SEP **SES** **TMN**

INSTITUTO TECNOLÓGICO DE CHIHUAHUA II

“CREACIÓN E INTEGRACIÓN DEL MÓDULO PARA LA DETECCIÓN DE POLARIDAD DE OPINIONES”

TESIS

PARA OBTENER EL GRADO DE

MAESTRÍA EN SISTEMAS COMPUTACIONALES

PRESENTA:

EMANUELLE PORTILLO TORRES

DIRECTOR DE TESIS
M.I.S.C. JESÚS ARTURO ALVARADO GRANADINO

CO-DIRECTOR DE TESIS
DR. HERNÁN DE LA GARZA GUTIÉRREZ

CHIHUAHUA, CHIH., MÉXICO, A DICIEMBRE 2018.

Dictamen

Chihuahua, Chih., 14 de diciembre del 2018

LIC. OLGA REBECA CASTILLO CRUZ
JEFA DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
Presente.-

Por medio de este conducto el comité tutorial revisor de la tesis para obtención de grado de Maestro en Sistemas Computacionales, que lleva por nombre **"CREACIÓN E INTEGRACIÓN DEL MÓDULO PARA LA DETECCIÓN DE POLARIDAD DE OPINIONES"**, que presenta el (la) **C. EMANUELLE PORTILLO TORRES**, hace de su conocimiento que después de ser revisado ha dictaminado la **APROBACIÓN** del mismo.

Sin otro particular de momento, queda de Usted.

Atentamente

La Comisión de Revisión de Tesis.

A. Alvarado G.

M.I.S.C. JESÚS ARTURO ALVARADO GRANADINO
Director de Tesis

DR. HERNÁN DE LA GARZA GUTIÉRREZ
Co-Director

M.C. ARTURO LEGARDA SÁENZ
Revisor

M.C. BLANCA MARICELA IBARRA MURRIETA
Revisor

SECRETARIA DE
EDUCACION PUBLICA
INSTITUTO TECNOLOGICO
DE CHIHUAHUA II
DIVISION DE ESTUDIOS DE
POSGRADO E INVESTIGACION

RESUMEN

El Análisis de Sentimientos es una creciente rama de la inteligencia artificial que es debido a dos aspectos; el primero de ellos es que las empresas le están dando mayor importancia a la opinión de sus clientes, y el segundo aspecto es el creciente uso de las redes sociales. Debido a lo anterior el procesamiento manual de las opiniones de los clientes en sus redes sociales sería una tarea demasiado tardada, costosa y poco eficiente.

Para analizar por medio de una computadora un comentario existen diferentes técnicas; como el análisis por semántica, o también mediante algoritmos de aprendizaje como lo son Máquinas de Vectores de Soporte (Support Vector Machines, SVM) y las redes neuronales.

Uno de los algoritmos de redes neuronales más utilizados para el análisis de texto cuando no se tiene una base de datos etiquetada, es el llamado Mapa Auto-organizado con Crecimiento Jerárquico (Growing Hierarchical Self-Organizing Map, GHSOM), que tiene como objetivo el poder agrupar un conjunto de texto, sin tener textos de prueba previamente analizados por alguna persona.

De este modo el proyecto conjuntará una red social gigante en opiniones cortas (Twitter) con algoritmos de aprendizaje no supervisado para poder analizar lo que un grupo de personas opina sobre un producto, servicio y/o persona.

ABSTRACT

The sentiment analysis is a growing branch of artificial intelligence is due to two aspects; the first of them is that companies are giving greater importance to the opinion of their customers, and the second aspect is the growing use of social networks. Due to the above, the manual processing of customer opinions in their social networks would be a task too late, costly and inefficient.

In order to analyze a comment by means of a computer there are different techniques; such as semantic analysis, or also by means of learning algorithms such as Support Vector Machines (SVM) and neural networks.

One of the neural network algorithms most commonly used for text analysis when you don't have a labeled database is the so-called Growing Hierarchical Self-Organizing Map (GHSOM), which aims to group a set of text, without having test texts previously analyzed by someone.

In this way the project will combine a giant social network in short opinions (Twitter) with unsupervised learning algorithms to analyze what a group of people say about a product, service and/or person.

DEDICATORIA

Esta tesis se la dedico a mis padres, quiénes siempre me han apoyado a seguir adelante en mis estudios, además de apoyarme incondicionalmente para que logre mis metas. Por todo el cariño y los abrazos que me dieron en los momentos difíciles para que pueda dar este paso y acercarme más a mi meta.

También se la dedico a mis sobrinos que siempre han sido mis grandes motores para llegar a mi meta y que ellos me vean como un ejemplo y estén orgullosos de su tío.

Por último y no menos importantes a mis hermanos biológicos y políticos que me han ayudado en cada pequeño avance que daba, día a día dándome su apoyo y palabras de aliento.

AGRADECIMIENTOS

El agradecimiento de mi tesis es a los catedráticos del Instituto Tecnológico de Chihuahua II que me han brindado su conocimiento y apoyo para poder concluir este proyecto y cerrar este ciclo en mi vida.

También en agradecimiento a mi profesor de informática de nivel bachillerato, quién no sólo me enseñó el bello arte de la programación, sino que también marcó mucho mi personalidad, dándome también la mayor razón para llegar hasta dónde estoy, y además me enseñó a compartir el poco o mucho conocimiento que tiene uno, a todo aquél que desee aprender.

Contenido

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1. INTRODUCCIÓN

1.1 Introducción.

Anteriormente en las empresas era muy común tener físicamente un buzón de quejas y sugerencias, sin embargo, con el advenimiento de la tecnología digital, éstas han optado por utilizar una página web para esos fines. El hecho de que se trate de una página web hace más sencillo el que los clientes hagan una queja o dejen una sugerencia, pero esto a su vez, genera posiblemente la necesidad de que tengan que almacenar en sus bases de datos una considerable cantidad de información, lo que trae como consecuencia el que las empresas tengan que invertir en recursos humanos dedicados a procesar dicha información, trayendo como consecuencia que el proceso sea tardado, costoso y poco eficiente.

Una manera de mejorar la eficiencia de este proceso, es el de hacer uso de las tecnologías en el manejo de la información, las cuales proporcionan diversas opciones para el procesamiento automático de grandes cantidades de información. Una de las ventajas de aplicar este tipo de tecnologías es el de:

- Conseguir tendencias de un grupo de usuarios y
- Poder extraer cierta información sobre la opinión u opiniones acerca de una persona, un objeto o un servicio.

Es por esto que el presente trabajo propone el desarrollo de un software especializado en el análisis de la información, que al procesar tuits escritos en español sea capaz de poder determinar si expresan una opinión positiva, negativa o neutra acerca de un bien o servicio.

1.2 Problema de la investigación.

Cuando se tiene que procesar una cantidad tan numerosa de información se vuelve demasiado complejo el que las personas tengan que procesar dicha información de manera manual y por tanto se requiere agilizar este proceso. Una manera de lograrlo es a través de la aplicación de las Tecnologías de la Información; específicamente mediante el uso de una técnica de Minería de Datos llamada *Análisis de Sentimientos* (proceso de determinar el tono emocional que hay

detrás de una serie de palabras) que forma parte del Procesamiento del Lenguaje Natural que a su vez es un campo de la Inteligencia Artificial. Lamentablemente, para el idioma español no hay técnicas suficientemente efectivas para el procesamiento del lenguaje; esto debido a que en nuestro idioma una misma palabra puede tener diferentes significados (palabras homónimas), además existen otras dificultades emanadas del hecho de que los usuarios dejan sus quejas y sugerencias en una red social, como son:

- Que escriban frecuentemente con faltas de ortografía.
- Que escriban palabras que suenan parecidas, pero tienen significado diferentes (homófonos), y
- En el peor de los casos, que escriban palabras que no existen.

Por lo tanto, para atacar estas problemáticas se necesita de un software de aprendizaje automático, con técnicas de procesamiento de texto que ayudarán a la clasificación (positiva negativa o neutra) de comentarios que servirán en una etapa inicial como un primer filtro, para que posteriormente una persona verifique la correcta clasificación. Posteriormente al ir acumulando registros de estas clasificaciones se puede entrenar mejor al sistema.

1.3 Alcances y Limitaciones.

1.3.1. Alcances.

El software a desarrollar tendrá la capacidad de clasificar la polaridad de un conjunto de opiniones (reconocer si un comentario fue positivo, negativo o neutro), los cuales se obtendrán a través de Twitter en los que se menciona o contesta a un perfil en específico de la empresa. Para tener una mejor visualización de la información, la aplicación tendrá una interfaz en la cual se podrán ver las estadísticas de un cliente, y en la misma se mostrará cada tuit individual en el que fue mencionado y la polaridad con la que el sistema lo clasificó, de este modo, por medio de botones se le da la oportunidad al administrador de seleccionar la polaridad y de esta manera el sistema ofrezca un cierto tipo de retroalimentación.

1.3.2. Limitaciones.

Durante el desarrollo del software se pueden presentar algunos inconvenientes: Uno de ellos es la posible falta de tiempo, debido a que el 100 % del tiempo en la empresa lo dedico a las actividades para las que fui contratado y por lo tanto, el desarrollo total del proyecto será en mi tiempo libre y solo se dedicará un breve tiempo para revisiones en la empresa de parte de mi jefe inmediato, por lo tanto la página Web posiblemente no quede terminada al 100 %.

Otra limitante es que el equipo utilizado tanto en el desarrollo y en las pruebas no es lo suficientemente potente y los resultados pueden tardar mucho en generarse.

1.4 Justificación.

Es muy común que entre las empresas se pueda destacar el uso óptimo de sus recursos, llámese financieros, capital humano o tiempo. Con base en esa premisa, se puede mejorar el uso del capital humano al ahorrar tiempo que se invierte realizando análisis de la información a través del uso de un software encargado de leer y clasificar los comentarios que los usuarios han dejado en la red social Twitter; con esto la cantidad de personal dedicado a revisar los comentarios se podrá disminuir y dejar solo a una persona encargada para el monitoreo y la revisión de los análisis, lo cual representa la parte medular del presente proyecto.

1.5 Objetivos de la investigación.

- Desarrollar un software para analizar un conjunto de opiniones, y poder determinar con una exactitud bastante alta cuál es la opinión general sobre dichos comentarios, dando así la posibilidad de examinar la información sobre un producto o servicio especificado.
- Implementar una interfaz web amigable para su uso.
- Desarrollar un módulo de retroalimentación del sistema.
- Desarrollar un sistema de guardado de la red neuronal.

Capítulo 2. ESTADO DEL ARTE

En este capítulo se habla sobre los conceptos fundamentales entorno al Análisis de Sentimientos, además de describir los principales métodos y recursos que se utilizaron como base para desarrollar el software.

2.1 Proyectos desarrollados.

2.1.1 Idioma inglés.

Los primeros intentos que se hicieron para minar las opiniones de las personas se hicieron en el idioma inglés a principios de los años 2000. Uno de ellos data del año 2003, en el cual se utilizó un método de aprendizaje automático (Machine Learning) en donde era necesaria una base de datos previamente etiquetada, dividida en dos corpus; uno el de C|net y el otro el de Amazon. Al desarrollarse esta herramienta de software se encontraron una serie problemas, como:

1. Inconsistencia al evaluar.- Se encontró el problema de que las personas que utilizaban el sistema de clasificación no entendían completamente cómo funcionaba.
2. Ambivalencia y comparación.- Se encontró el problema de que algunas personas que dejaron sus opiniones usaban palabras de connotación negativa y al final el software terminaban erróneamente dando una opinión satisfactoria.
3. Información escasa.- El sistema debía reconocer muchas palabras claves para poder evaluar correctamente cada comentario, debido a que cada uno aportaba poca información.
4. Distribución sesgada.- En ambos corpus se encontró que había una tendencia positiva en las opiniones y eso hacía que el sistema tuviera una clasificación incorrecta, al no tener la suficiente información de opiniones negativas.

Estos problemas pueden ser ocasionados al aplicar las técnicas tradicionales de aprendizaje automático como las Máquinas de Vectores de Soporte (Support Vector Machines, SVM) y las métricas comunes, como información mutua (Dave, et al., 2003).

El siguiente avance que se obtuvo fue dos años después, tras ver los problemas que se tenían al interpretar exclusivamente los textos, por tanto, se optó por agregar la funcionalidad de reconocer los emoticones[1]. Para analizar esta información se utilizaron dos clasificadores para realizar una comparativa, los cuáles una es utilizando Naïve Bayes y la otra utilizando una SVM. En este proyecto se usó una base de datos con un subconjunto de noticias obtenidas de *Newswire*, el cual contiene una variada combinación de temas (Read, 2005).

Los sistemas desarrollados no sólo se hicieron con textos de un tema específico como los mencionados anteriormente, sino con temas variados.

En contra parte, una de las primeras investigaciones que se tiene registro en la que no se especificaba un tema, hace uso de la página de Twitter para alimentar su base de datos; para el propósito de este se utilizaron tres diferentes algoritmos, Naïve Bayes, Máxima entropía y SVM para ver cuál funcionaba de mejor manera. Para poder alimentar la base de datos hicieron uso de la Application Programming Interface (API) de Twitter.

Al analizar los textos se utilizó como características Unigramas, Bigramas y una combinación de Unigramas y Bigramas, además de una técnica conocida como Part Of Speech (POS), dando con mejores resultados la combinación Unigramas y Bigramas con el algoritmo de Máxima Entropía y por último también se observó que utilizar las características con POS, no aportaba un beneficio notable (Go, et al., 2009).

Un año después se realizó una investigación en la que se mostró que, con las características correctas, el método de POS puede ser funcional, esto lo lograron con un clasificador polinomial Naïve Bayes que usa como características POS-tags y N-gramas (Pak & Paroubek, 2010).

2.1.2 Idioma Español.

Todos los avances que se han visto hasta el momento eran en el idioma inglés, fue hasta el año 2013, en el cual se utilizaron tuits clasificados manualmente para posteriormente entrenar

[1] interpretación de una emoción por medio de caracteres, por ejemplo: :) :(:|

los algoritmos elegidos, los cuales fueron Naïve Bayes, Árbol de Decisiones y SVM, al igual que en el idioma inglés se encontraron problemas parecidos, como por ejemplo mensajes:

- Muy cortos.
- Con errores de ortografía.
- Con humor, ironía y sarcasmo. (Sidorov, et al., 2013).

2.1.2.1 Taller de Análisis de Sentimientos (TASS).

Los siguientes proyectos fueron hechos en el Taller de Análisis de Sentimientos (TASS) de la Sociedad Española para el Procesamiento del Lenguaje Natural (SEPLN); evento en el que se juntaron grupos para desarrollar aplicaciones que prueben diferentes técnicas y su efectividad en el análisis de sentimientos.

Se desarrolló un sistema con 6 etiquetas (Negativo, Negativo +, Neutro, Ninguno, Positivo y Positivo +), además de utilizar hasta 3 N-gramas, haciendo diferentes ciclos con configuraciones distintas, escrito en lenguaje Python utilizando librerías para el uso del algoritmo de SVM, tales como: *LibSVM* y *LibLinear*.

La definición de la polaridad se hizo a través de una ventana fija, por medio de experimentación sobre el conjunto de entrenamiento. El valor máximo que se ha considerado es de 7 palabras a la izquierda y a la derecha (Hurtado Oliver & Pla, 2014).

De forma paralela había un proyecto que se estuvo desarrollando con las mismas bases, salvo la implementación del algoritmo SVM, utilizando una librería incluida del software de minería de datos, llamado Weka, llegando a un resultado parecido, pero por debajo de la precisión del trabajo anterior (San Vicente Roncal & Saralegi Urizar, 2014).

En la misma competencia que los dos proyectos anteriores un equipo de investigadores hizo una aproximación por medio de distancia de Hamming, precisión basada en etiquetas y coincidencia exacta, dando otra perspectiva hasta esa fecha (Vilares, et al., 2014).

Un sistema no supervisado basado en el uso de recursos léxicos se implementó en el mismo concurso, el cual se usó POS Tagging y lematización junto con la detección de la negación.

Para el módulo de detección de polaridad se utilizó un diccionario previamente definido, entre hashtags (etiquetas usadas en la red social) y emoticones. Para la etapa de clasificación, se utilizó el diccionario que se mencionó anteriormente, de esta manera cada palabra importante tenía asignado un valor de polaridad, también tiene en consideración el uso de las negaciones, y debido a que éstas cambian el significado de la oración, éstas multiplican por -1 el valor de las siguientes palabras, de esta forma el resultado final es sumar el valor de cada palabra, y una vez hechas las operaciones correspondientes, el resultado se evalúa en una serie de condiciones para clasificarlo en la categoría correspondiente (Jiménez Zafra, et al., 2014).

El siguiente proyecto dejó los signos de puntuación, también para no tener mucho texto que procesar en el análisis cambió las URL de los tuits por una palabra reservada, siguiendo con esta misma idea se cambiaron los emoticonos por una palabra que previamente fue agregada a un diccionario, también se hizo una corrección de abreviaturas y de palabras de la mano del Corpus de Referencia del Español Actual (CREA) de la Real Academia Española, las últimas correcciones que se hacían a los textos fue de caracteres repetidos.

Al igual que en el proyecto anterior para clasificar el tuit hacen uso de un diccionario en este caso llamado: SODictionariesV1.11Spa, que tiene una orientación semántica, pero una diferencia fue que aquí se implementa la detección de intensificadores de sentimientos usando el mismo diccionario y para finalizar (al igual que en el anterior) la categoría se asigna por medio de una condición (Hernández Petlachi & Li, 2014).

2.1.2.2 Proyecto con programación estadística.

En la mayoría de los proyectos que se han mencionado anteriormente se han usado lenguajes de programación imperativa, pero a continuación se verá con profundidad un proyecto desarrollado con un lenguaje de programación estadística llamado: “R”.

Algunas recomendaciones que se hacen al empezar un desarrollo de análisis de sentimientos es que se deben de tener en cuenta las siguientes preguntas:

- ¿Qué tan largo es el conjunto de información de entrenamiento?
    - Una buena estimación sería multiplicar la cantidad de comentarios que se van a analizar con la cantidad de palabras que tiene el comentario más largo.
- ¿Cuánta memoria tiene el sistema?
    - Se debe de evitar estar leyendo archivos que ocupen más espacio en memoria del que se tiene disponible.
- ¿Cuántas aplicaciones se tienen abiertas y de qué sirven?
    - Se debe de considerar tener la menor cantidad de aplicaciones abiertas y lecturas en memoria que no sean necesarias.

Al ser un lenguaje de programación estadística, uno de los clasificadores usados fue el de Naïve Bayes. Con algoritmos no supervisados, se utiliza "Item Response Theory" (IRT), la forma en qué se calcula la asignación de categoría es por medio de la estimación de una regresión logística, uno de los problemas encontrados sobre este último algoritmo mencionado es que su precisión está sujeto a una categoría.

La forma en la que se etiquetaron los tuits para el algoritmo supervisado fue con la búsqueda de los tuits que tuvieran cierta etiqueta (mayormente se utiliza el termino en inglés "hashtag") contrarios para así poder diferenciarlos correctamente, las búsquedas que se hicieron fueron *"#prochoice"* con un 1 y *"#prolife"* con un 0, que contextualizando serían las posturas a favor del aborto y contra del mismo, respectivamente (Danneman & Heimann, 2014).

Otra técnica para seleccionar las características[2] es por medio de mínima Redundancia Máxima Relevancia (mRMR por sus siglas en inglés, minimum Redundancy Maximum Relevance), que se usa para la selección de características prominentes de una clase, la forma en la que selecciona las técnicas es de la siguiente manera:

1. Las características son seleccionadas tal que tienen una alta correlación con los atributos de la clase (Máxima Relevancia).

[2] En sistemas de aprendizaje automático se les llama características a las entradas que serían las propiedades que distinguen a una clase.

2. Las características son seleccionadas tales que sean poco redundantes y mantengan la alta correlación con los atributos de la clase (mínima Redundancia).

La técnica mRMR utiliza información mutua para medir la correlación entre características y atributos de la clase. De modo que la información mutua mide la correlación no lineal entre dos atributos.

Obtención de información o IG (por sus siglas en inglés, Information Gain) es una técnica importante usada para el análisis de sentimientos, en la cual las características seleccionadas sirven para reducir el tamaño del vector de características para obtener así mejores resultados de clasificado.

Los métodos para extracción de características más usados son los siguientes:

1. Unigramas.
2. Bigramas.
3. Bi-Tagged.
4. Características de relación.

Las técnicas de selección de características son importantes para mejorar el rendimiento de los métodos de Machine Learning en términos de precisión y tiempos de ejecución. Al eliminar las características que hacen ruido y no son de relevancia se mejora la eficiencia del clasificador. Para mejorar la eficacia es recomendable usar unigramas y bigramas, pero, esto conlleva un costo computacional alto.

Las técnicas de agrupación características (Clustering features technics) es otra técnica de selección de características el cual consiste en no desechar ninguna característica del grupo de entrenamiento, esto se hace creando clusters de características teniendo una orientación semántica similar. La ventaja de este método es que el clasificador es independiente de las (palabras/términos) elegidos por el autor.

En el análisis de sentimientos también se puede considerar hacer uso del análisis semántico mediante reglas de dependencia, que a continuación se mencionan:

1. ConceptNet.
    a. Es una larga red semántica que consiste en un gran número de conceptos de sentido común; el conocimiento es contribuido por cualquier persona en internet. Teniendo una fuente de más de 250,000 relaciones.
2. Syntactic N-Grams.
    a. Puede ser usado como características del algoritmo de Machine Learning para aprender patrones del texto para el análisis de sentimientos. Son una representación de entidades, por lo que tienen más información y menos ruido, porque contiene relaciones sintácticas entre una palabra y una oración.

Para poder hacer de uso de la ontología ConceptNet e información contextual, se usa una base de datos léxica del lenguaje inglés para expandir la ontología y está formada por un total de 117,000 sinónimos (palabras relacionadas y conceptos que pueden ser usados para inferir información útil, agrupando las palabras según su significado).

Un léxico de sentimientos es un diccionario que contiene palabras con su valor de polaridad. A continuación, se muestran tres recursos de léxico de polaridad:

- SenticNet, es un recurso disponible público para análisis de sentimientos. Es un recurso léxico construido por una agrupación del espacio modelo vectorial afectivo. Teniendo más de 14,000 conceptos con su puntaje de polaridad que va en el rango de -1.0 a +1.0, además de contener conceptos con múltiples palabras.
- SentiWordNet, es un diccionario de sentimientos que contiene un puntaje de polaridad de opinión de palabras, conteniendo alrededor de dos millones de palabras como adjetivos, adverbios, verbos, sustantivos y palabras PoS-Tagged; el clasificador está basado en una bolsa de sinónimos. Cada palabra tiene un puntaje positivo, negativo o neutro teniendo un valor entre 0.0 y 1.0, sumando un total de 1 entre las tres categorías.
- General Inquierer, fue uno de los primeros diccionarios de sentimientos, contiene 1,195 palabras positivas y 2,291 negativas.

A continuación, se describirán 5 formas distintas en las que se realizó el análisis de sentimientos para este proyecto:

- Línea base (aproximación simple con diccionarios léxicos).- En el cual se usaron los 3 léxicos anteriormente mencionados, para obtener el valor de polaridad, se extraen todas las características del documento y se suman los valores, el valor más alto obtenido es el que se asignó.
- Método basado en la ontología específica del dominio.- Se extraen las características con sujeto y combinaciones como sujeto + sujeto de los documentos; además las características extraídas se hacen coincidir con la ontología y se selecciona el dominio específico de las características importantes, luego se obtienen los valores de cada característica de los diccionarios léxicos, y al sumar los valores obtener la polaridad.
- Consideración de la importancia de las características.- Aquí se considera la importancia de las características en la ontología específica del dominio, las características que están más cercas del nodo raíz es considerado el más importante para el dominio y el que esté más alejado del mismo es considerado el menos importante.
- Léxico contextual de sentimientos.- Lleva el mismo proceso que la ontología específica del dominio, después se aplica la determinación de términos ambiguos (es un término que puede aparecer con una polaridad negativa o positiva en mayor o menor medida), al terminar se ve la determinación del término por el contexto (un término contextual es todo lo ocurrido dos frases anteriores y posteriores a la aparición de un término ambiguo) y al finalizar se suma los valores.
- Considerando información contextual y la importancia de las características.- Aquí se utiliza la información contextual y la importancia de las características en la ontología de dominios específicos. (Agarwal & Mittal, 2016)

A continuación, se muestra una tabla con los resultados obtenidos de todos los experimentos.

Tabla 2.1 Precisión de los resultados (en %).

Método	Software	Película	Restaurant
Línea base.	67.8	70.1	65.7
Ontología con dominio específico.	69.2	71.3	68.3
Importancia de las características.	72.6	71.9	71.1
Información contextual.	77.3	76.2	76.2
Información contextual e importancia de las características.	**80.1**	**78.9**	**79.4**

Capítulo 3. MARCO TEÓRICO

Para poder dar forma a este trabajo de investigación, a continuación, se definen algunos conceptos importantes.

3.1 Lenguaje Natural.

Es una de las manifestaciones de la capacidad cognitiva del ser humano en la que da curso a la exteriorización de sus pensamientos a sus semejantes. Para ello las personas se valen de una serie de convenciones fonéticas y visuales que hacen posible el entendimiento (Ayala, 2005).

3.2 Procesamiento de lenguaje natural.

Es un intento de comunicación clara entre humano-máquina y máquina-humano, dejando el uso de lenguajes de programación o de conjuntos de comandos para procesar el lenguaje humano natural (Hernández, 2012).

3.3 Tipos de procesamiento de lenguaje natural.

3.3.1. Comprensión del lenguaje natural escrito.

Se enfoca en la recepción de texto cuyo contenido es interpretado léxica, sintáctica y semánticamente en función al conocimiento que se tiene del lenguaje, del contexto y de la persona que lo expresa, además del conocimiento "ordinario" (Ayala, 2005).

3.4 Análisis de sentimientos.

Trata de clasificar cualquier documento digitalizado (archivos de texto, archivos HTML, etc.) en función de la polaridad de la opinión que expresa su autor. Esta área combina procesamiento de lenguaje natural y minería de textos, incluyendo una gran cantidad de tareas. (Martínez Cámara, et al., s.f.)

MARCO TEÓRICO

Una opinión es una cuádrupla dada de la siguiente forma *(g, s, h, t),* donde *"g"* es el objetivo, *"s"* es la opinión del objetivo, *"h"* es quién opina y *"t"* es el momento en el que se expresa. El objetivo del sentimiento es la opinión de una entidad o parte o atributo de la entidad sobre el que se ha expresado un sentimiento.

Un sentimiento es una actitud, evaluación o una emoción asociada a una opinión, y se representa con una tripleta *(y, o, i)* donde *"y"* es el tipo de sentimiento, *"o"* es la orientación e *"i"* es la intensidad, también se debe de considerar que existen diferentes tipos de sentimientos que a continuación se detallan:

- **Sentimiento racional.** Tales como creencias tangibles y actitudes utilitarias que no expresan emociones.
- **Sentimiento emocional.** Provienen de respuestas no tangibles y emocionales a entidades que se presentan en estados psicológicos de la mente de las personas.
- **Orientación de sentimientos.** Pueden ser positivos, negativos o neutros (significa en ausencia de sentimiento). La orientación de los sentimientos se le llama polaridad, orientación semántica o valencia.
- **Intensidad de sentimientos.** Los sentimientos de cada tipo pueden tener diferentes niveles de intensidad o fuerza, las personas frecuentemente utilizan dos maneras de expresar la intensidad de sus sentimientos en texto. La primera es elegir expresiones de sentimientos (palabras o frases) con las fortalezas adecuadas. La segunda es utilizar intensificadores y disminuidores que son términos que cambian el grado del sentimiento expresado.
- **Sentimientos calificados.** En aplicaciones prácticas, a menudo se utilizan algunas clasificaciones discretas para expresar la intensidad del sentimiento, por lo regular se emplean cinco niveles, que se representan por medio de estrellas teniendo como 1 estrella muy negativo y 5 estrellas muy positivo.

A continuación, se definen otros conceptos necesarios para entender la diferencia de varias palabras similares:

- Afecto. Sentimiento o emoción, específicamente manifestado como expresión facial o lenguaje corporal.
- Emoción. Un estado mental que surge espontáneamente en lugar de por el esfuerzo consciente y frecuentemente va acompañado de cambios fisiológicos.
- Estado de ánimo. Un estado mental o emocional.
- Sentimiento. Un estado afectivo de conciencia, como el que resulta de emociones, sentimientos o deseos.

En seguida, se muestra una clasificación de los diferentes tipos de comentarios que se pueden dejar:

- **Regular**. Comúnmente se refiere simplemente como una opinión y tiene dos subtipos principales:
    - **Directa**. Es una opinión expresada directamente a la entidad o al aspecto de la entidad.
    - **Indirecta**. Es una opinión expresada indirectamente a una entidad basado en efectos positivos o negativos sobre otras entidades.
- **Comparativa**. Expresa una relación de parecidos o diferencias entre dos o más entidades y/o preferencias de un opinante basado en aspectos compartidos de las entidades.
- **Subjetiva**. Su opinión contiene oraciones personales.
- **Implícita**. Opinión que expresa hechos.
- **Personal**. Expresada en primera persona.
- **No personal**. Expresa lo que otra persona puede pensar sobre una entidad o servicio.
- **Meta-opiniones**. Opinión sobre una opinión.

La clasificación de los documentos de sentimientos está definida de la misma forma que una opinión, además de un parámetro extra haciendo una quíntupla *(g, GENERAL, s, h, t)*, dónde GENERAL es resultado obtenido del análisis (Liu, 2015).

3.5 Redes Neuronales Artificiales.

Se puede definir a una red neuronal artificial (RNA) como modelos matemáticos inspirados en sistemas biológicos, adaptados y simulados en computadoras convencionales. Las RNA están inspiradas en el sistema biológico natural. En este sistema la neurona es la unidad de procesamiento, y aunque las RNA sean mucho menos complejas que una red neuronal biológica, también realizan cálculos complejos para procesar información (Navarrete García, 2003).

Este modelo se puede subdividir en dos grandes grupos los cuales son: aprendizaje supervisado y no supervisado, que a continuación se detallará más a fondo cada una de las topologías.

3.5.1. Aprendizaje Supervisado.

La principal característica de esta topología es que se requiere de la intervención de una persona para poder aprender, teniendo un conjunto de información con su respuesta relacionada, y así mapear los patrones.

3.5.1.1. Perceptrón.

La red tipo Perceptrón es un modelo matemático de una neurona biológica propuesto por Frank Rosenblatt en 1959 en su obra Principles of Neurodynamics. Los Perceptrones son redes de propagación hacia adelante basados en unidades binarias. El objetivo de esta red es aprender, usando parte de la información, de tal modo que siempre que se presente un patrón de entrada *p* la salida sea la esperada *t*. En la Figura 3.1 se observa un ejemplo de la forma general del perceptrón (Peláez Chávez & Borja Macías, 2012).

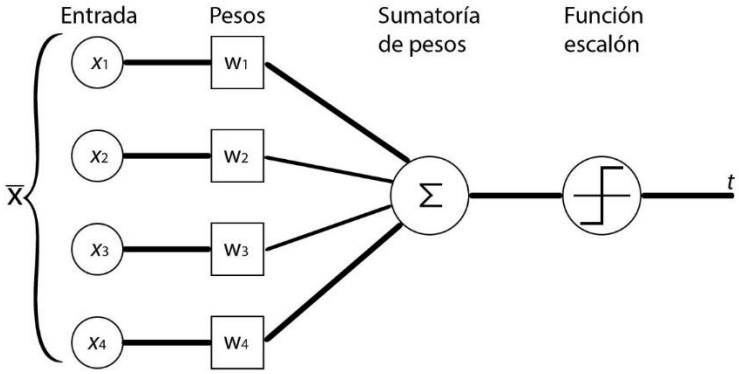

Figura 3.1. Arquitectura del Perceptrón.

3.5.1.2. Perceptrón multicapa.

Es el modelo por excelencia del aprendizaje profundo (Deep Learning, en inglés). El propósito de la red de propagación hacia delante es una aproximación a una función f^*. Por ejemplo, para clasificar $y = f(x)$ mapeando una entrada x a la categoría y. Una red de propagación hacia delante mapea $y = f(x, \theta)$ y aprende el valor de parámetro θ como resultado de una mejor aproximación de la función, como se puede observar en la Figura 3.2. (Goodfellow, et al., 2016).

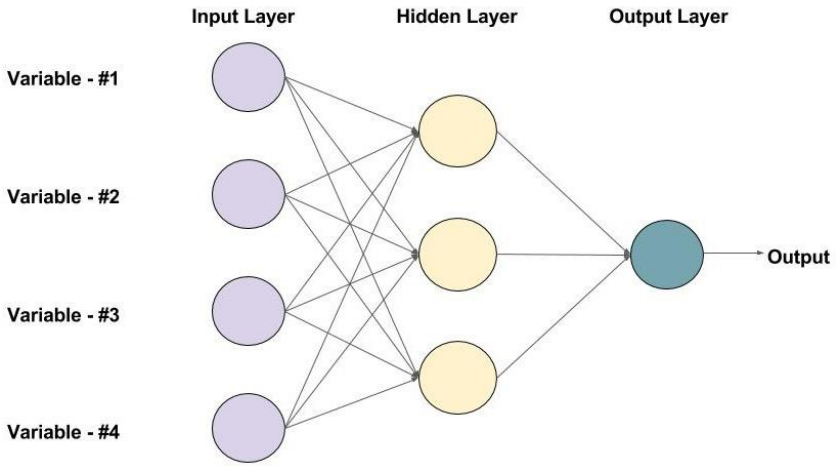

Figura 3.2. Un ejemplo de una red neuronal feed-forward con una sola capa oculta.

3.5.1.3. Red de Retropropagación.

La Red de Retropropagación (del inglés, Backpropagation) es un tipo de red con aprendizaje supervisado. Su funcionamiento consiste en el aprendizaje de un conjunto predefinido de pares entradas-salidas, empleando un ciclo de propagación-adaptación de dos fases: una vez que se ha aplicado un patrón a la entrada de la red como estímulo, éste se propaga desde la primera capa a través de las capas intermedias u ocultas de la red, hasta generar una salida. La señal de salida se compara con la salida deseada y se obtiene un error para cada una de las salidas. El error se propaga hacia atrás, partiendo de la capa de salida, pasando por las capas ocultas hasta llegar a la capa de entrada, como se observa en la Figura 3.3. Las neuronas de las capas ocultas y de entrada, reciben una fracción del total del error, basándose aproximadamente en la contribución relativa que haya aportado cada neurona a la salida original. Este proceso se repite capa por capa, hasta que todas las neuronas de la red hayan recibido un error que describa su contribución relativa al error total. Basándose en el error percibido, se actualizan los pesos de conexión de cada neurona, para hacer que la red converja hacia un estado que permita clasificar correctamente todos los patrones de entrenamiento. La importancia de este proceso consiste en que, a medida que se entrena la red, las neuronas de las capas intermedias se organizan a sí mismas de tal modo que las distintas neuronas aprenden a reconocer distintas características del conjunto de entradas (Valencia Reyes, et al., 2006).

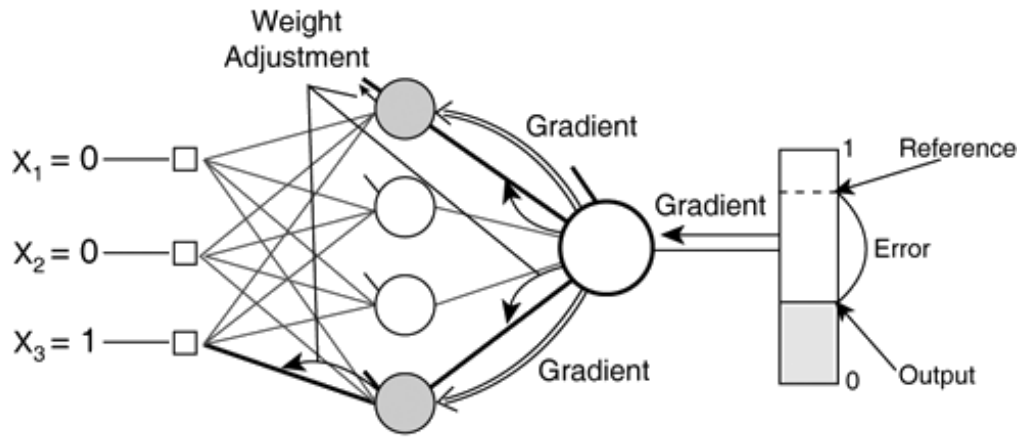

Figura 3.3. Funcionamiento de Backpropagation.

3.5.1.4. Support Vector Machines (SVM).

"Support Vector Machines" (SVM) es un algoritmo de aprendizaje supervisado que puede ser utilizado tanto para clasificación o para problemas de regresión. Sin embargo, se utiliza principalmente en problemas de clasificación. En este algoritmo, se traza cada elemento de datos como un punto en un espacio n-dimensional (donde n es el número de características que se tiene), el valor de cada característica es el valor de una coordenada en particular en dicho espacio. Luego, se realiza la clasificación encontrando el hiperplano (ver Figura 3.4) que diferencia muy bien las clases existentes (Ray, 2017).

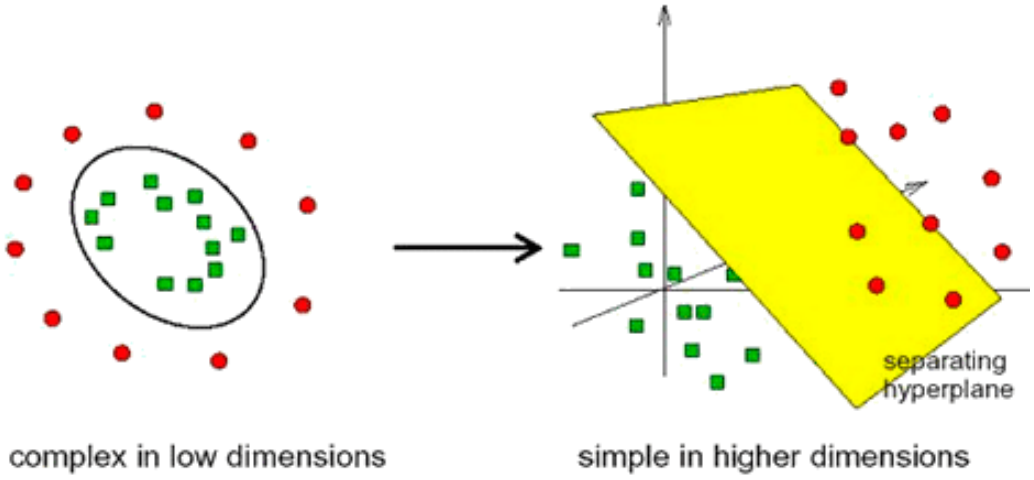

Figura 3.4. Separación lineal en espacios dimensionales mayores (SVM).

3.5.1.5. Learning Vector Quantization (LVQ).

La red Learning Vector Quantization (LVQ), emplea tanto aprendizaje supervisado como no supervisado. Es una red bicapa en la que cada neurona de la primera capa es asignada a una clase con varias neuronas generalmente asignadas a la misma clase. En la segunda capa cada clase es asignada a una neurona. El número de neuronas en la primera capa, S^1, debe ser mayor o al menos igual que el número de neuronas en la segunda capa, S^2, si se identifican N subclases, entonces, S^1 debe contener al menos N neuronas.

Al igual que con redes competitivas, cada neurona en la primera capa de la red LVQ aprende un vector prototipo, el cual permite a la neurona clasificar una región del espacio de entrada, sin

embargo, en lugar de calcular la distancia entre la entrada y el vector de pesos por medio del producto punto, la red LVQ calcula la distancia directamente. Una ventaja de hacer el cálculo de la distancia directamente es que los vectores no necesitan ser normalizados; cuando los vectores son normalizados la respuesta de la red será la misma sin importar la técnica que se utilice. Ver Figura 3.5 (Anon., 2000).

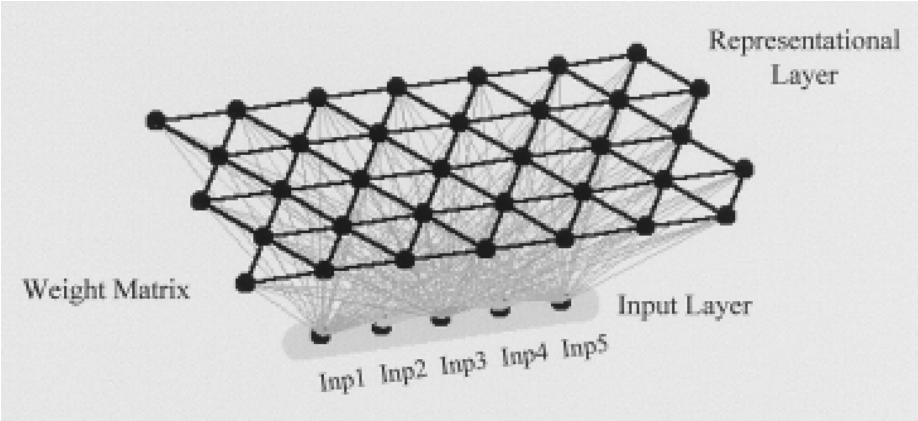

Figura 3.5. Arquitectura LVQ.

3.5.2. Aprendizaje no supervisado.

Los problemas que normalmente se presentan, se tienen solo los datos a tratar sin estar etiquetados, por lo que los algoritmos de aprendizaje supervisado se utilizan principalmente de manera didáctica, y para resolver las dificultades del día a día se utilizan otros tipos de algoritmos en los que no se requiera de la respuesta.

3.5.2.1. Red Hopfield

Una de las mayores contribuciones en el área de las redes neuronales fue desarrollada por el físico John Hopfield en la década de los 80, quien propuso un modelo neuronal no lineal, conocido como la red de Hopfield. La red de Hopfield es presentada como un modelo de memoria asociativa de patrones, en el sentido de que es capaz de recuperar patrones almacenados a partir de información incompleta sobre los patrones o incluso a partir de patrones

con ruido. En la Figura 3.6 se ilustra cómo una red de este tipo está interconectada en una sola capa (Peláez Chávez & Borja Macías, 2012).

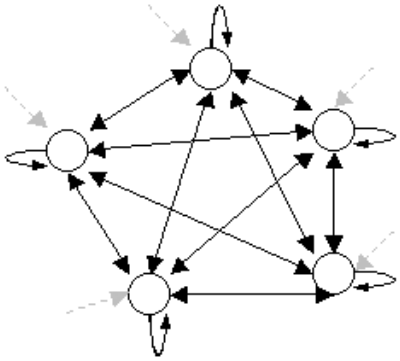

Figura 3.6. Arquitectura de una red de Hopfield.

3.5.2.2. Mapas Auto-Organizados (del inglés, Self-Organizing Map, SOM).

Teuvo Kohonen presentó en 1982 un modelo de red neuronal con capacidad para formar mapas de características de manera similar a como ocurre en el cerebro. El objetivo de Kohonen era demostrar que un estímulo externo por sí solo (suponiendo una estructura propia y una descripción funcional del comportamiento de la red), era suficiente para forzar la formación de los mapas, uno de los problemas que se puede encontrar con este tipo de arquitectura es que la red se debe de definir desde un principio como se ve en la Figura 3.7, haciendo que no se pueda adaptar a nuevos datos (Gómez Coaboy, 2015).

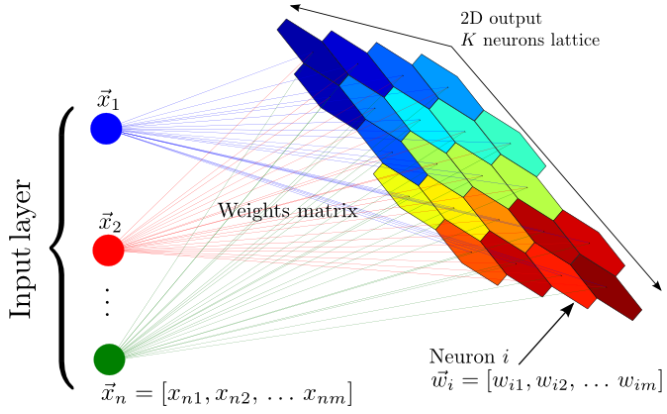

Figura 3.7. Arquitectura de SOM.

3.5.2.3. Mapas auto-organizados con crecimiento jerárquico (del inglés, Growing Hierarchical SOM, GHSOM).

Uno de los defectos del SOM radica en su arquitectura fija, que hay que definir de antemano. Por otra parte, las variantes de crecimiento dinámico del SOM tienden a producir enormes mapas que son difíciles de manejar. Esto lleva al desarrollo del GHSOM, una nueva arquitectura que crece tanto de forma jerárquica según la distribución de los datos, permitiendo una descomposición y navegación jerárquica en subpartes de los datos, como de forma horizontal, lo que significa que el tamaño de cada mapa individual se adapta a las necesidades del espacio de entrada (ver Figura 3.8). Esto proporciona una interfaz conveniente para navegar por grandes bibliotecas digitales, ya que sigue de cerca el modelo de las bibliotecas convencionales, que también están estructuradas, por ejemplo, diferentes pisos, secciones y estantes (Rauber, et al., 2000).

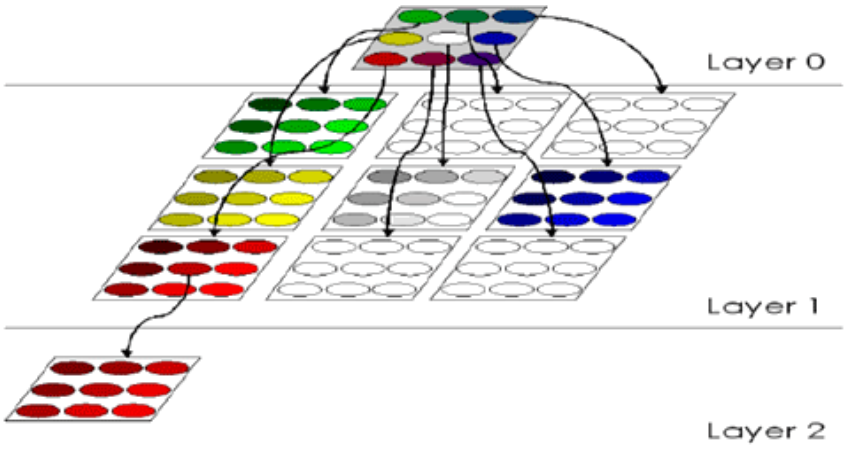

Figura 3.8. Arquitectura de GHSOM.

3.6 Lenguajes de Programación.

Son lenguajes formales artificiales diseñados para poder comunicarse con la computadora con la finalidad de crear programas informáticos.

3.6.1. C#.

Los principales inventores de este lenguaje fueron Anders Hejlsberg, Scott Wiltamuth y Peter Golder. La primer gran distribución implementada de C# fue liberada por Microsoft en julio de 2000, como parte de la iniciativa .NET Framework. También se formó un estándar para la biblioteca y entorno de ejecución llamado Infraestructura de Lenguaje Común (CLI, por sus siglas en inglés).

A medida que C# evolucionó, los objetivos utilizados en su diseño fueron los siguientes:

- Está orientado a ser simple, moderno, de propósito general y es un lenguaje de programación orientado a objetos.
- También pueden dar soporte a los principios de la ingeniería de software, detección de intentos de usar variables no inicializadas, un recolector de basura automático, etc.
- Es un lenguaje robusto.
- Proporciona una mejor productividad al programador, haciendo algunas actividades más sencillas.
- El lenguaje está desarrollado para uso en el desarrollo de componentes de software adecuado para el despliegue en entornos distribuidos.
- Su soporte a la internacionalización es muy importante.
- Pretende que sea adecuado para la escritura de aplicaciones en sistemas alojados y embebidos, desde algo muy grande como sistemas operativos sofisticados hasta lo más pequeño con funciones dedicadas.
- Aunque las aplicaciones de C# están destinadas a ser económicos con respecto a la memoria y potencia de procesamiento, este no pretende competir directamente con C o lenguaje ensamblador (Anon., 2006).

3.6.1.1 Ejemplos en C#.

A continuación, se muestra un código para que en la pantalla muestre el texto “Hola mundo”, la primera línea es necesaria para que cargue las librerías necesarias para su funcionamiento, y

en la parte que dice *Console.Write* es la instrucción que muestra el texto previamente mencionado.

```
using System;
class ejemplo {
    static void Main() {
        Console.Write(“Hola mundo”);
    }
}
```

A continuación, se presenta un segundo ejemplo que hace un ciclo desde el 1 hasta el 99 en el cual se muestran todos los números que sean par; esto se hace con la condición en la que comprueba que si el residuo de la división del número actual “i” entre 2 es igual a 0 imprime el número.

```
using System;
class ejemplo2 {
    static void Main() {
        for (i = 1; i < 100; i++) {
            if (i % 2 == 0) {
                Console.Write("{0}|", i);
            }
        }
    }
}
```

3.6.2 Perl (del inglés, Practical extraction and report language).

Es un lenguaje de programación desarrollado por Larry Wall, especialmente diseñado para procesamiento de texto. Funciona en una variedad de plataformas, como Windows, Mac OS y

las diversas versiones de UNIX. A continuación, se muestran de las características más importantes del lenguaje.

- Es un lenguaje estable y de plataforma cruzada; esto quiere decir, que se puede ejecutar en diferentes sistemas operativos sin problema alguno.
- Se utiliza para proyectos de misión crítica en los sectores públicos y privados.
- Es un software de código abierto licenciado bajo Artistic License o GNU (General Public License).
- Puede manejar la encriptación de datos en web, incluyendo transacción para comercios electrónicos.
- Toma lo mejor de otros lenguajes como lo es C, AWK, sh y BASIC.
- La interfaz DBI para la integración de bases de datos soporta otros lenguajes como Oracle, Sybase, Postgres, MySQL entre otros.
- Trabaja con HTML, XML y otros lenguajes de marcado.
- Soporta Unicode.
- Soporta programación modular y orientada a objetos.
- Es interpretado por lo que puede ser embebido (hacer librerías y ser llamado desde otros lenguajes) en otros sistemas (Anon., s.f.).

3.6.2.1 Ejemplos en Perl

En seguida se muestra un ejemplo de un programa en Perl que imprime en pantalla “Hola Mundo”, la primera línea es obligatoria en todos los programas para que el intérprete Perl lo reconozca correctamente.

```
#!/usr/bin/perl
print “Hola Mundo”;
```

El siguiente ejemplo se hace un ciclo para mostrar los números pares del 1 al 99.

```
#!/usr/bin/perl
```

```
for my $i (1..99){
   if($i % 2 == 0){
        print “$i|”;
   }
}
```

3.6.3 PHP (del inglés, Hypertext Pre-Processor)

Es un lenguaje de programación para construir sitios web interactivos y dinámicos. Como regla general, los programas de PHP se ejecutan en un servidor Web. Una de las características principales de PHP es que se pueden unir códigos de HTML con PHP en las páginas web.

¿Qué significa exactamente la frase “páginas web interactivas y dinámicas”? Una página web dinámica es una en la que todo su contenido puede cambiar cada vez que la página es visitada. En contraste con una página estática que su contenido es fijo.

Todas estas características significan que puedes usar PHP para crear prácticamente cualquier aplicación web dinámica que se desee.

Algunos ejemplos comunes de scripts PHP incluyen:

- Foros web que permiten a los visitantes mandar mensajes y discutir sobre temas.
- Motores de búsqueda que le da a la gente el contenido de un sitio web o una base de datos.
- Scripts de sondeo que permiten a los visitantes hacer encuestas.
- Sistemas de gestión de contenidos y blogs, que permite a los Webmaster crear con facilidad sitios con el mínimo de conocimiento técnico.
- La aplicación de Webmail permite a la gente enviar y recibir correos electrónicos usando el navegador web.
- Tiendas en línea que permiten a los usuarios comprar productos y servicios (Doyle, 2009) .

3.6.3.1 Ejemplo de PHP

A continuación, se tiene un ejemplo que muestra en la pantalla el texto “Hola mundo”, la primera línea es la que se encarga de que el intérprete reconozca todo lo que sigue como código de PHP, la instrucción *echo* sirve para imprimir una cadena de caracteres.

```
<?php
    echo “Hola mundo”;
?>
```

Para el siguiente ejemplo se imprimen todos los números pares desde el 1 hasta el 99.

```
<?php
    for($i = 1; $i < 100; $i++){
          if($i % 2 == 0){
                echo “$i|”;
          }
    }
?>
```

3.6.4 HTML5.

Este lenguaje sirve para construir la estructura de una página web. Es un lenguaje de etiquetado, por lo que, lo hace fácil de usar y aprender. De este modo se tiene una etiqueta para decir que el texto que se encuentre adentro es un título, un párrafo o que debe de mostrarse en negritas. En las etiquetas se pueden introducir atributos, para así modificar el comportamiento normal o agregar información, y así poder hacer uso de ella más adelante.

No se requiere de una aplicación especial para poder desarrollar páginas web, se puede hacer desde un bloc de notas (Ducket, 2011).

3.6.4.1 Ejemplo de HTML5.

A continuación, se muestra la estructura básica para mostrar en una página web correctamente el texto “Hola mundo”.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Ejemplo de Hola mundo</title>
</head>
<body>
    <div>
        Hola mundo
    </div>
</body>
</html>
```

3.6.5 CSS (del inglés, Cascading Style Sheets, Hojas de Estilo en Cascada).

CSS es un lenguaje, que permite crear reglas que especifican cómo debe de aparecer el contenido de la página web, por ejemplo, cambiar el fondo a una sección o el tipo y color de letra a un texto. Existen diferentes formas de agregar estilos a una etiqueta de HTML, los cuales son: con un archivo externo, en el mismo archivo con una etiqueta *style* y como atributo en una etiqueta. En los primeros dos puntos se hace referencia por atributos o el tipo de etiqueta a la cual se le quiere aplicar.

También se pueden crear animaciones al pasar el puntero del ratón por encima de una etiqueta seleccionada, además de poder acomodar las secciones de diferentes formas (Ducket, 2011).

3.6.5.1 Ejemplo de CSS

A continuación, se muestra un ejemplo de código en el que por medio de CSS se les da formato a las etiquetas *div*, dando como resultado un recuadro negro, con un texto color azul y alineado al centro que ocupa todo el ancho de la pantalla.

```
<style type=“text/css”>
   div{
        width: 100%;
        text-align: center;
        color: blue;
        background-color: black;
   }
</style>
```

3.6.6 Javascript.

Es el lenguaje más popular para scripts de navegadores, una de las mayores opciones que nos ofrece es la posibilidad de modificar el contenido o estructura de la página, a través de acciones que el usuario ejecute. Teniendo la capacidad de guardar información en el navegador que se está visualizando y así poder dar un servicio personalizado.

Este lenguaje junto con CSS hacen que se tenga la mejor experiencia para el usuario, dando como resultado las mejores interfaces posibles.

3.6.6.1 Ejemplos de JavaScript.

A continuación, se ve un ejemplo de código en JavaScript en el cual en la consola del navegador muestra el mensaje “Hola mundo”.

```
<script type=“text/javascript”>
   console.log(“Hola mundo”);
</script>
```

El siguiente ejemplo muestra en la consola del navegador todos los números pares desde el 1 hasta el 99.

```
<script type=“text/javascript”>
   for(var i = 1; i < 100; i++){
        if(i % 2){
             console.log(i);
        }
   }
</script>
```

3.7. Programación extrema.

Es una disciplina de desarrollo ágil basado en valores de simplicidad, comunicación, retroalimentación y coraje.

La programación extrema tiene 12 prácticas principales las cuales son:

1. Planeación de procesos.
2. Pequeñas entregas.
3. Testing (pruebas).
4. Metáforas (para tener una mejor comunicación con el cliente).
5. Diseño simple.
6. Refactorización (cambios internos sin alterar lo externo).
7. Programación en parejas.
8. Código de dueños colectivos.
9. Integración continúa.
10. 40 horas de trabajo a la semana (para mayor eficiencia en el código).
11. El cliente en el lugar (siempre debe de estar disponible y formar parte del equipo).
12. Estándares de código (Lindstrom & Jeffries, s.f.).

Capítulo 4. DESARROLLO

En este capítulo se aborda el desarrollo del módulo de detección de polaridad, dividido en una serie de secciones, entre las cuáles se encuentra la metodología que se usó en los algoritmos aplicados. A continuación, un breve resumen:

El proceso de clasificado de textos es un área que se ha empezado a investigar en profundidad recientemente, debido a que se puede obtener información destacable de una forma rápida. Para empezar con todo este análisis se debe de hacer primeramente un preprocesamiento en el texto, para obtener una información más manejable computacionalmente hablando, luego se le asignan valores a cada palabra de los documentos con base en un algoritmo llamado Frecuencia de un Término en la Frecuencia Inversa de un Documento (TF-IDF, por sus siglas en inglés), de este modo se agregan a archivo para que el sistema reconozca la información y haga los cálculos correspondientes, con el algoritmo GHSOM y al final se hacen cálculos posprocesamiento para obtener el resultado correspondiente.

4.1 Metodología

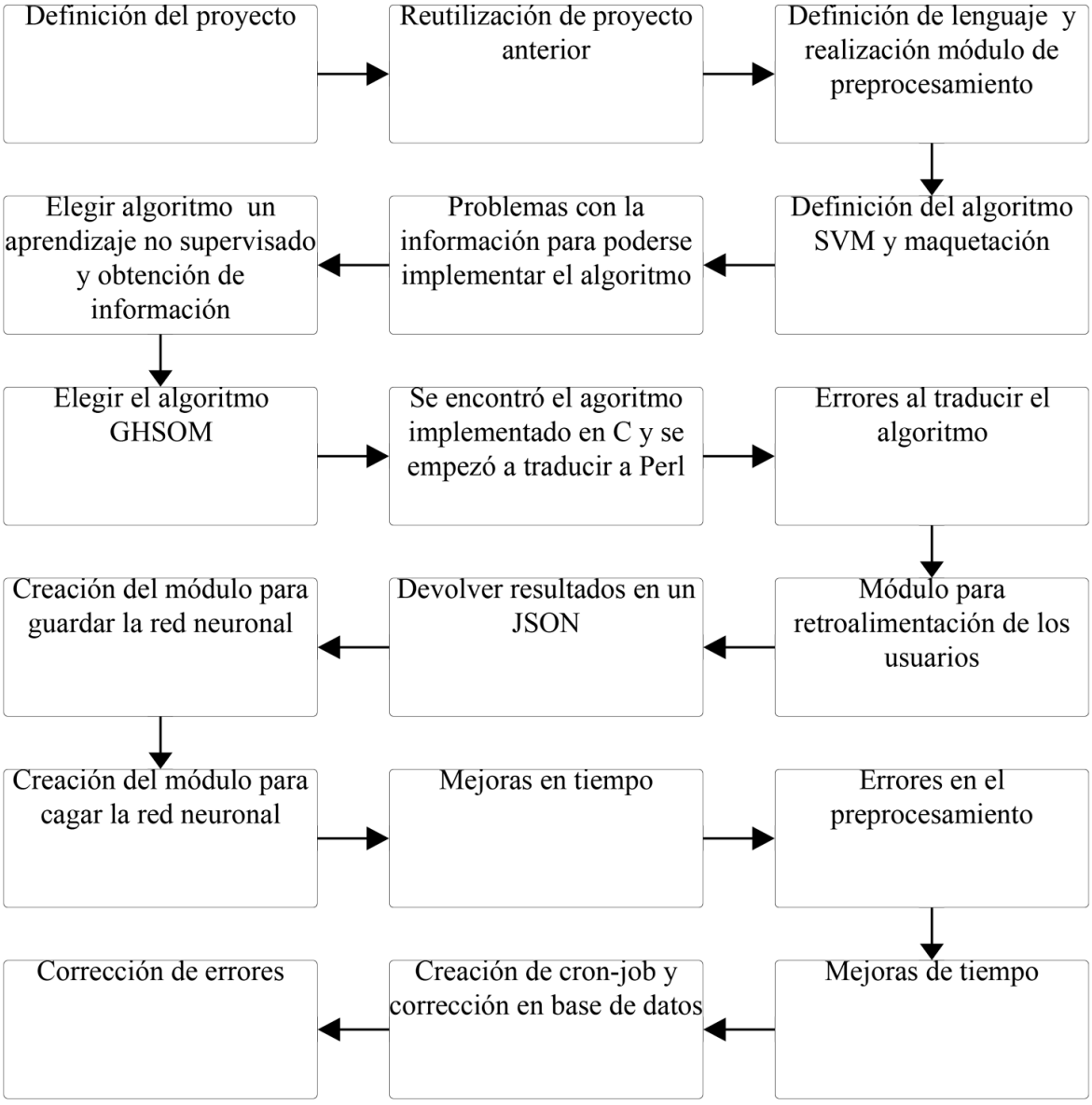

Figura 4.1. Diagrama de la metodología.

Tras encontrarse el área de oportunidad en el área de trabajo, se comentó en División de Estudios de Posgrado e Investigación (DEPI) del Instituto Tecnológico de Chihuahua II, para evaluar si el proyecto cumplía con los requisitos mínimos como proyecto de tesis de posgrado,

tras obtener una respuesta positiva se concertó una reunión dentro del área de trabajo, para exponer, grosso modo, de qué se trataba el proyecto y definir los requisitos del proyecto.

Al tener un acercamiento previo a un proyecto de análisis de sentimiento, el cual se desarrolló de una forma arcaica en el que solo sumaba los valores de las palabras con base en un diccionario con valores asignados (proyecto final de una asignatura de nivel licenciatura), se empezó a desarrollar una aplicación que se encargaría de la limpieza del texto. También se buscaron opciones para implementar algoritmos de tipo stemming[3] para el idioma español, con ello, en una entrevista informal en el área de trabajo, se contempló la posibilidad de ampliar el desarrollo del software, para que también se analizarán los comentarios en redes sociales.

Posteriormente se formalizó totalmente el alcance del proyecto, dejando también establecido el lenguaje de programación a utilizar. También se empezó a implementar el código necesario para el preprocesamiento del texto, haciendo la limpieza necesaria con un programa escrito en PHP para posteriormente pasarlo a un módulo en Perl que es el encargado de aplicar el algoritmo de stemming a un texto completo.

Debido a que la empresa cuenta con una gran base de datos de comentarios, se optó por el algoritmo de aprendizaje supervisado llamado Suport Vector Machines (SVM), debido a que es un algoritmo óptimo para datos previamente etiquetados. Una vez que se eligió el sistema de aprendizaje, se hizo una revisión de todo lo necesario para implementar el software, como el algoritmo, además de planear y maquetar el funcionamiento del sistema.

Mientras se hacía la búsqueda del algoritmo SVM, simultáneamente se revisaba la información del buzón de sugerencias y en éste se encontró el inconveniente de que la información útil no era la necesaria para el algoritmo, debido a que aproximadamente un 50 % de la información era contenido basura (SPAM) o bots que rellenaban la información. Esto hizo que se buscara otro sistema de aprendizaje para el software, el cual fue a un algoritmo de aprendizaje no supervisado, además se reconsideró el módulo de los comentarios en redes

[3] Método para reducir una palabra a su mínima expresión (stem, en inglés), por ejemplo, bibliotecas y bibliotecario su stem sería “bibliotec”.

sociales tomándolo como un hecho, para obtener nueva información y así tener un mejor filtro que son las redes sociales por sí mismas.

El mejor algoritmo para implementar que se encontró fue GHSOM debido a que su arquitectura va cambiando dinámicamente mientras se analiza la información, lo que hace que pueda ir mejorando conforme más información va analizando.

Una vez elegida la arquitectura adecuada se buscó el algoritmo, y se encontró un software implementado en C, éste se tradujo al lenguaje de programación que se eligió para el desarrollo del sistema. Al terminar de traducir el programa se encontró con el problema de que a pesar de ser pocos comentarios los que se dejaban analizando este no terminaba de ejecutarse después de horas de haber empezado, por lo que, se empezó un escrutinio minucioso en el código para encontrar el error.

Al depurar el código se encontró el problema que tenía el software, el cual consistía en una variable que se usaba para detener uno de los ciclos principales, ésta hacía referencia a su dirección de memoria en vez de a su valor.

Después de un tiempo dejando el cron-job[4] de guardado de comentarios de Twitter se observó que no funcionaba correctamente debido a que los comentarios se guardaban repetidos, llegando a la conclusión de que no solo se deberá de diferenciar con el id del tuit, sino que además se debería de agregar la posibilidad de comparar los textos. Después de esto se observó que los emojis no eran agregados a los comentarios, esto era debido a que la base de datos no tenía la codificación adecuada para aceptar estos caracteres, la codificación de caracteres (charset, en inglés) correcto en la base de datos es utf8mb64_unicode.

Después de analizar profundamente el sistema de análisis para encontrar la forma de mejorar los tiempos de respuesta, con la meta de que cada comentario dure analizándose un promedio de 15 a 20 segundos, y actualmente la duración es de 1 minuto por comentario. Se encontró el problema en varios ciclos de las funciones de entrenamiento que tenían el límite de detenido

[4] Comandos que se ejecutan con una frecuencia fija.

mal, haciendo un ciclo de más, con esto la meta propuesta casi se alcanza, dejándolo en 40 segundos por comentario.

Revisando los comentarios con los que se contaba en ese momento, se observó que el preprocesamiento no se hacía correctamente en algunos textos en particular, dejando pasar información que no es de utilidad para el sistema, por lo que se optó por utilizar expresiones regulares, que se encarga de corregir los enlaces a páginas o imágenes.

Intentando llegar a la meta anterior mencionada, se empieza a modificar y probar diferentes configuraciones en el algoritmo GHSOM, reduciendo el tiempo a 30 segundos, con el inconveniente de que se tuvo que sacrificar la exactitud de esta, dejándola con un 84.1 %.

Se implementó un módulo el cual guarda el estado de la neurona al terminar de analizar la información, dando la posibilidad de guardarla en la carpeta que se deseé, para potenciar y mejorar el desempeño de esta, también se carga esa red, para que no tenga que recorrer la misma información ni reconstruir una red neuronal desde cero. Se tuvieron distintos problemas al cargar la red neuronal, como el guardar la información de las palabras.

Para finalizar se implementaron diferentes módulos, como lo son un archivo que guarda un archivo tipo JSON con la información necesaria para poder evaluar si un comentario es positivo, negativo o neutro, se hicieron diferentes evaluaciones para obtener la mejor interpretación. Los módulos para agregar nuevos clientes con sus respectivas cuentas de Twitter, para tener una administración adecuada. Para finalizar con la implementación de módulos, se hizo un nuevo módulo para obtener una retroalimentación de las personas, además de agregar un diccionario de palabras que se va alimentando automáticamente con la interacción del administrador tomando las palabras más comunes.

En las siguientes secciones se explica paso a paso, el proceso que se siguió para el desarrollo del software.

4.2 Obtención de información.

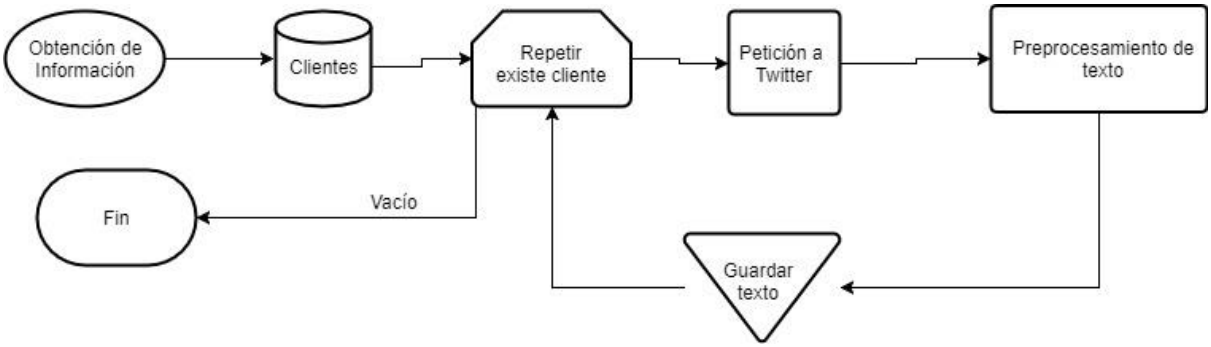

Figura 4.2. Diagrama de obtención de información.

Como primer paso, se desarrolló un módulo que se encarga de obtener los tuits en los que se menciona una cuenta registrada en el sistema. Para la obtención de los tuits se utiliza la API oficial de Twitter, esta obtención se hace de manera automática por medio de cron-job que se ejecuta cada hora; se guarda en una base de datos el texto original, y el texto preprocesado. En el siguiente apartado se abordará con más profundidad lo realizado.

4.3 Técnicas de preprocesamiento de texto.

Cuando se va a procesar texto escrito en español, los tipos de palabras a eliminar y procesos que se le hace al texto va a depender totalmente del análisis que se requiera. A continuación, se mencionan los que se utilizaron en el módulo automatizado de preprocesamiento de este proyecto de análisis de sentimientos.

1. **Artículos.-** Los artículos solo son usados para dar soporte a sustantivos, adjetivos, adverbios y participios por lo que no brinda información relevante para el sistema de análisis.
2. **Números.-** Se eliminan los números, ya que al analizar los textos para obtener una opinión no le brinda información relevante.
3. **Guiones.-** Aquí se tuvo especial cuidado debido a que existen emoticones que contienen guiones y se procuró solo eliminar los que estaban entre textos.

4. **Tildes.-** Las tildes se eliminan debido a que duplican información al reconocer como dos palabras distintas el significado de una misma.
5. **Signos de puntuación.-** Se quitan porque el sentido de la oración queda casi intacta y hace más eficiente el procesamiento.
6. **Cambiar a minúsculas.-** La utilidad de este preprocesamiento es porque las palabras tienen el mismo significado si se usan todo en minúsculas y el procesamiento es menor ya que las reconoce como una sola palabra.
7. **Stemmer.-** El stemmer es una técnica utilizada para obtener la raíz de la palabra, no es una raíz etimológica, sino la palabra mínima de donde salen todas las conjugaciones en un verbo y los afijos que pueda tener una palabra.

4.4 Preprocesamiento de la información

Para que los textos puedan ser analizados, deben de convertirse en medidas cuantitativas para que la computadora lo pueda procesar con facilidad. Para obtención de información lo mejor es usar la técnica de TF-IDF, usando las siguientes formulas:

Para obtener la Frecuencia de Término en un documento se utiliza la fórmula 1.

$$tf(t,d) = \frac{f(t,d)}{\max\{f(w,d): w \in d\}} \quad (1)$$

Donde:

- $f(t,d)$: es la frecuencia del término o palabra en el documento.
- $\max\{f(w,d): w \in d\}$: Para evitar que los documentos grandes tengan predisposición se divide por la palabra que más se repite en el documento.

Para obtener la frecuencia inversa de un término en los documentos se utiliza la fórmula 2.

$$idf(t,D) = \log\left(\frac{|D|}{1 + |\{d \in D : t \in d\}|}\right) \quad (2)$$

Donde:

- $|D|$: cardinalidad de D, o número de documentos en la colección.

- $|\{d \in D : t \in d\}|$: número de documentos donde aparece el término t. Si el término no está en la colección se producirá una división-por-cero. Por lo tanto, es común ajustar esta fórmula a $1 + |\{d \in D : t \in d\}|$.

Por último, para poder calcular el TF-IDF se aplica la fórmula 3:

$$tfidf(t, d, D) = tf(t, d) * idf(t, D) \quad (3)$$

4.5 Archivos

Después de tener esta información de los tuits a analizar, se guardan 3 archivos con la siguiente información y formato:

1. Archivo .tfxidf.-
    a. Es el tipo de archivo que se va a usar; $TYPE vec_tfxidf.
    b. La cantidad de documentos disponibles; $XDIM 50.
    c. Dimensión en Y de la matriz (siempre va 1); $YDIM 1.
    d. La cantidad de palabras distintas; $VEC_DIM 363.
    e. TF-IDF de cada palabra separadas por espacio, el último texto antes del salto de línea se va a reconocer como el identificador del documento; para separar cada documento es con un nuevo renglón.
2. Archivo .idv.-
    a. Tipo de archivo; $TYPE template.
    b. Número de características por renglón; $XDIM 7.
    c. Cantidad de documentos; $YDIM 50.
    d. Cantidad de palabras distintas; $VEC_DIM 363.
    e. 1 bronc 4 8 0 2 2; A continuación, se explica qué es cada una de las variables mostradas anteriormente.
        i. Id de la palabra.
        ii. Palabra.
        iii. Número de tuits en los que se encuentra.
        iv. Número de veces que aparece

v. Cantidad menor de veces que aparece en un Tweet.
vi. Cantidad mayor de veces que aparece en un Tweet.
vii. Media de veces que aparece en un Tweet.

3. Archivo .mqe0.-
    a. Es el error medio de cada palabra que existe en los documentos. Se obtiene con la fórmula 4; $MEAN_VEC 0.000958949149973496 0.00627154699977218 …

$$Mean(w) = \frac{\sum_{i=0}^{n} tfidf(w,i,D)}{n} \quad (4)$$

Donde:

- $\sum_{i=0}^{n} tfidf(w,i,D)$: es la sumatoria de los TF-IDF de un término en todos los documentos.
- n: Número total de documentos.
    b. Minimum Quantization Error (MQE) calculado con la fórmula 5; $MQE0 11.4854676978636

$$MQE = \sum_{i=0}^{n} \sum_{w=0}^{W} (tfidf(w,i,D) - Mean(w))^2 \quad (5)$$

Donde:

- $Mean(w)$: Error medio de cada palabra.
- $tfidf(w,i,D)$: TF-IDF de cada palabra de todos los documentos.

4.6 Análisis.

4.6.1 Generar archivo de propiedades.

Para poder analizar los archivos que se generaron se debe de crear un archivo más; las configuraciones del algoritmo GHSOM, una propiedad por renglón y separados por un "=" entre nombre de la propiedad y el valor; los posibles valores se muestran en la Tabla 4.1.

Este archivo debe de ir en la misma carpeta o en una subcarpeta que el programa principal; al ejecutar el programa se debe de pasar como parámetro la(s) subcarpeta(s) y el nombre del archivo con la extensión.

Tabla 4.1 Tabla descriptiva de las propiedades necesarias para el funcionamiento del sistema de análisis.

Propiedad	Rango	Descripción
EXPAND_ CYCLES	>= 1	Número de ciclos después de cada mapa revisado, para una eventual expansión; Ejemplo 100 vectores de entradas, 10 ciclos = 1000 patrones elegidos analizados para el SOM para el aprendizaje.
MAX_CYCLES		Número máximo de ciclos que da para pasar a la siguiente SOM. Si se deja el valor en 0, va a iterar el ciclo de aprendizaje hasta que el valor máximo del error sea alcanzado.
TAU_1	[0-1]	Porcentaje de error restante que tiene que ser alcanzado por cada mapa, también conocido como criterio de parada de crecimiento horizontal. Cuanto más pequeño sea el valor cada mapa crecerá más y la jerarquía será más plana. Un ejemplo para empezar es 0.25.
TAU_2	[0-1]	Grado de granularidad representado por los mapas en la capa más baja. Cuanto menor sea, más detallada será la representación de los datos y, por lo tanto, mayor será la estructura general de GHSOM. Un ejemplo para empezar es 0.1, si se deja el valor de 1 sólo se formará una sola SOM en la primera capa.
INITIAL_ LEARNRATE	[0-1]	Determina qué tan fuerte se adapta el ganador y sus unidades vecinas. Con el tiempo decrementa su valor, un ejemplo es 0.8.

NR	<= 0	Altura del vecindario gaussiano en la frontera del mapa define la "altura" del vecindario gaussiano en la unidad que es max-x (anchura) o max-y (altura) lejos del ganador (usado para calcular sigma del kernel gaussiano).
*HTML_PREFIX		Prefijo de los archivos de salida.
*DATAFILE_ EXTENSION	Puede ser vacío	Sufijo para la referencia de los archivos de datos en la tabla HTML.
randomSeed	any	Valor de la semilla inicial del generador de números aleatorios.
inputFile		Archivo en el que se guardó la información del tf-idf.
descriptionFile		Archivo con la extensión .idv que contiene la información de las palabras.
*printMQE	true | false	Sirve para imprimir en un archivo los MQE.
normInputVectors	NONE | LENGTH | INTERVAL	Cómo se van a normalizar los vectores de entrada; NONE = se utilizarán datos de entrada sin procesar; LENGTH = los vectores están normalizados a la longitud 1; INTERVAL = elementos vectoriales se transforman en el intervalo [0-1].
*saveAsHTML	true | false	Guardar archivos HTML
*saveAsSOMLib	true | false	Guardar archivo SOM
INITIAL_X_SIZE	>= 1	Tamaño inicial de los nuevos mapas en dirección x.
INITIAL_Y_SIZE	>= 1	Tamaño inicial de los nuevos mapas en dirección y.
*LABELS_NUM	>= 0	Número máximo de etiquetas.

*LABELS_ONLY	true | false	Las etiquetas sólo son mostradas en los nodos que se expanden.
*LABELS_ THRESHOLD	[0-1]	Las características más importantes se utilizan como etiqueta; un valor 0.8 significa que sólo las características con valores en el 20 % superior de todos se imprime como etiqueta; cuanto más bajo sea el valor, más etiquetas se mostrarán (limitado por LABELS_NUM).
ORIENTATION	true | false	Indica si los mapas van a tener un sentido. Cambia el cálculo de los pesos en las neuronas.
MQE0_FILE		Ruta donde se encuentra el archivo con la extensión .mqe0.
saveNN	true | false	Variable que indica si se va a guardar la red neuronal formada en el ciclo de aprendizaje.
NN		Ruta donde se va a guardar la red neuronal y el archivo de descripción, *.nnept y *.descept, respectivamente.
loaderNN		Ruta del archivo donde se obtiene una red neuronal ya existente.
loaderDescNN		Ruta donde se va a cargar el archivo de la descripción de la red neuronal.
FAST		Variable que indica si se va a hacer un análisis rápido, basado en la probabilidad.
probability	[0-1]	Probabilidad usada para analizar una red neuronal con análisis rápido.

4.6.2 Proceso.

Lo primero que hace la aplicación al iniciar es cargar las propiedades definidas en el archivo mencionado arriba, después inicializa las variables necesarias para empezar el análisis, que se

encuentran en los archivos arriba mencionados (.tfidf, .idv y .mqe0), una vez cargados los archivos, el sistema revisa en las propiedades si se va a cargar una red neuronal, a continuación, se describe el proceso:

- *Carga de red neuronal:*

Primero se abre el archivo donde está guardada la información de la red neuronal, leyendo renglón por renglón se guarda la información de las capas (arreglos solo con la información de las capas) y las neuronas (arreglo de arreglos con variables de tipo neurona) en arreglos temporales. El siguiente paso es leer el archivo con el vector descripción, después, se genera un único arreglo entre el archivo .idv y .descept y ese arreglo se guarda en las variables globales. Después hace un ciclo recorriendo el arreglo que contiene la información de las capas de la red neuronal, se crean los mapas en las capas proporcionadas con sus neuronas correspondientes. Ya que se tiene cargada la red neuronal el siguiente paso es iniciar el proceso de entrenamiento.

- *Entrenamiento:*

Cada capa que se va a entrenar, se deben de obtener algunas variables globales, las cuales son: TAU_1, TAU_2 e INITIAL_LEARNRATE, que sirven para el aprendizaje de la red neuronal. Lo primero que se hace es conseguir una entrada de forma aleatoria a partir de ahí se obtiene el vecino más cercano por cada patrón de entrada, para calcular la distancia se utiliza la fórmula 6:

$$D = \max\{\forall x \in X \sum_{i=0}^{n} \left(w_i^1 - w_i^2\right)^2\} \quad (6)$$

Dónde:

- D: Distancia.
- w: Pesos de los vectores.
- x: Elemento del vector de entrada.
- n: Número de elementos que contiene x.
- X: Patrón de entrada.

La tasa de aprendizaje del sistema es dinámica, y desde el principio se debe de calcular la brecha de la tasa de aprendizaje y se calcula con la fórmula 7:

$$STRETCH\_PARAM\_LEARN = \frac{EXPAND\_\text{CYCLES} * \text{n}}{16} \quad (7)$$

Donde:

- $EXPAND_$CYCLES: Variable global que se utiliza para definir cuántos ciclos se hacen antes de expandir un mapa.
- n: Número de documentos.

La tasa de aprendizaje se ajusta en cada ciclo con las siguientes variables, el ciclo actual, la tasa de aprendizaje inicial (parámetro que se carga desde el archivo de propiedades), la variable para expandir los mapas, el número total de documentos y la brecha de la tasa de aprendizaje, utilizando la fórmula 8.

$$\alpha = \alpha_i e^{\frac{-(c \bmod (\text{EXPAND\_CYCLES} * \text{n}))}{STRETCH\_PARAM\_LEARN}} + 0.0001 \quad (8)$$

Dónde:

- α: aprendizaje.
- α_i: aprendizaje inicial.
- c: ciclo actual.
- mod: operación módulo.
- n: tamaño de los pesos.

La variable del vecindario, al igual que la tasa de aprendizaje se usa un parámetro de brecha del vecino más cercano.

$$STRETCH\_PARAM\_NEIGHB = \frac{EXPAND\_CYCLES * n}{6.67} \quad (9)$$

Donde:

- $EXPAND_$CYCLES: Variable global que se utiliza para definir cuántos ciclos se hacen antes de expandir un mapa.
- n: Número de documentos.

Para ajustar los pesos de las neuronas en cada ciclo se utiliza la variable *neighborhood* que se calcula con la fórmula 10.

$$neighborhood = neighborhood_i e^{\frac{-(c \bmod (\text{EXPAND\_CYCLES} * \text{n}))}{STRETCH\_PARAM\_NEIGHB}} + 0.55 \quad (10)$$

Donde:

- neighborhood: Parámetro que se utiliza para ajustar los pesos de cada neurona.

Los análisis se van a seguir haciendo hasta que se cumpla la condición de la fórmula 11.

$$si\ MQE \leq (SuperMQE * TAU\_1) \text{ o}$$
$$MAX\_CYCLES > 0 \ \&\&\ currentCycle \geq (MAX\_CYCLES * tamañoDeDataItems) \quad (11)$$
entonces Detener Análisis,
si no: Agregar columna o renglón y ajustar las variables *neighborhood* y α.

Después de los cálculos anteriores mencionados se comprueba si la capa se va a expandir o no, después de expandir la capa, se hace una última comparación para saber si se debe de crear una nueva capa en una de las neuronas creadas.

- *Expansión horizontal:*

Se obtiene el vecino más alejado de la neurona con el MQE más grande de la capa, haciendo comprobaciones para elegir el lugar dónde se va a agregar y si es columna o renglón. Una vez que se haya agregado se actualiza el Vecino Más Cercano inicial con la fórmula 12:

$$neighborhood_i = \sqrt[2]{\frac{\max\{x \mid y\}}{2\sqrt[2]{-\log(NR)}}} \quad (12)$$

Dónde:

- x: Tamaño en “x” de la capa.
- y: Tamaño en “y” de la capa.
- max{x | y}: Obtener el máximo de una de las dos variables.

- *Expansión vertical:*

Primero se comprueba que al menos una de las neuronas tenga menos elementos representativos que la súper neurona, después se hace un ciclo para encontrar las neuronas adecuadas con la condición de la fórmula 13:

$si\ MQE_{primerCapa} * TAU\_2\ > MQE_i$ entonces
Agregar nueva capa (13)

Capítulo 5. RESULTADOS Y DISCUSIÓN

Durante el desarrollo de este proyecto de tesis, se analizaron algoritmos de aprendizaje supervisado y no supervisado, tales como SVM y GHSOM, respectivamente. También se estudió e implementó la forma en que el sistema pudiera manejar la retroalimentación de una persona y que los tiempos de ejecución no fueran excesivos. También en la parte final del proyecto se realizaron algunas mejoras, como el guardado completo de la red neuronal, y un diccionario de palabras relacionadas con una polaridad que se incrementa con cada retroalimentación.

La aplicación de software tuvo las siguientes características:

- Independencia del lenguaje de programación en el que se quiera incluir, esto es porque se eligió PERL que es un lenguaje interpretado que se instala fácilmente en cualquier sistema operativo, y que sus rutinas (códigos escritos) pueden ser llamados desde línea de comandos.
- La arquitectura del sistema hace que sea fácilmente escalable, debido a que el sistema se hizo por módulos, dejando la posibilidad a próximas mejoras.
- GHSOM es un algoritmo robusto para el aprendizaje profundo y clasificación de características, cumpliendo con los objetivos del proyecto.

La manera de evaluar el módulo del analizador de sentimientos se basó en la validación de los requerimientos establecidos. Dando como resultado que el módulo presenta una robustez válida para hacer las siguientes tareas:

1. Revisar los tuits y validar la información que contenga la información necesaria para el análisis.
2. Preprocesar la información para obtener la información optimizada.
3. Convertir el texto preprocesado en información útil para la computadora.
4. Obtener la clasificación de cada texto procesado.
5. Guardar el estado de la red neuronal.
6. Cargar el estado de una red neuronal previa.
7. Procesar la retroalimentación obtenida por una persona.

8. Agregar al diccionario las palabras con mayor relevancia en los documentos que se obtuvo retroalimentación.

Al finalizar el desarrollo del software se puso a prueba con comentarios que se habían estado obteniendo durante la codificación del analizador de sentimientos. Debido a que los comentarios no estaban clasificados con anterioridad, la evaluación de la efectividad del software se hizo de manera manual, leyendo el comentario y compararlo con el resultado obtenido por medio del software, de esta forma se obtuvo que la efectividad del desarrollo con una muestra de 278 comentarios tuvo como resultado que en el 84.17 % de los casos los clasificaba correctamente en un tiempo de 120 minutos.

A continuación, se mostrarán un par de comentarios recogidos por el sistema que clasifico correcta e incorrectamente respectivamente, dejando el segundo como un comentario neutro.

1) *"Flipando desde ayer con Fire Emblem Echoes! El mejor remake que he jugado nunca. 🖐🖐🖐 Larga vida a la 3ds!! @NintendoES @Nintenderos".*
2) *"@ZumaGt @NintendoES Es un juegazo, ha sido una buena sorpresa".*

Capítulo 6. CONCLUSIONES

Una de las principales conclusiones es que se logró desarrollar una aplicación de software que implementa técnicas de Inteligencia Artificial como lo son el Análisis de Sentimientos, y las Redes Neuronales Artificiales.

Una de las mayores características que tiene el proyecto es que realiza una categorización de los comentarios realizados por seres humanos en la red social Twitter de forma automatizada, utilizando:

- Aprendizaje no supervisado, empleado para que el sistema no tenga que ser alimentado previamente.
- Ingeniería de software ágil, necesario para una progresión en etapas y cambios no considerados, y
- Técnicas de programación eficientes y útiles para poder minimizar los tiempos de análisis.

Gracias a la arquitectura GHSOM, que es utilizado principalmente en problemas en el que las entradas no están definidas, y que tiene la posibilidad de adaptar su arquitectura con base en los parámetros de aprendizaje y los casos que se analizan, se obtiene como resultado una red neuronal flexible al ambiente.

El análisis de los resultados obtenidos en las pruebas realizadas, sirvió para medir el desempeño del software, el cual fue satisfactorio y teniendo en cuenta que la computadora en la que se realizaron dichas pruebas fue la misma que la utilizada para realizar el desarrollo de la aplicación, se concluye que los tiempos de respuestas que se mencionan en el apartado de resultados, pueden ser mejorados con un mejor hardware.

Cabe mencionar también que para poder mejorar los tiempos del rendimiento de la aplicación se podría utilizar un lenguaje de programación compilado, debido a que el que se utilizó es un lenguaje interpretado y es más lento.

CONCLUSIONES

Por último, este proyecto ha sido para mí como autor, una gran experiencia en cuanto a desarrollo de software debido a que se obtuvo un proyecto con bastantes posibilidades a futuro. Así como también, por haber cumplido con los objetivos que se plantearon al principio del proyecto. Sólo queda esperar que esta tesis sirva para aquellos que deseen usarla como punto de partida para nuevos proyectos.

Capítulo 7. REFERENCIAS

Agarwal, B. & Mittal, N., 2016. *Prominent Feature Extraction for Sentiment Analysis.* s.l.:Springer.

Anon., 2000. *Ciencia médica y ciencia de la complejidad.* [En línea] Available at: http://medicinaycomplejidad.org/pdf/redes/Competitivas.pdf [Último acceso: 14 septiembre 2017].

Anon., 2006. Standard ECMA-334 C# Language Specification. *Ecma International.*

Anon., s.f. [En línea] Available at: http://www.tutorialspoint.com/perl/index.htm

Ayala, A. P., 2005. *Lenguaje Natural: Descripción de las Etapas para su Tratamiento.* México: APA.

Danneman, N. & Heimann, R., 2014. *Social Media Mining with R.* Primera ed. Birmingham: Packt Publishing Ltd. .

Dave, K., Lawrence, S. & Pennock, D. M., 2003. Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. *12th international conference on World Wide* , pp. 519-528.

Doyle, M., 2009. *Begining PHP 5.3.* s.l.:s.n.

Ducket, J., 2011. *HTML & CSS Design and Build Websites.* Indianapolis: John Wiley & Sons, Inc.

Go, A., Huang, L. & Bhayani, R., 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford,* 1(12).

Gómez Coaboy, E. L., 2015. *Red Kohonen.* [En línea] Available at: https://inteligeciatrials.files.wordpress.com/2015/07/7-red-kohonen.pdf [Último acceso: 19 septiembre 2017].

REFERENCIAS

Goodfellow, I., Bengio, Y. & Courville, A., 2016. *Deep Feedforward Networks.* Cambridge: MIT Press.

Hernández Petlachi, R. & Li, X., 2014. *Análisis de sentimiento sobre textos en Español basado en aproximaciones semánticas con reglas lingüísticas.* Gerona, s.n.

Hernández, J. L. O., 2012. El Procesamiento del Lenguaje Natural para extraer conocimiento de la Web, Documentos y Redes sociales.

Hurtado Oliver, L. F. & Pla, F., 2014. *ELiRF-UPV en TASS 2014: Análisis de Sentimientos, Detección de Tópicos y Análisis de Sentimientos de Aspectos en Twitter.*. València, Universitat Politècnica de València.

Jiménez Zafra, S. M., Martínez Cámara, E., Martín Valdivia, M. T. & Ureña López, L. A., 2014. *SINAI-ESMA: An unsupervised approach for Sentiment Analysis in Twitter.* Gerona, s.n.

Lindstrom, L. & Jeffries, R., s.f. Extreme Programming and Agile Software Development Methodologies.

Liu, B., 2015. *Sentiment Analysis Mining Opinions, Sentiments, and Emotions.* Primera ed. Cambridge: Cambridge University Press.

Martínez Cámara, E., Martín Valdivia, M. T. & Ureña, L. A., s.f. Análisis de sentimientos. *Actas IV Jornadas,* pp. 61-63.

Navarrete García, J., 2003. *Mejora en el algoritmo de segmentación para el reconocimiento de caracteres de telegramas escritos por el Gral. Porfirio Díaz.* Cholula: s.n.

Pak, A. & Paroubek, P., 2010. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. *LREC.*

Pang, B. & Lee, L., 2008. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval,* Volumen 2, pp. 1-135.

Peláez Chávez, N. & Borja Macías, V., 2012. *Aprendizaje no supervisado y el algoritmo wake-sleep en redes neuronales.* HUAJUAPAN DE LEON: s.n.

REFERENCIAS

Rauber, A., Merkl, D., Dittenbach, M. & Pampalk, E., 2000. *Department of Software Technology Vienna University of Technology.* [En línea] Available at: http://www.ifs.tuwien.ac.at/~andi/ghsom/description.html [Último acceso: 24 mayo 2016].

Ray, S., 2017. *Analytics Vidhya.* [En línea] Available at: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/
[Último acceso: 24 enero 2018].

Read, J., 2005. Using Emoticons to reduce Dependency in Machine Learning Techniques for Sentiment Classification. *ACL Student Research Workshop,* pp. 43-48.

San Vicente Roncal, I. & Saralegi Urizar, X., 2014. *Looking for Features for Supervised Tweet Polarity Classification.* Girona, s.n.

Sidorov, G. y otros, 2013. Empirical study of machine learning based approach for opinion mining in tweets. *11th Mexican international conference on Advances in Artificial Intelligence,* Volumen 1, pp. 1-14.

Valencia Reyes, M. A., Yáñez Márquez, C. & Sánchez Fernández, L. P., 2006. *Algoritmo Backpropagation para Redes Neuronales: conceptos y aplicaciones.* Ciudad de México: s.n.

Vilares, D., Doval, Y., Gómez Rodriguez, C. & Pardo, M. Á. A., 2014. *LyS at TASS 2014: A Prototype for Extracting and Analysing.* Girona, s.n.

Capítulo 8. ANEXOS

A continuación, se anexa el código de las funciones principales para el entrenamiento de la red neuronal.

```
  sub train{
          my $self = shift;
          my $path = $Globals::savePath;
          my $prefix = $Globals::HTML_PREFIX;
          my $gid = $self->{gid};
          my $level = $self->{level};
          my $pos1 = $self->{superPos}->[0];
          my $pos2 = $self->{superPos}->[1];
          my                          $mqeName                          =
"$FindBin::Bin/$path/$prefix\_$gid\_$level\_$pos1\_$pos2.mqe";


          $self->{stupidity} = $Globals::TAU_1;
          $self->{ini_learnrate} = $Globals::INITIAL_LEARNRATE;
          $self->{learnrate} = $Globals::INITIAL_LEARNRATE;
          $self->{min_learnrate} = $Globals::MIN_LEARNRATE;
          $self->{min_neighbourhood} = $Globals::MIN_NEIGHBOURHOOD;
          $self->{currentCycle} = 0;
          $self->{ini_neighbourhood}    =    sqrt(($self->{x}>$self-
>{y}?$self->{x}:$self->{y})/(2*sqrt(-1*log($Globals::NR))));
          $self->{neighbourhood} = $self->{ini_neighbourhood};
          my $run = 'true';
          use Data::Dumper;
          use FindBin;
          my $increase = 1;
          while ($run){
```

```
            $self->{currentCycle}++;

            #Obtener el siguiente patrón.

            my    $num    =    Globals->getIntRandom({limit=>$self-
>{dataItems}->size()}) % $self->{dataItems}->size();

            $currentDataItem = $self->{dataItems}->elementAt($num);

            if($self->{MQE} <= $stupy || $Globals::MAX_CYCLES > 0 &&
$self->{currentCycle}   >=   ($Globals::MAX_CYCLES*$self->{dataItems}-
>size())){

                $run = 1==2;

            }

            $winnerDist = MAX_DOUBLE;

            $winner = [];

            for my $i (0..$self->{y}){

                for my $j (0..$self->{x}){

                    my  $currDist  =  $self->{neuronMap}->[$j][$i]-
>calcDist($currentDataItem);

                    if($currDist < $winnerDist){

                        $winnerDist = $currDist;

                        $winner->[0] = $j;

                        $winner->[1] = $i;

                    }

                }

            }

            $self-
>adaptWeights({winner=>$winner,di=>$currentDataItem});

            undef $winner;

            $self->{learnrate}    =    $self->{ini_learnrate}    *
$Globals::E         **         (-1*(($self->{currentCycle}         %
($Globals::EXPAND_CYCLES*$self->{dataItems}->size()))))/$self-
>{STRETCH_PARAM_LEARN})+0.0001;
```

```
            $self->{neighbourhood} = $self->{ini_neighbourhood} *
$Globals::E         **         (-1*(($self->{currentCycle}         %
($Globals::EXPAND_CYCLES*$self->{dataItems}->size()))/$self-
>{STRETCH_PARAM_NEIGHB})+0.55;

            if($Globals::printMQE){

                for my $i (0..$self->{y}){

                    for my $j(0..$self->{x}){

                        $self->{neuronMap}->[$j][$i]-
>clearRepresentingDataItems();

                    }

                }

                $self->calcMQE();

            }

            if($self->{currentCycle} % ($Globals::EXPAND_CYCLES *
$self->{dataItems}->size()) == 0){

                if(!$Globals::printMQE){

                    for my $i (0..$self->{y}){

                        for my $j(0..$self->{x}){

                            $self->{neuronMap}->[$j][$i]-
>clearRepresentingDataItems();

                        }

                    }

                    $self->calcMQE();

                }

            }

            my $stupy = $self->{stupidity} * $self->{superMQE};

            $m = $Globals::MAX_CYCLES*$self->{dataItems}->size();

            if($self->{MQE} <= $stupy || $Globals::MAX_CYCLES > 0 &&
$self->{currentCycle}   >=   ($Globals::MAX_CYCLES*$self->{dataItems}-
>size())){

                $run = 1==2;
```

```
            }else{
                my $dissNeigbour = $self->getMaxDissNeighbour({n=>$self->{MQENeuron}});
                if($increase){
                    if($self->{MQENeuron}->[0] > $dissNeigbour->[0]){
                        $self->insertColumn($self->{MQENeuron}->[0]);
                    }elsif($self->{MQENeuron}->[0] < $dissNeigbour->[0]){
                        $self->insertColumn($dissNeigbour->[0]);
                    }
                    if($self->{MQENeuron}->[1] > $dissNeigbour->[1]){
                        $self->insertRow($self->{MQENeuron}->[1]);
                    }elsif($self->{MQENeuron}->[1] < $dissNeigbour->[1]){
                        $self->insertRow($dissNeigbour->[1]);
                    }
                }
                $self->{learnrate} = $self->{ini_learnrate};
                $self->{ini_neighbourhood} = sqrt(($self->{x}>$self->{y}?$self->{x}:$self->{y})/(2*sqrt(-1*log($Globals::NR))));
                $self->{neighbourhood} = $self->{ini_neighbourhood};
                undef $dissNeigbour;
            }
        }
        my $r = 'true';
        my $exp = 'true';
```

```
            if($level > 1){
                my $ii = 0;
                while($ii < $self->{y} && $r){
                    my $jj = 0;
                    while ($jj < $self->{x} && $r){
                        if($self->{superNeuron}-
>representsMultiDataItems()    ==    $self->{neuronMap}->[$jj][$ii]-
>representsMultiDataItems()){
                            $exp = (1==2);
                            $r = (1==2);
                        }elsif($self->{superNeuron}-
>representsMultiDataItems()    >    $self->{neuronMap}->[$jj][$ii]-
>representsMultiDataItems()){
                            $exp = 'true';
                            $r = (1==2);
                        }
                        $jj++;
                    }
                    $ii++;
                }
            }


            if(defined $Globals::FAST && $Globals::FAST){
                my $random = rand();
                if($random <= $Globals::probabilityIncrease){
                    $increase = (1==2);
                }else{
                    $increase = 1;
                }
            }
```

```
            if($exp){
                for my $i (0..$self->{y}){
                    for my $j (0..$self->{x}){
                        if((Globals->getFirstLayerMap()->{superMQE}   *
$Globals::TAU_2) < $self->{neuronMap}->[$j][$i]->{MQE}){
                            if($Globals::ORIENTATION){
                                my ($UL,$UR,$LL,$LR);
                                $UL = $UR = $LL = $LR = [];
                                $ref               =               $self-
>getNewWeights({xPos=>$j,yPos=>$i,UL=>$UL,UR=>$UR,LL=>$LL,LR=>$LR});
                                $UL = $ref->[0];
                                $UR = $ref->[1];
                                $LL = $ref->[2];
                                $LR = $ref->[3];
                                $self->{neuronMap}->[$j][$i]->addMap({
                                    sn => $self->{neuronMap}->[$j][$i],
                                    MQE      =>      $self->{neuronMap}-
>[$j][$i]->{MQE},
                                    inlevel => $level + 1,
                                    posX => $j,
                                    posY => $i,
                                    ULweight => $UL,
                                    URweight => $UR,
                                    LLweight => $LL,
                                    LRweight => $LR
                                });
                            }else{
                                $self->{neuronMap}->[$j][$i]->addMap({
                                    sn => $self->{neuronMap}->[$j][$i],
```

```
                                    MQE     =>     $self->{neuronMap}-
>[$j][$i]->{MQE},
                                    inlevel => $level + 1,
                                    posX => $j,
                                    posY => $i,
                                    sizeX => $Globals::INITIAL_X_SIZE,
                                    sizeY => $Globals::INITIAL_Y_SIZE
                                });
                            }
                            Globals->addLayer({level => $level+1,nl =>
$self->{neuronMap}->[$j][$i]->getMap()});
                        }
                    }
                }
            }
        }
  sub getNewWeights{
            my $self = shift;
            my $params = shift;
            my $xPos = $params->{xPos};
            my $yPos = $params->{yPos};
            my $UL = $params->{UL};
            my $UR = $params->{UR};
            my $LL = $params->{LL};
            my $LR = $params->{LR};
            my $twgt = $self->{neuronMap}->[$xPos][$yPos]->{weights};
            my
($nUL,$nU,$nL,$nBL,$nB,$nBR,$nUR,$nR,$dUL,$dUR,$dLL,$dLR);
            $nUL = $nU = $nL = $nBL = $nB = $nUR = $nR = undef;
```

```
            if($xPos > 0 && $yPos > 0){
                $nUL      =      $self->{neuronMap}->[$xPos-1][$yPos-1]-
>{weights};
            }
            if($yPos > 0){
                $nU = $self->{neuronMap}->[$xPos][$yPos-1]->{weights};
            }
            if($xPos > 0){
                $nL = $self->{neuronMap}->[$xPos-1][$yPos]->{weights};
            }
            if($xPos > 0 && $yPos < $self->{y}-1){
                $nBL      =      $self->{neuronMap}->[$xPos-1][$yPos+1]-
>{weights};
            }
            if($yPos < $self->{y}-1){
                $nB = $self->{neuronMap}->[$xPos][$yPos+1]->{weights};
            }
            if($xPos < $self->{x}-1 && $yPos < $self->{y}-1){
                $nBR    =       $self->{neuronMap}->[$xPos+1][$yPos+1]-
>{weights};
            }
            if($xPos < $self->{x}-1 && $yPos > 0){
                $nUR      =      $self->{neuronMap}->[$xPos+1][$yPos-1]-
>{weights};
            }
            if($xPos < $self->{x}-1){
                $nR = $self->{neuronMap}->[$xPos+1][$yPos]->{weights};
            }
            if($xPos == 0 && $yPos == 0){#UpperL
                $dUL = $twgt;
```

```
                $dUR     =     Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorDiffMean({a=>$nR,b=>$twgt})});
                $dLL     =     Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorDiffMean({a=>$nB,b=>$twgt})});
                $dLR     =     Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorAdd3Mean({
                    Globals->vectorDiff({a=>$nR,b=>$twgt}),
                    Globals->vectorDiff({a=>$nBR,b=>$twgt}),
                    Globals->vectorDiff({a=>$nB,b=>$twgt})
                })});
            }elsif($xPos == $self->{x}-1 && $yPos == 0){#UpperR
                $dUL     =     Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorDiffMean({a=>$nL,b=>$twgt})});
                $dUR = $twgt;
                $dLL     =     Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorAdd3Mean({
                    Globals->vectorDiff({a=>$nL,b=>$twgt}),
                    Globals->vectorDiff({a=>$nBL,b=>$twgt}),
                    Globals->vectorDiff({a=>$nB,b=>$twgt})
                })});
                $dLR     =     Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorDiffMean({a=>$nB,b=>$twgt})});
            }elsif($xPos==0 && $yPos == $self->{y}-1){#LowerL
                $dUL     =     Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorDiffMean({a=>$nU,b=>$twgt})});
                $dUR     =     Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorAdd3Mean({
                    Globals->vectorDiff({a=>$nU,b=>$twgt}),
                    Globals->vectorDiff({a=>$nUL,b=>$twgt}),
                    Globals->vectorDiff({a=>$nR,b=>$twgt})
                })});
```

```
            $dLL = $twgt;
            $dLR     =     Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorDiffMean({a=>$nR,b=>$twgt})});
          }elsif($xPos == $self->{x} - 1 && $yPos == $self->{y} -
1){#LowerR
            $dUL     =     Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorAdd3Mean({
                Globals->vectorDiff({a=>$nUL,b=>$twgt}),
                Globals->vectorDiff({a=>$nU,b=>$twgt}),
                Globals->vectorDiff({a=>$nL,b=>$twgt})
            })});
            $dUR     =     Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorDiffMean({a=>$nU,b=>$twgt})});
            $dLL     =     Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorDiffMean({a=>$nL,b=>$twgt})});
            $dLR = $twgt;
          }elsif($yPos == 0 && $xPos > 0 && $xPos < $self->{x}-
1){#UpperB
            $dUL     =     Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorDiffMean({a=>$nL,b=>$twgt})});
            $dUR     =     Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorDiffMean({a=>$nR,b=>$twgt})});
            $dLL     =     Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorAdd3Mean({
                Globals->vectorDiff({a=>$nL,b=>$twgt}),
                Globals->vectorDiff({a=>$nBL,b=>$twgt}),
                Globals->vectorDiff({a=>$nB,b=>$twgt})})
            });
            $dLR     =     Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorAdd3Mean({
                Globals->vectorDiff({a=>$nR,b=>$twgt}),
                Globals->vectorDiff({a=>$nBR,b=>$twgt}),
```

```
                Globals->vectorDiff({a=>$nB,b=>$twgt})
            })});
        }elsif($xPos == $self->{x} -1 && $yPos > 0 && $yPos < $self-
>{y} -1 ){#RightB
            $dUL    =    Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorAdd3Mean({
                Globals->vectorDiff({a=>$nL,b=>$twgt}),
                Globals->vectorDiff({a=>$nUL,b=>$twgt}),
                Globals->vectorDiff({a=>$nU,b=>$twgt})
            })});
            $dUR    =    Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorDiffMean({a=>$nU,b=>$twgt})});
            $dLL    =    Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorAdd3Mean({
                Globals->vectorDiff({a=>$nL,b=>$twgt}),
                Globals->vectorDiff({a=>$nBL,b=>$twgt}),
                Globals->vectorDiff({a=>$nB,b=>$twgt})})
            });
            $dLR    =    Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorDiffMean({a=>$nL,b=>$twgt})});
        }elsif($yPos == $self->{y} -1 && $xPos > 0 && $xPos < $self-
>{x} -1){#LowerB
            $dUL    =    Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorAdd3Mean({
                Globals->vectorDiff({a=>$nL,b=>$twgt}),
                Globals->vectorDiff({a=>$nUL,b=>$twgt}),
                Globals->vectorDiff({a=>$nU,b=>$twgt})
            })});
            $dUR    =    Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorAdd3Mean({
                Globals->vectorDiff({a=>$nR,b=>$twgt}),
```

```
                Globals->vectorDiff({a=>$nUR,b=>$twgt}),
                Globals->vectorDiff({a=>$nU,b=>$twgt})})
            });
            $dLL    =    Globals->vectorAdd({a=>$twgt,b=>Globals->vectorDiffMean({a=>$nL,b=>$twgt})});
            $dLR    =    Globals->vectorAdd({a=>$twgt,b=>Globals->vectorDiffMean({a=>$nR,b=>$twgt})});
        }elsif($xPos == 0 && $yPos > 0 && $yPos < $self->{y} -1){#LeftB
            $dUL    =    Globals->vectorAdd({a=>$twgt,b=>Globals->vectorDiffMean({a=>$nU,b=>$twgt})});
            $dUR    =    Globals->vectorAdd({a=>$twgt,b=>Globals->vectorAdd3Mean({
                Globals->vectorDiff({a=>$nR,b=>$twgt}),
                Globals->vectorDiff({a=>$nUR,b=>$twgt}),
                Globals->vectorDiff({a=>$nU,b=>$twgt})})
            });
            $dLL    =    Globals->vectorAdd({a=>$twgt,b=>Globals->vectorDiffMean({a=>$nB,b=>$twgt})});
            $dLR    =    Globals->vectorAdd({a=>$twgt,b=>Globals->vectorAdd3Mean({
                Globals->vectorDiff({a=>$nR,b=>$twgt}),
                Globals->vectorDiff({a=>$nBR,b=>$twgt}),
                Globals->vectorDiff({a=>$nB,b=>$twgt})
            })});
        }elsif($xPos > 0 && $xPos < $self->{x} -1 && $yPos > 0 && $yPos < $self->{y} -1){#M
            $dUL    =    Globals->vectorAdd({a=>$twgt,b=>Globals->vectorAdd3Mean({
                Globals->vectorDiff({a=>$nL,b=>$twgt}),
                Globals->vectorDiff({a=>$nUL,b=>$twgt}),
```

```
                Globals->vectorDiff({a=>$nU,b=>$twgt})})
            });
            $dUR    =    Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorAdd3Mean({
                Globals->vectorDiff({a=>$nR,b=>$twgt}),
                Globals->vectorDiff({a=>$nBR,b=>$twgt}),
                Globals->vectorDiff({a=>$nB,b=>$twgt})})
            });
            $dLL    =    Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorAdd3Mean({
                Globals->vectorDiff({a=>$nL,b=>$twgt}),
                Globals->vectorDiff({a=>$nBL,b=>$twgt}),
                Globals->vectorDiff({a=>$nB,b=>$twgt})})
            });
            $dLR    =    Globals->vectorAdd({a=>$twgt,b=>Globals-
>vectorAdd3Mean({
                Globals->vectorDiff({a=>$nR,b=>$twgt}),
                Globals->vectorDiff({a=>$nBR,b=>$twgt}),
                Globals->vectorDiff({a=>$nB,b=>$twgt})
            })});
        }
        for my $i (0..$Globals::vectorLength){
            $UL->[$i] = $dUL->[$i];
            $UR->[$i] = $dUR->[$i];
            $LL->[$i] = $dLL->[$i];
            $LR->[$i] = $dLR->[$i];
        }
        return [$UL,$UR,$LL,$LR];
    }
```