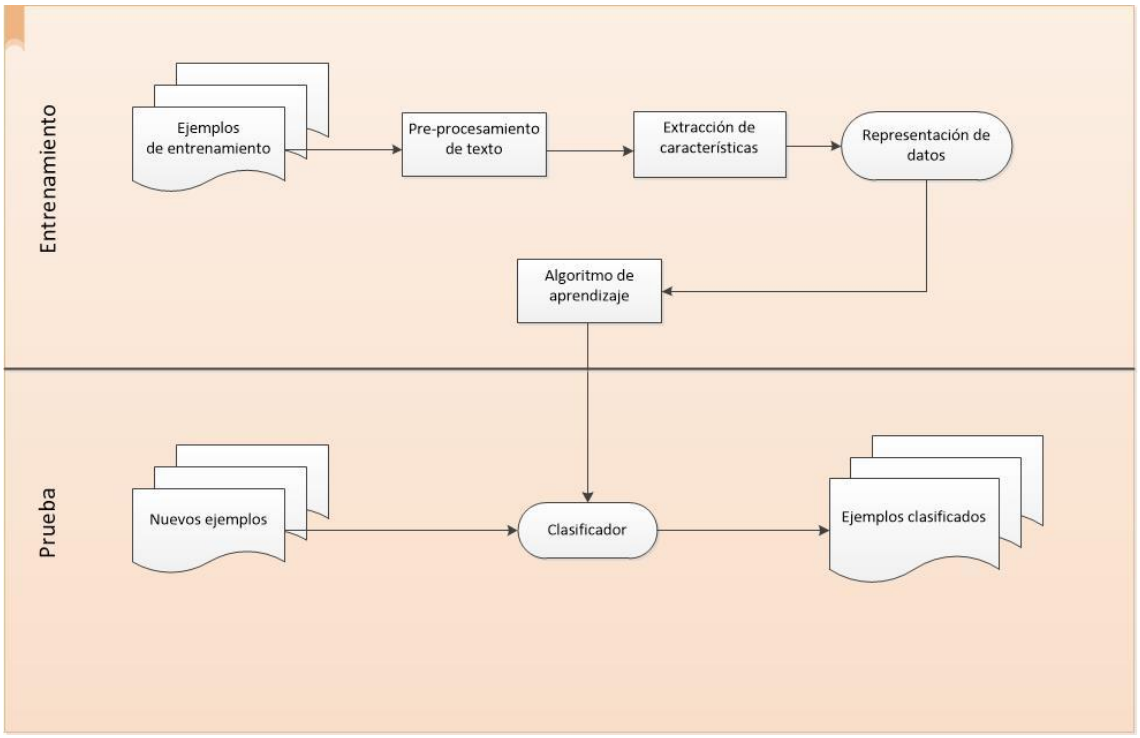

Figura 6. Etapas de desarrollo del proyecto.

6.1 Etapa de Entrenamiento.

La etapa de entrenamiento se basó en el análisis, extracción, procesamiento y clasificación de textos.

Para llevar a cabo la correcta implementación se utilizó el lenguaje de programación Java haciendo uso de NetBeans y la biblioteca de análisis WEKA como ambiente de trabajo además, se utilizó el modelo de bolsa de palabra, los métodos de vectores de peso de frecuencia de aparición y el booleano.

Se clasificó los productos con estos dos métodos elementales, donde se realizó la representación de los textos, mediante la presencia o ausencia de las palabras que forman el vocabulario de las clases para obtener la clasificación de positivo, negativo o neutro. Con el primer método es posible examinar la frecuencia de aparición de los términos mayor, es decir la suma de todas las ocurrencias que aparecen en un documento (TF/IDF), el segundo se analizó la información donde el modelo es binario para determinar la presencia o ausencia de los términos de los textos y existe un valor de precisión de los términos, no considera el peso específico en el contexto.

Para llevar a cabo la extracción de las características de los datos, es uno de lo más complejos ya que en base a estos se entrenan los algoritmos que posteriormente se realizará las clasificaciones.

6.1.1 Pre-procesamiento del texto

En esta etapa se inicia con el pre-procesamiento de los textos en español de la red social *twitter*, una vez que se ha llevado a cabo la lectura de estos textos, se procedió a una lematización sintáctica, consiste en etiquetar cada parte del tweet es un etiquetado morfológico de oraciones el cual se lleva a cabo en hallar el lema correspondiente, clasificando las secciones morfológicas: palabra, número, puntuación.

Además se elaboro el reconocimiento de entidades, el cual se encarga de examinar los elementos de un texto en categoría predefinidas: sustantivos, adjetivos y verbos, obteniendo un diccionario de los mismos con todas las palabras de los tweet.

Parte de esta etapa se hizo el cambio de codificación de los documentos.tsv a la codificación utf-8, donde se encuentra el texto de todos los tweet, para su lectura de carácter Unicode e ISO 10646, estos documentos.tsv se localizan en un directorio.

Después se realizo la extracción de características (palabras claves) mediante etiquetado morfo-sintáctico para reducir la dimensión de caracteres especiales y así poder eliminar información no relevante en el análisis y poder estudiar aquellos aspectos que contribuyen significativamente a su detención.

6.1.2 Extracción de características

En esta etapa de Extracción de características se apoya y hace uso de la etapa Pre-procesamiento de texto para el análisis de un texto limpio, después de tener la lectura sin un carácter Unicode se realizo la extracción del el texto y la polaridad de cada tweet, con algunas restricciones por cada texto y polaridad detectado se extrae de lo contrario se ignora , también se eliminó todos los link (http:...) que contenía el texto extraído.

Después de esto se remplazó las vocales con acentos o diéresis a pasar sin acentos y eliminamos caracteres y signos, por mencionar algunos como símbolos, aritméticos, puntos, dos puntos, punto y coma, guion, arroba, barra, guion bajo, guion medio, comillas, signos de exclamación e interrogación, entre otros por un espacio en blanco en cada tweet.

Posteriormente después de tener el texto y la polaridad en un String, tuvimos en cuenta la escritura en mayúsculas. Por ello, se paso el texto y polaridad a minúsculas guardándolo

en un fichero.txt como se muestra en la siguiente Figura 7 y en un método guardar de la clase vector, que se seguirá utilizando en el siguiente paso.

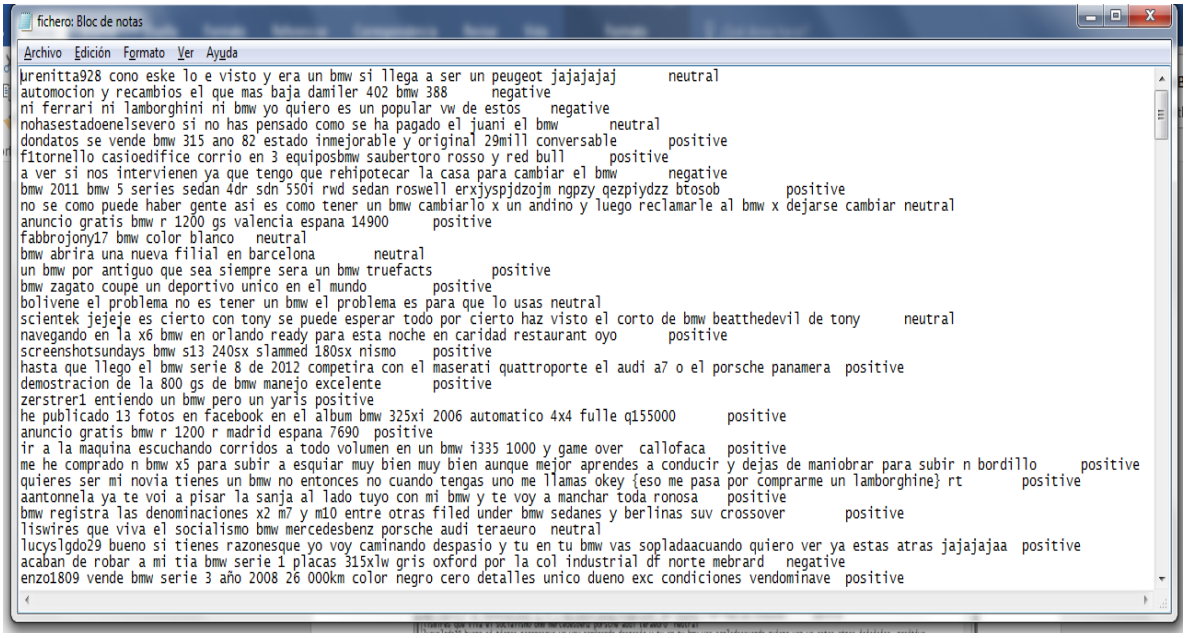

Figura 7. Fichero con Texto y Polaridad.

6.1.3 Representación de Datos.

Para poder realizar la representación de los datos se realizo lo siguiente:

Después de tener el texto y la polaridad en el método guardar de la clase Vector, estos datos se guardaron en una objeto dato de tipo Tweet que es un Arralist con un objeto vector, esta variable tiene los string texto y polaridad.

Posteriormente se recorrió el vector con la ayuda de un ciclo, este proceso de encargó de pasar todos los tweet, así como pasaban los tweet se convirtió cada palabra en token, después cada palabra se insertó en una colección de hashset, es decir cada palabra que se introduce al hashset no se vuelve a repetir 2 veces, esta hashset se convirtió en mi vocabulario.

Entonces se comparo cada token del vocabulario con cada token de cada tweet hasta recorrer todo el tweet, este proceso se dio que el primer token del vocabulario pasa por todo el primer tweet encontrando confidencias del mismo hasta recorrer todo el tweet, después pasaba por el siguiente token del vocabulario y hace lo mismo recorre todo el tweet, después por el siguiente token del vocabulario y así sucesivamente hasta pasar con todos los token del vocabulario compararlos con el primer tweet, después de esto pasan por el siguiente tweet, es decir el segundo tweet y así hasta compararlo con todo los tweet esto nos dio como resultado el número de veces que se repetía cada palabra

en cada tweet como se muestra en la Figura 8, los datos fueron guardados en documento.csv como se muestra en la Figura 9.

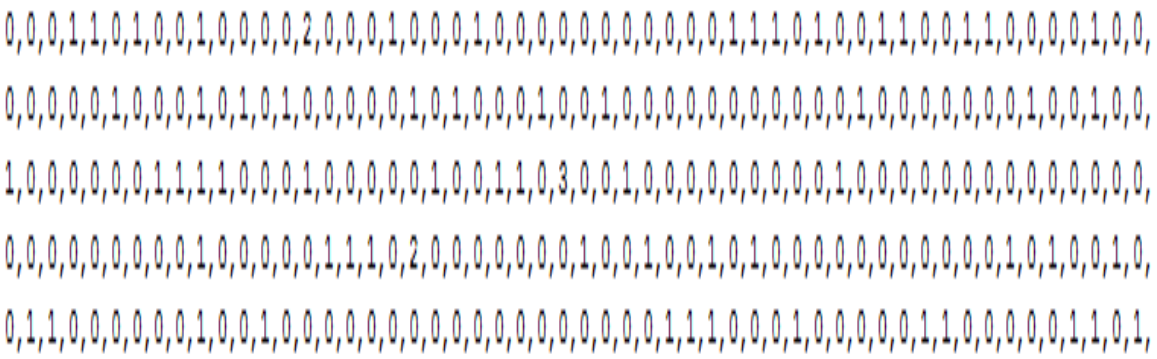

Figura 8. Tabla de comparación con la consola de NetBeans IDE 8.1.

que	cono	lo	damiler	jajajajaj	bmw	era	si	peugeot	automocion	baja	un	llega	mas	a	ser	e	eske	el	yurenita928	recambios	visto	388	402	y	CLASE
0	1	1	0	1	1	1	1	1	0	0	2	1	0	1	1	1	1	0	1	0	1	0	0	1	neutral
1	0	0	1	0	1	0	0	0	1	1	0	0	1	0	0	0	0	1	0	1	0	1	1	1	negative

Figura 9. Tabla de comparación con el número de coincidencias del vocabulario y su polaridad Archivo.csv.

La transformación de textos en características (pesos) de cada tweet. Posteriormente, estas características fueron representadas mediante un modelo de bolsa de palabras.

La extracción de características se realizó mediante dos métodos estadístico: peso binario, frecuencia de aparición de la palabra (frecuencia de término – frecuencia inversa de documento).

Estos datos fueron utilizados para entrenar una red neuronal artificial la cual se encargó de predecir la clase de nuevos textos.

6.2 Etapa de pruebas.

En esta etapa se utilizó una Red Neuronal Artificial, para la clasificación de opiniones delas redes sociales, donde se eligieron nuevos textos de tweets (ejemplos) a clasificar. Dichos textos se clasificaron con sus respetivas clases: positivo, negativo y neutro con la herramienta WEKA haciendo el uso del algoritmo de clasificación de una red neuronal MultilayerPerceptrón, como se muestras en la Figura 10.

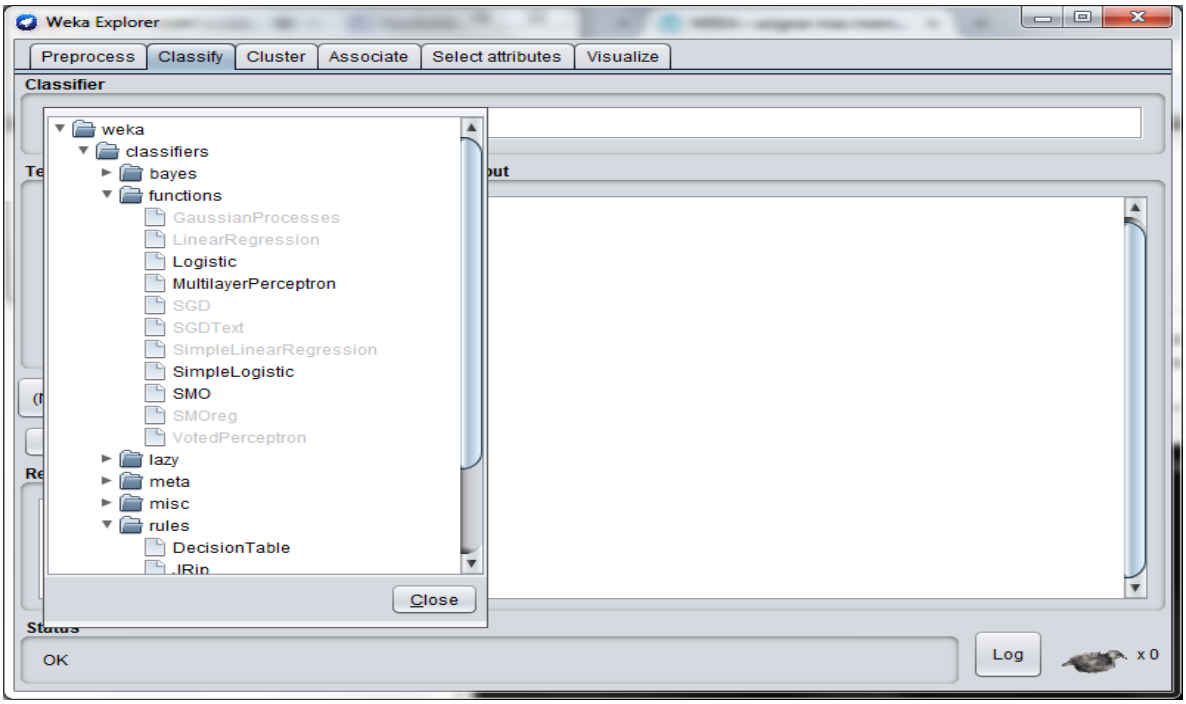

Figura 10. Red neuronal MultilayerPerceptrón.

Se utilizó el modelo de bolsa de palabra con un pesado de vectores de frecuencia de aparición y el peso booleano, con el modo **Percentage Split** con los diferentes porcentajes como son el 60%, 70%, 80% y 90% para el entrenamiento de la red neuronal, como se muestra en la Figura 11.

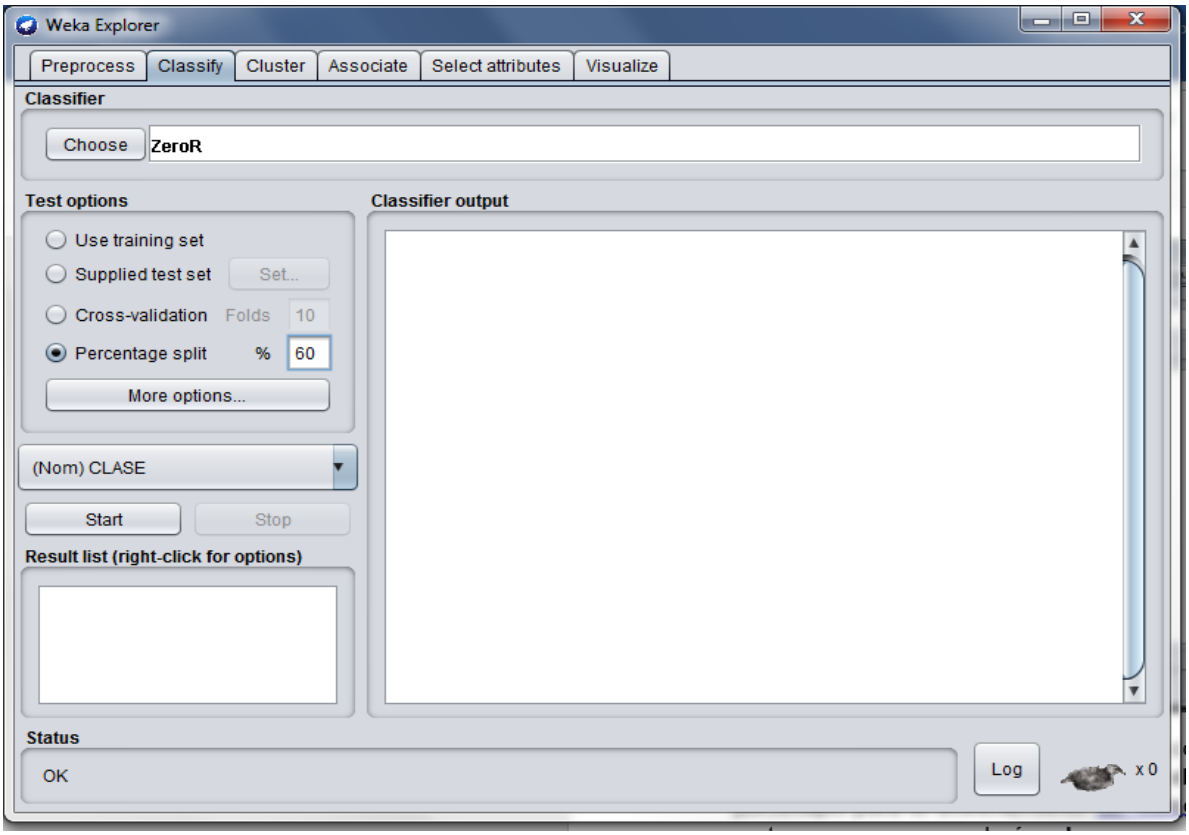

Figura 11. Modo Split.

Como resultado se obtuvo la clase a la que pertenece y se procederá a **la comprobación** de resultados de cada prueba de entrenamiento descritas anteriormente.

Entrenamiento con el 60%

A continuación, se presenta la salida de la prueba correspondiente al 60%, la cual se utilizará para la comparativa de resultados.

Instancias 6556.

Atributos 104.

Tiempo de la construcción del modelo clasificador: 327.24 segundos

Instancias correctamente clasificadas	1630	62.1663 %
Instancias incorrectamente clasificadas	992	37.8337 %

Precision detallada por clase

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.255	0.057	0.650	0.255	0.367	0.282	0.759	0.570	neutral
0.521	0.013	0.934	0.521	0.669	0.633	0.844	0.743	negative
0.921	0.593	0.556	0.921	0.694	0.372	0.765	0.693	positive

Matriz de Confusión

a	b	c	<-- classified as
197	10	565	a = neutral
29	354	296	b = negative
77	15	1079	c = positive

Donde las columnas indican las categorías clasificadas por el clasificador que son a, b y c y las filas las categorías reales de los datos neutral, negativo y positivo, por lo que los elementos en la diagonal principal son los elementos que han acertado en este caso la clase neutral 197 correctos contra 575 incorrectos, negativo 354 correctos contra 325 y positivo 1079 correctos contra 92 incorrectos.

Entrenamiento con el 70%

A continuación, se presenta la salida de la prueba correspondiente al 60%, la cual se utilizará para la comparativa de resultados.

Instancias 6556.

Atributos 104.

Tiempo de la construcción del modelo clasificador: 319.96 segundos

Instancias correctamente clasificadas	1269	64.5145 %
Instancias incorrectamente clasificadas	698	35.4855 %

Precision detallada por clase

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.300	0.023	0.848	0.300	0.443	0.411	0.759	0.597	neutral
0.519	0.016	0.915	0.519	0.662	0.623	0.848	0.738	negative

0.950 0.589 0.565 0.950 0.708 0.414 0.765 0.693 positive

Matriz de Confusión

a b c <-- classified as

178 8 407 | a = neutral

4 258 235 | b = negative

28 16 833 | c = positive

Donde las columnas indican las categorías clasificadas por el clasificador que son a, b y c y las filas las categorías reales de los datos negativo, neutral y positivo, por lo que los elementos en la diagonal principal son los elementos que han acertado en este caso la clase neutral 178 correctos contra 415 incorrectos, negativo 258 contra 239 incorrectos y positivo 833 contra 44 incorrectos.

Entrenamiento con el 80%

A continuación, se presenta la salida de la prueba correspondiente al 60%, la cual se utilizará para la comparativa de resultados.

Instancias 6556.

Atributos 104.

Tiempo de la construcción del modelo clasificador: 309.47 segundos

Instancias correctamente clasificadas 848 64.6834 %

Instancias incorrectamente clasificadas 463 35.3166 %

Precision detallada por clase.

TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class

0.300 0.023 0.848 0.300 0.443 0.411 0.759 0.597 neutral

0.519 0.016 0.915 0.519 0.662 0.623 0.848 0.738 negative

0.950 0.589 0.565 0.950 0.708 0.414 0.765 0.693 positive

Matriz de Confusión

a b c <-- classified as

151 3 245 | a = neutral

19 174 143 | b = negative

46 7 523 | c = positive

Donde las columnas indican las categorías clasificadas por el clasificador que son a, b y c y las filas las categorías reales de los datos negativo, neutral y positivo, por lo que los elementos en la diagonal principal son los elementos que han acertado en este caso la clase neutral 151 correctos contra 248 incorrectos, negativo 174 correctos contra 162 incorrectos y positivo 523 correctos contra 53 incorrectos.

Entrenamiento con el 90%

A continuación, se presenta la salida de la prueba correspondiente al 60%, la cual se utilizará para la comparativa de resultados.

Instancias 6556.

Atributos 104.

Tiempo de la construcción del modelo clasificador: 325.1 segundos

Instancias correctamente clasificadas 1269 64.7866 %

Instancias incorrectamente clasificadas 698 35.2134 %

Precision detallada por clase

TP Rate	FP Rate	**Precision**	**Recall**	**F-Measure**	MCC	ROC Area	PRC Area	Class
0.255	0.057	0.650	0.255	0.367	0.282	0.759	0.570	neutral
0.521	0.013	0.934	0.521	0.669	0.633	0.844	0.743	negative
0.921	0.593	0.556	0.921	0.694	0.372	0.765	0.693	positive

Matriz de Confusión

a b c <-- classified as

72 2 130 | a = neutral

2 87 81 | b = negative

11 5 266 | c = positive

Donde las columnas indican las categorías clasificadas por el clasificador que son a, b y c y las filas las categorías reales de los datos negativo, neutral y positivo, por lo que los elementos en la diagonal principal son los elementos que han acertado en este caso la clase neutral 72 correctos contra 132 incorrectos, negativo 87 correctos contra 83 incorrectos y positivo 266 correctos contra 16 incorrectos.

En la siguiente sección se visualiza y compara cada porcentaje para ver quién es el mejor.

7. Resultados

Estos resultados se validaron con un conjunto de tweets que han sido clasificados de manera manual por expertos con la finalidad de evaluar la eficiencia de la Red Neuronal Artificial implementada y en un futuro, acortar muchos pasos de la minería de opiniones.

Como se visualiza en Figura 12 se muestran los resultados de las 4 corridas ejecutas en el programa WEKA.

	60%	70%	80%	90%
Instancias correctamente	62.1663 %	64.5145 %	64.6834 %	64.7866 %
Instancias incorrectamente	37.8337 %	35.4855 %	35.3166 %	35.2134 %
Tiempo transcurrido	327.24 segundos	319.96 segundos	309.47 segundos	325.1 segundos

Figura 12. Tabla de Resultados.

Nos proporciona varios datos esta tabla, tanto el mejor porcentaje como el peor, así mismo con el tiempo de ejecución de cada porcentaje.

Unos de los datos destacables tras realizar el entrenamiento de la red neuronal se observa que el mejor porcentaje es el de 90% de mayores instancias correctas con el 64.7866% y un tiempo de ejecución de 325.1 segundos.

Aunque los resultados son un poco inferiores con el de 80% en instancias correctas, es más rápido en tiempo de ejecución, con un tiempo de 309.47.

El peor sería el del 60% tanto como en el tiempo de ejecución de 327.24 segundos como en instancias correctas de 37.8337%.

Los resultados obtenidos nos da a entender que al realizar el proceso de entrenamiento de una red MultilayerPerceptrón en WEKA, mientras el numero de instancias sea mayor, este algoritmo es suficiente para estimar con precisión y predecir las clases de los tweet.

8. Conclusiones

La tecnología de las ANN han demostrado ser una forma muy poderosa de obtener buenas y confiables resultados, basados en complejos algoritmos matemáticos, capaces de entregarnos la mas cercana información respecto de un proyecto que puede significar múltiples alternativas y opciones, eliminando los algoritmos tradicionales, adquiriendo en la actualidad como evidencia el hecho de formar parte de los estudios centrales de instituciones gubernamentales a nivel mundial.

Como hemos podido apreciar en el presente trabajo, existe varias aplicaciones que explotan las características de la Red Neuronal Artificial donde se requiere la solución de problemas, unas de las cuales se aplican en el campo de recuperación de la información. Las nuevas tecnologías han crecido de forma increíble en los últimos años, la web se ha convertido en una inmensa red de información el cual es imposible de analizar todos los datos que aparecen en ella, una de la información más importante que se encuentra hoy en día en la web son las opiniones de los consumidores sobre productos y servicios de diversas marcas, el cual se puede analizar las opiniones con la ayuda de la red neuronal artificial utilizando un Perceptrón multicapa como clasificador y así auxiliar a las empresas, escuelas, instituciones gubernamentales entre otros a detectar las tendencias del mercado.

A partir del uso y resultados obtenidos se observa que mientras que el número de porcentaje de entrenamiento sea elevando, la red neuronal presenta mayor predicción, el aprendizaje de una ANN se basa en la capacidad de realizar tareas basadas en un entrenamiento sin necesidad de introducir todo los parámetros si no con los mas destacados, esto reduce el tiempo de entrenamiento creando su propia organización, ya que normalmente entrenar la red neuronal se llevarían horas, días o meses, según el conjunto de atributos que le demos como entrada, es decir la red es entrenada con información de primer nivel debido a que los resultados generan una mejor interpretación de los datos y llegan a ser una toma de decisión de carácter muy importante, tales como : la correcta predicción de un conjuntos de patrones en el mercado, transacciones en la bolsa, entrenamiento a seres no orgánicos, proceso de refinería, petroquímica, pronósticos de ventas, administración de riesgo o como en nuestro caso la clasificación de opiniones de tweet.

Por último cabe destacar que las ANN es un campo muy interesante, adquiriendo en la actualidad un gran avance en la identificaciones de patrones, esto nos ayuda para trabajos a futuros en reconocimiento de imagen y sonido, están emergiendo chips basado en redes neuronales como también aplicaciones para la resolución de problemas complejos, el presente es un periodo de transición para la tecnología de las redes neuronales, o en la vida cotidiana en la extracción y clasificación de información, algunas de las predicciones con el uso de la redes neuronales son: Interfaces Maquina-Hombre,

Integración de la Inteligencia Artificial con la vida orgánica, Realidades alternativas producidas por entornos sensitivos.

9. Bibliografía

1) García Rodríguez, Ramírez de la Rosa, “Detectando la prioridad de contenidos generados en Twitter por medio de n-gramas de palabra”, proyecto terminal, Universidad Autónoma Metropolitana Cuajimalpa, México, D.F., 2014.

2) S.M. de Jesús Ugalde Sánchez, “Sistema de Recuperación de Información Semántico”, Universidad Autónoma Metropolitana Azcapotzalco, México, D.F., 2011.

3) Cárdenas Cárdenas, "Inteligencia Artificial, Métodos Bio-Inspirados: Un enfoque Funcional para la Ciencia de la Computación”, Universidad Tecnológica de Pereira, Colombia 15 de diciembre 2012.

4) L. Dubian, “Procesamiento de Lenguaje Natural en Sistemas de Análisis de Sentimientos”, Universidad de Buenos Aires, Argentina, 2013.

5) Dueñas, J. De Velazquez, “Una Aplicación de Web OpinionMining para la Extracción de Tendencias y Tópicos de relevancias a partir de las Opiniones Consignadas en Blog y Sitios de Noticias”, Revista Ingeniería de Sistemas, Volumen XXVII, Septiembre 2013.

6) **WordStat** https://provalisresearch.com/products/content-analysis-software/

7) J. Damian Matich, “Redes neuronales: Conceptos básicos y Aplicaciones”, Universidad Tecnológica Nacional-Facultad Región Rosario, Argentina Marzo 2001.

8) X. Basogain Olabe, “Redes Neuronales Artificiales y sus aplicaciones”, Escuela Superior de Ingeniería de Bilbao, EHU
http://www.ciberesquina.una.edu.ve:8080/2014_2/350_E.pdf

9) Corso, C.L., “Aplicaciones de Algoritmos de clasificación supervisada usando WEKA”, Universidad Tecnologica Nacional, Facultad Regional Córdoba, http://www.investigacion.frc.utn.edu.ar/labsis/Publicaciones/congresos_labsis/cynthia/CNIT_2009_Aplicacion_Algoritmos_Weka.pdf

10. Anexos

La Clase Principal del Programa donde mandas a llamar a otras clases instanciadas como AbrirArchivo, Vector, Escritura

10.1 Clase Principal

```
/*

*

*/

package principal;


import clases.Tweet;

import java.util.ArrayList;

import operaciones.AbrirArchivo;

import operaciones.Escritura;

import operaciones.Vector;


/**

*

* @author Victor Hugo Rojas Luis

*/

public class Principal {


/**

* @param args the command line arguments

*/

public static void main(String[] args) {
```

```
AbrirArchivo abrir= new AbrirArchivo();

Vector vector=new Vector();

Escritura e = new Escritura();


abrir.abrir();

//vector=e.getGuardar();

vector.imprimir();

}


}
```

De la clase principal vamos a la clase de AbrirArchivo, aquí es donde el programa listara el contenido del directorio, es decir va Abrir lo documentos.tsv y esto no direccionarnos a la clase Escritura

10.2 Clase Abrir Archivo

```
/*

*

*/

package operaciones;


import java.io.File;

import java.util.HashMap;


/**

*

* @author Victor Hugo Rojas Luis

*/

public class AbrirArchivo {
```

```
public void abrir (){
File f = new File("Tweets/");
String files = null;



Vector r= new Vector();


if (f.exists()){


File[] ficheros = f.listFiles();
Escritura escritura = new Escritura();


for (int x=0;x<ficheros.length;x++){
if (ficheros[x].isFile())
files = ficheros[x].getName();
//System.out.println(ficheros[x]);
r=escritura.escritura(files);
}
// r.imprimir();
//System.out.println();
r.treeMap();
}
```

```
else {

System.out.println("El archivo no existe");


}

}

}
```

Después nos direcionamos con la clase Escritura, donde leeremos los documentos.tsv y extraeremos texto y polaridad de los archivos, los guardaremos en un archivo y un objeto de tipo Vector.

10.3 Clase Escritura

```
/*

*

*/

package operaciones;


import clases.Tweet;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.util.ArrayList;

import java.util.StringTokenizer;
```

```
/**
*
* @author
*/
public class Escritura {

Vector v = new Vector();

public Vector escritura(String nombreFichero) {

int nComillas = 0;
try (
//Leer un Arcvhivo con una iso latin 1 o lo que es lo mismo ISO-8859-1, Latin Alphabet No. 1.
//BufferedReader br=new BufferedReader(new FileReader("Tweets/"+nombreFichero) );
BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream("Tweets/"
+ nombreFichero), "ISO-8859-15"));
//Escribir en un archivo
//BufferedWriter out = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("Tweets/"+nombreFichero), "utf-8"));
BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("Tweets/fichero.txt",true), "utf-8"));)
{
// BufferedWriter bw=new BufferedWriter(new FileWriter("Tweets/fichero.txt",true));){
String linea = br.readLine();

while (linea != null) {

char[] array = linea.toCharArray();
```

```
/*---*/// System.out.println(array);
int comillas = 0;
String texto = "";
String polaridad = "";

int iTexto = 0;
int fTexto = 0;
int iPolaridad = 0;
int fPolaridad = 0;
// Aqui teien todo el texto de.tsv
//System.out.println(array);
for (int index = 0; index < array.length; index++) {
if (array[index] == "") {
comillas = comillas + 1;
}
if (comillas == 7) {
iTexto = index;
comillas = comillas + 1;
}
if (comillas == 9) {
fTexto = index;
comillas = comillas + 1;
}
if (comillas == 11) {
iPolaridad = index;
comillas = comillas + 1;
```

```
}
if (comillas == 13) {
fPolaridad = index;
comillas = comillas + 1;
}

if (comillas == 15 || comillas == 17 || comillas == 19 || comillas == 21 || comillas == 23 || comillas
== 25 || comillas == 27 || comillas == 29) {
fTexto = iPolaridad;
iPolaridad = fPolaridad;
fPolaridad = index;
comillas = comillas + 1;
}
}

if (nComillas < comillas) {
nComillas = comillas;
}

for (int index = iTexto + 1; index < fPolaridad; index++) {
if (index < fTexto) {
/*if(array[index]=='ñ'){
array[index]='n';
}*/
texto = texto + array[index];
//System.out.println(texto);cada twwet
}
```

```
if (index > iPolaridad && index < fPolaridad) {
polaridad = polaridad + array[index];
}
}
char[] vinculo = texto.toCharArray();
texto = "";
int n = 0;
/*if (n == vinculo.length - 1) {
vinculo[n]=0;*/
for (int i = 0; i < vinculo.length; i++) {

if ((i+3) < vinculo.length) {
if (vinculo[i] == 'h'&& vinculo[i + 1] == 't'&& vinculo[i + 2] == 't'&& vinculo[i + 3] == 'p') {
break;
}
}
texto = texto + vinculo[i];
// System.out.println(texto);
}

String lineaPolaridad = null;

if (texto.equals("text") || texto.equals("") || polaridad.equals("polarity") || polaridad.equals("") ||
texto.equals("http:")) {
;
} else {
```

```
texto = texto.replaceAll("á", "a");
texto = texto.replaceAll("é", "e");
texto = texto.replaceAll("í", "i");
texto = texto.replaceAll("ó", "o");
texto = texto.replaceAll("ú", "u");
texto = texto.replaceAll("ü", "u");
texto = texto.replaceAll("ñ", "n");
texto = texto.replaceAll("Á", "A");
texto = texto.replaceAll("É", "E");
texto = texto.replaceAll("Í", "I");
texto = texto.replaceAll("Ó", "O");
texto = texto.replaceAll("Ú", "U");
texto = texto.replaceAll("Ü", "U");
texto = texto.replaceAll(",", "");
texto = texto.replaceAll("\\”", "");
texto = texto.replaceAll("@", "");
texto = texto.replaceAll("\\|", "");
texto = texto.replaceAll("\\$", "");
texto = texto.replaceAll("!", "");
texto = texto.replaceAll("\\?", "");
texto = texto.replaceAll("\\¿", "");
texto = texto.replaceAll("\\¡", "");
texto = texto.replaceAll("\\!", "");
texto = texto.replaceAll("\"", "");
texto = texto.replaceAll("\\'", "");
texto = texto.replaceAll("#", "");
```

```
texto = texto.replaceAll("%", "");

texto = texto.replaceAll(":", "");

texto = texto.replaceAll("\\(", "");

texto = texto.replaceAll("\\)", "");

texto = texto.replaceAll("\\[", "");

texto = texto.replaceAll("\\]", "");

texto = texto.replaceAll("\\º", "");

texto = texto.replaceAll("\\ª", "");

texto = texto.replaceAll("\\.", "");

texto = texto.replaceAll("/", "");//espacio

texto = texto.replaceAll("’", "");


texto = texto.replaceAll(">", "");

texto = texto.replaceAll("<", "");

texto = texto.replaceAll("©", "");

texto = texto.replaceAll("£", "");

texto = texto.replaceAll("\\Â€", "");

texto = texto.replaceAll("Â—", "");

texto = texto.replaceAll("\\-", "");//espacio

//texto = texto.replaceAll("", "");

texto = texto.replaceAll("", "");//espacio

texto = texto.replaceAll("", "");//espacio

texto = texto.replaceAll("", "");

texto = texto.replaceAll("", "");

texto = texto.replaceAll("                                        ", "");

texto = texto.replaceAll("\\_", "");
```

```
texto = texto.replaceAll("\\*", "");

texto = texto.replaceAll("\\^", "");

texto = texto.replaceAll(";", "");

texto = texto.replaceAll("&", "");

texto = texto.replaceAll("\\+", "");

texto = texto.replaceAll("\\=", "");

texto = texto.replaceAll("\\•", "");

texto = texto.replaceAll("€ ", "");

texto = texto.replaceAll("\\t", "");//tabulador

texto = texto.replaceAll("™", "");


texto = texto.toLowerCase();

lineaPolaridad = polaridad.toLowerCase();

//                bw.write("Texto:");

bw.write(texto);

//System.out.println(texto);

//System.out.println("Mayor numero de comillas\t"+ lineaTexto);

//                bw.newLine();

//                bw.write("Polaridad:");

bw.write("\t" + lineaPolaridad);

bw.newLine();

//                bw.newLine();

v.guardar(texto, lineaPolaridad);

//guardar.imprimir(); https://java-spain.com/metodos-replace-replacefirst-y-replaceall-y-expresiones-regulares

}

//bw.flush();
```

```
linea = br.readLine();

}

//System.out.println("----------\t"+linea);

bw.close();

} catch (IOException e) {

System.out.println("Error E/S: " + e);

}

return v;

}

}
```

Nos direccionamos al método guardar de la clase Vector, donde recorro el objeto v de tipo Vector para imprimir todos los tweets, convertirlos en token y después guardarlos en una colección hashset que es mi vocabulario, para compararlo con cada tweet y tener el número de repeticiones de cada token del vocabulario en cada tweet, estos son los datos de entrada para entrenar a la red neuronal.

10.4 Clase Vector

```
/*

*

*/

package operaciones;


import clases.Tweet;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.FileWriter;
```

```
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.io.Writer;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.TreeMap;

/**
*
* @author
*/
public class Vector {

private ArrayList<Tweet> vector = new ArrayList<>();
private ArrayList<Integer> tf = new ArrayList<Integer>();
List<String> list = new ArrayList<String>();

// Pasa el ArrayList a un Array
//Object[] array = nombreArrayList.toArray();
public void guardar(String texto, String polaridad) {
Tweet datos = new Tweet();
datos.setTexto(texto);
```

```
//System.out.println(texto + "\t\tcadena texto" ); /*tienes el texto de cada twwet*/

datos.setPolaridad(polaridad);

this.vector.add(datos);

//System.out.println("------------------TAMAÑO----->" +vector.size());

}


public void treeMap() {

// https://www.dotnetperls.com/hashset-java

int i = 0;

Iterator<Tweet> itrTweet = vector.iterator();

HashSet<String> vocabulario = new HashSet<String>();

/*HashSet<String> vocabulario1 = new HashSet<String>();

List<String> list = new ArrayList<String>();*/

Tweet tweet = new Tweet();

int p=0;

while (itrTweet.hasNext()) {

tweet = itrTweet.next();

p++;

System.out.println(p+tweet.getTexto()+"\n");//imprime cada tweet

//System.out.println(tweet.getPolaridad());

String[] dividirArray = tweet.getTexto().split("");

/*for(int a=0; a<dividirArray.length;i++){

System.out.println("El token en la posicion "+ i+" es:" + dividirArray[a]);

}*/

for (String DivideToken : dividirArray) {

//System.out.println("El token en la posicion "+ i+" es:" + DivideToken);//imprime la posicion de
cada twwet
```

```
/*if (!DivideToken.startsWith(",")) {
vocabulario.add(DivideToken);
}
*/
if(!DivideToken.equals(""))
vocabulario.add(DivideToken);
}
i++;//Se aumenta por cada tweet
// System.out.println("\t"+vocabulario.size()); //Se aumenta el tamaño del hashset sin repetir
}
System.out.println("\t"+vocabulario.size()); //Se aumenta el tamaño del hashset sin repetir
//System.out.println(vocabulario);//Tamaño total del hashset y todo el vocabulario
//String cadena1 = "";
for (String VocabularioSinRepeticiones : vocabulario) {
//VocabularioSinRepeticiones.replaceAll(",", "");
//System.out.println("--"+VocabularioSinRepeticiones);
//cadena1 = VocabularioSinRepeticiones.replaceAll(",", "");
//System.out.println(cadena1);//Muestra el diccionario*/
}
// System.out.println("\t"+vocabulario.size()); //Se aumenta el tamaño del hashset sin repetir
int[] numero = new int[10000000];
itrTweet = vector.iterator();
int contador = 0, contador1 = 0;
//String ruta =
"C:\\Users\\Profesor\\Downloads\\Cremas21\\Tweets\\fichero_TodoLosTweets.csv";//csv
String ruta ="/home/siisa/Descargas/Cremas22/Tweets/fichero_194tweet_50Archivo.csv";
String cadena = "", str = "";
```

```
File ArchivoTweet = new File(ruta);

try {

//BufferedWriter bw=new BufferedWriter(new FileWriter(ArchivoTweet));

BufferedWriter out = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream(ArchivoTweet), "UTF-8"));

//String cadena2 = "";

for (String VocabularioSinRepeticiones : vocabulario) {

//VocabularioSinRepeticiones = VocabularioSinRepeticiones.replaceAll("\\", "");

/*cadena2 = VocabularioSinRepeticiones.replaceAll(",", "");

out.write("" + cadena2 + ",");//comas de las palabras*/

out.write(""+ VocabularioSinRepeticiones+",");

}

out.write("CLASE");

out.newLine();

p=0;

while (itrTweet.hasNext()) {


int x = 0;

i++;

tweet = itrTweet.next();

p++;

System.out.println(p+tweet.getTexto()+"\n");

String[] PrimerTweet = tweet.getTexto().split("");//llega el primer tweet

//aqui devo hacer la comparación del vocabulario con el primer tweet

//String ruta="/media/cremas/ADATA UFD/PT_Version/Cremas14/Tweets/fichero3.arff.txt";

contador1 = 0;

for (String VocabularioSinRepeticiones : vocabulario) {
```

```
contador = 0;
for (String DivideToken : PrimerTweet) {
if (VocabularioSinRepeticiones.equals(DivideToken)) {
contador++;
}
}//if(contador>0){

//System.out.print("" + contador + ",");
str += contador + ",";//comas de los numeros
numero[contador1++] = contador;

}
//System.out.println(tweet.getPolaridad());
//str=str.substring(0, str.length()-1);
out.write(str);
out.write(tweet.getPolaridad());
str = "";
out.newLine();
//System.out.println();
}
out.close();
} catch (Exception e) {
//    System.out.println(ruta);
}
//System.out.println(vocabulario);
}
```

```
public void set() {


}


public void imprimir() {
//System.out.println("Vector");
Map<String, String> nomMap = new HashMap<String, String>();
Iterator<Tweet> itrTweet = vector.iterator();
while (itrTweet.hasNext()) {
Tweet tweet = itrTweet.next();
nomMap.put(tweet.getPolaridad(), tweet.getTexto());
for (Iterator<String> it = nomMap.keySet().iterator(); it.hasNext();) {
while (it.hasNext()) {
String v = it.next();
// System.out.print(v);
// System.out.println(tweet.getTexto() + ""
//   + tweet.getPolaridad());


}
}


}
}


public ArrayList<Tweet> getVector() {
```

```
return vector;
}

public void setVector(ArrayList<Tweet> vector) {
this.vector = vector;
}

}

class NumeroDeRepeticiones {

private String palabraVocabulario;
private int x = 0;

public NumeroDeRepeticiones(String palabraVocabulario) {
this.palabraVocabulario = palabraVocabulario;
}

public String getPalabraVocabulario() {
return palabraVocabulario;
}

public void setPalabraVocabulario(String palabraVocaulario) {
this.palabraVocabulario = palabraVocaulario;
}
```

```
public int getX() {
return x;
}

public void setX(int x) {
this.x = x;
}

public void suma() {
this.x++;
}
}
|
```