Aprendizaje Profundo y Neuroevolución para el Análisis de Sentimientos en Tweets Escritos en Español Mexicano

Maestría en Inteligencia Artificial

José Clemente Hernández Hernández
S19019835

Instituto de Investigaciones en Inteligencia Artificial
Universidad Veracruzana

Director: Dr. Guillermo de Jesús Hoyos Rivera
Co-director: Dr. Efrén Mezura Montes

03.12.2021

Agradecimientos

"Si he logrado ver más lejos ha sido porque he subido a hombros de gigantes". Una frase de Isaac Newton que he repetido y repetiré siempre que me acuerde de aquellas personas que me han apoyado a subir tantos peldaños como los que, hasta ahora, he recorrido. Esto es un agradecimiento a la confianza y comprensión que me han regalado todos mis allegados en cualquier ámbito, tanto personal, social, académico y familiar.

A mi amigo, compañero, profesor y asesor de tesis, Dr. Guillermo de Jesús Hoyos Rivera, quien desde hace algunos años, como aquel que pastorea a sus ovejas, lo ha hecho conmigo en el campo de la Ciencia. Que siempre estuvo ahí resolviéndome dudas de cualquier tema y regalándome datos sumamente interesantes. Sin su apoyo incondicional, esto no hubiera sido posible.

A mi profesor y asesor de tesis, Dr. Efrén Mezura Montes, quien con su conocimiento en mi área de interés, he podido concluir satisfactoriamente mi tesis. Sin todos los consejos que me dio, hasta el último día, no hubiera logrado tener tanto saber como el que tengo actualmente.

Ambos son altamente ejemplos a seguir hoy y siempre.

A mis compañeros de generación, Toño R., Aarón J., David H. y Gustavo V., que me acompañaron en el camino del conocimiento haciéndolo más atractivo y lleno de muchos retos que entre nosotros mismos nos pusimos. Y como nos dijo una vez Gustavo: "no son solo unos compañeros, son mis amigos y mis hermanos"

A mis amigos, compañeros de vida y hermanos que se eligen, Abraham T., Gustavo V., Yessica Z., Eduardo L., Leopoldo A., Fernanda O. y Stephanie A. (y muchos más), no se mueran nunca

—muchas risas—, nunca me vayan a faltar. Ustedes son grandes y lograrán lo que se propongan.

A mi familia, tío Fabián, papá Clemente, mamá Nora, y mis hermanos, Emmanuel y David. Con el apoyo tan apreciado y superior que nos hemos dado entre todos, logramos todo, tener unidad y armonía.

Y a todos los que están, por mantenerse, a los que estuvieron, por encontrar otros caminos, y a los que estarán... aunque no los conozca aún.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por el apoyo brindado a través de una beca para realizar estudios de posgrado en la Universidad Veracruzana.

José Clemente

Resumen

El Análisis de Sentimientos, perteneciente al área del Procesamiento del Lenguaje Natural, utiliza diversas técnicas de la Inteligencia Artificial para el descubrimiento de sentimientos en texto. Normalmente, el texto es generado por usuarios de alguna red social digital. Este trabajo plantea ejecutar un Análisis de Sentimientos en tweets escritos en Español Mexicano utilizando, para esto, técnicas de Aprendizaje Profundo y Neuroevolución. La propuesta incluye un algoritmo genético de diseño propio, llamado *DeepNEWT*, el cual usa la similitud de las redes neuronales convolucionales para ejecutar el operador de cruza. Asimismo, este algoritmo genético también busca los mejores pesos de los filtros convolucionales y de la capa clasificadora. Los tweets son representados a través de dos diferentes modelos: *Word2Vec* y *BERT*, ambos fiables para ser usados con redes neuronales convolucionales. Se demuestra que una red neuronal convolucional generada con el algoritmo de Neuroevolución *DeepNEWT* encuentra una estructura de menor complejidad y número de parámetros comparada con una extraída de la revisión de la literatura. Dicha red neuronal convolucional es hasta un 96 % menor en cuanto al número de parámetros, entre los que se encuentran los pesos y sesgos convolucionales y de la capa clasificadora. Además, las redes neuronales convolucionales generadas por *DeepNEWT* fueron entrenadas con *retroprpagación* y obtuvieron resultados mejores que las técnicas de Aprendizaje Máquina tradicional para el Análisis de Sentimientos.

Índice general

Resumen III

Índice de Figuras VI

Índice de Tablas VIII

Lista de Abreviaciones X

1 Introducción 1

1.1 Antecedentes . . . 2
1.2 Propuesta . . . 3
1.3 Hipótesis . . . 4
1.4 Justificación y Motivación . . . 4
1.5 Objetivos . . . 4
1.5.1 Objetivo General . . . 4
1.5.2 Objetivos Específicos . . . 5

2 Marco Teórico 6

2.1 Análisis de Sentimientos en Twitter . . . 6
2.1.1 La opinión a través de un tweet . . . 6
2.2 Aprendizaje Automático para el Análisis de Sentimientos 7
2.2.1 Clasificador *Máquina de Soporte Vectorial* . . . 8
2.2.2 Clasificador *Ingenuo Bayesiano* . . . 8
2.3 Aprendizaje Profundo en el Análisis de Sentimientos . . 9
2.3.1 Redes Neuronales y la *retropropagación* . . . 9
2.3.2 Redes Neuronales Convolucionales . . . 12
2.3.3 Modelo *Word2Vec*: *Continuous Bag-Of-Words* y *Skip-Gram* . . . 14
2.3.4 Modelo *BERT*: *Bidirectional Encoder Representations from Transformers* . . . 15

2.4 Computación Evolutiva 16
2.4.1 Algoritmos Genéticos 16
2.4.2 Neuroevolución 18

3 Marco Referencial **20**

4 Propuesta **24**
4.1 Metodología 24
4.2 Datos 24
4.3 Limpieza de los datos 25
4.4 Transformación de los datos 25
4.5 Red Neuronal Convolucional para el Análisis de Sentimientos 27
4.6 Algoritmo Genético para la Evolución de una Red Neuronal Convolucional para el Análisis de Sentimientos 28
4.6.1 Representación 30
4.6.1.1 Capa de bloques *CNP* 30
4.6.1.2 Similitud de RNCs 32
4.6.1.3 Bloque *last-CNP* 33
4.6.1.4 Capa *FC* 35
4.6.1.5 Objetivo de la representación 35
4.6.2 Operador de selección y cruza 36
4.6.2.1 Operador de cruza del bloque *CNP* . . . 36
4.6.2.2 Operadores de cruza del bloque *last-CNP* y de la capa *FC* 36
4.6.3 Operador de mutación 37
4.6.3.1 Secuencias de mutaciones 38
4.6.3.2 Operador de mutación de los bloques *CNP* y *last-CNP* 39
4.6.3.3 Operador de mutación de la capa *FC* . . 41
4.6.4 Recomposición de las soluciones 41

5 Experimentos y resultados **43**
5.1 Datos utilizados 43
5.2 Ambiente de Ejecución 43
5.3 Forma de evaluar los resultados 44

5.4 Resultados 44
5.4.1 Experimentos preliminares con *DeepNEWT* 46
5.4.2 Búsqueda de RNC para la mejora de la precisión con *DeepNEWT* 48
5.4.3 AM tradicional y entrenamiento de las estructuras preliminares encontradas por *DeepNEWT* para *BERT* 50
5.4.4 AM tradicional y *DeepNEWT* para *Word2Vec* . . . 55
5.4.5 Estructuras de RNC obtenidas por *DeepNEWT* . . 59

6 Conclusiones y Trabajo futuro **62**

Bibliografía **I**

Índice de Figuras

2.1 Ejemplo de una opinión con sus elementos 7
2.2 Estructura de una RNA, donde X es la capa de entrada, O representa la oculta, Y la de salida y W son los pesos que hay entre las capas 11
2.3 Partes y funcionamiento de la convolución 12
2.4 Capa convolucional con varios canales de entrada X y de salida Y 13
2.5 Estructura de un autoencoder 15
2.6 Esquema del funcionamiento de un AG (basado en Fogel (2000)) 17

4.1 Estructura de un tweet después de su transformación a través de *Word2Vec* o *BERT* 26
4.2 Estructura de la RNC utilizada para clasificación de texto 27
4.3 Estructura y flujo de la información a través del filtro convolucional de $5 \times m$ 28
4.4 Estructura predeterminada de una RNC dentro de DeepNEWT 30
4.5 Estructura de una capa de bloques CNP 30
4.6 Cambio de la dimensionalidad después de la operación de convolución 31
4.7 Cambio de la dimensionalidad después de la operación de pooling 32
4.8 Flujo de la información dentro del bloque last-CNP . . . 33
4.9 Cambio de la dimensionalidad después de la operación de convolución en el bloque last-CNP 34
4.10 Cambio de la dimensionalidad después de la operación de pooling en el bloque last-CNP 35
4.11 Flujo de la información dentro de la capa FC 35

4.12 Cruza de elementos en el bloque CNP 37
4.13 Esquema de momentos en el operador de mutación . . . 37

5.1 Convergencia de la precisión en los experimentos preliminares $pre2$ y $pre3$ 47
5.2 Convergencia de precisión con diferente número de padres seleccionados, y su gráfica de convergencia con respecto a la precisión media 50
5.3 Convergencia de precisión de *DeepNEWT* con respecto al conjunto de entrenamiento utilizando los parámetros de los experimentos preliminares, $pre2$ y $pre3$ 56
5.4 Estructura de RNC encontrada por *DeepNEWT* en el experimento $pre2$ 59
5.5 Estructura de RNC encontrada por *DeepNEWT* en el experimento $pre3$ 60

Índice de Tablas

2.1 Palabras y su correspondiente contexto en la frase de ejemplo 14

3.1 Estudios en el ámbito simbólico 21
3.2 Estudios en el ámbito del aprendizaje supervisado con técnicas tradicionales del AM; los símbolos $+$, $-$, son las polaridades sentimentales positiva y negativa 21
3.3 Estudios en el ámbito del aprendizaje profundo; los símbolos $+$, $-$ y $/$, son las polaridades sentimentales positiva, negativa y neutral 22
3.4 Estudios en el ámbito del aprendizaje profundo utilizando *BERT*; los símbolos $+$, $-$ y $/$, son las polaridades sentimentales positiva, negativa y neutral . 23
3.5 Estudios en el ámbito de la Neuroevolución 23

4.1 Número de tweets por polaridad. 25
4.2 Números de similitud y sus correspondiente no linealidad y operación de pooling 33
4.3 Esquema de mutaciones a ejecutar en los movimientos y en cada bloque, se agrega el tipo de mutación; el símbolo $+/-$ significa incrementar o reducir 40

5.1 Número de tweets en la base de datos original, después de seleccionar aleatoriamente datos, para lidiar con el desbalanceo de clases 43
5.2 Esquema de experimentos realizados y sus correspondientes resultados 45
5.3 Parámetros utilizados en la ejecución de *DeepNEWT* . . . 47
5.4 Parámetros de los experimentos y sus respectivos porcentajes de precisión finales 49

5.5 Parámetros del experimeto es de $pre2$ y $pre3$ con *BERT* . 51
5.6 Porcentajes de precisión de los datos correctamente clasificados de las 10 particiones de prueba de los experimentos de AM tradicional y la estructura preliminar $pre2$ de RNC ajustada para *BERT* 52
5.7 Porcentajes de precisión de los datos correctamente clasificados de las 10 particiones de prueba de los experimentos de AM tradicional y la estructura preliminar $pre3$ de RNC ajustada para *BERT* 53
5.8 Comparación de los clasificadores de AM tradicional con los experimentos es de $pre2$; los valores en negritas muestran diferencias significativas cuando se compara la estructura $pre2$ contra un clasificador tradicional (MSV, IB o RNC) 54
5.9 Comparación de los clasificadores de AM tradicional con los experimentos es de $pre3$; los valores en negritas muestran diferencias significativas cuando se compara la estructura $pre3$ contra un clasificador tradicional (MSV, IB o RNC) 55
5.10 Porcentajes de precisión de los datos correctamente clasificados de las 5 particiones de entrenamiento durante la búsqueda de RNC mediante *DeepNEWT* en la CV 56
5.11 Porcentajes de precisión de los datos correctamente clasificados de las 5 particiones de prueba 57
5.12 Comparación de los resultados de clasificación en el conjunto de prueba de los clasificadores de AM tradicional, la RNC y los experimentos de 58
5.13 Estructuras resultantes con sus respectivos elementos y sus correspondientes valores de los experimentos $pre2$ y $pre3$ 60

Lista de Abreviaciones

AG	Algoritmos Genéticos
AGs	Algoritmos Evolutivos
AM	Aprendizaje Máquina
ANOVA	Análisis de Varianza
AP	Aprendizaje Profundo
AS	Análisis de Sentimientos
ASu	Aprendizaje Supervisado
BERT	Bidirectional Encoder Representations from Transformers
CBOW	Continuous Bag-of-Words
CE	Computación Evolutiva
CEOA-UV	Centro de Estudios de Opinión y Análisis de la Universidad Veracruzana
CMC	Comunicación Mediada por Computadora
CV	Cross-validation
DeepNEWT	Deep NeuroEvolution of Weights and Topologies
ECM	Error Cuadrático Medio
EM	Entropía Máxima
IB	Ingenuo Bayesiano
LSTM	Long Short-Term Memory
MO	Minería de Opiniones
MSV	Máquina de Soporte Vectorial
NB	Naïve-Bayes
NE	Neuroevolución

NEAT	NeuroEvolution of Augmenting Topologies
PG	Programación Genética
PReLU	Parametric Rectified Linear Unit
ReLU	Rectified Linear Unit
RNA	Redes Neuronales Artificiales
RNC	Redes Neuronales Convolucionales
RNM	Redes Neuronales Multicapa
RNR	Redes Neuronales Recurrentes
RPP	Redes de Promediación Profunda
SEPLN	Sociedad Española para el Procesamiento del Lenguaje Natural
SG	Skip-Gram
Sigm	Sigmoidal
TanH	Tangente Hiperbólica
TASS	Taller de Análisis de Sentimientos de la Sociedad Española para el Procesamiento del Lenguaje Natural
VC	Visión por Computadora

1. Introducción

El Análisis de Sentimientos (AS) hace uso de técnicas de Aprendizaje Máquina (AM) tomando como base diccionarios de palabras pre-etiquetadas en diferentes emociones o utilizando su significado simbólico como sus sinónimos o el campo semántico, aunando los clasificadores tradicionales como las Máquinas de Soporte Vectorial (MSV), Naïve-Bayes (NB), o Ingenuo Bayesiano (IB) y el clasificador de Entropía Máxima (EM), para el etiquetado automático, a nivel de documento, sentencia o aspecto de una *opinión* (S. Sun *et al.*, 2017), en diferentes polaridades sentimentales como la positiva y negativa. Las opiniones pueden ser etiquetadas previamente con una polaridad, haciendo que esta parte sea pre-procesada por expertos humanos.

Asimismo, y debido al incremento en la necesidad de mejorar el desempeño de clasificación, otras herramientas como las del Aprendizaje Profundo (AP) se han utilizado para experimentar en el AS, en el cual se reporta que en varias ocasiones el AP sobrepasa los niveles de desempeño de las técnicas tradicionales (Ain *et al.*, 2017; Yadav & Vishwakarma, 2020; Young *et al.*, 2018; Zhang *et al.*, 2018), haciendo uso, por ejemplo, de Redes Neuronales Convolucionales (RNC), Redes Neuronales Recurrentes (RNR) y Redes Neuronales Recurrentes de tipo *Long Short-Term Memory* (LSTM).

La idea principal de la propuesta surge como derivado del poco o nulo agregado de la Neuroevolución en el AS. Se piensa que estas técnicas pueden mejorar el desempeño de clasificación y, además, que se pueden llegar a obtener mejores estructuras, en cuanto a redes neuronales, con respecto a las que comúnmente son usadas.

La Neuroevolución (NE) es el conjunto de herramientas de Computación Evolutiva (CE) que hacen la búsqueda de las mejores redes neuronales, o más cercanas a las estructuras óptimas, así como el entrenamiento de los pesos de las conexiones entre neuronas (K. O. Stanley, Clune *et al.*, 2019). Estas técnicas pueden ser usadas sobre redes neuronales *feed-forward* a través del algoritmo NEAT (NeuroEvolution of Augmenting Topologies) (K. O. Stanley & Miikkulainen, 2002), con sus agregados HyperNEAT

(Hypercube-based NEAT) (Gauci & Stanley, 2007) y ES-HyperNEAT (Evolvable-substrate HyperNEAT) (Risi & Stanley, 2012), para redes más profundas, así como para RNC el algoritmo EvoCNN (Evolving deep Convolutional Neural Networks) (Y. Sun *et al.*, 2020), entre otros (Galván & Mooney, 2020; Martinez *et al.*, 2021).

El objetivo de esta tesis es el uso de técnicas de NE para mejorar el desempeño de clasificación en el AS de tweets escritos en Español Mexicano.

1.1. Antecedentes

El AS tiene sus bases en la Comunicación Mediada por Computadora (CMC), que se define como la comunicación e iteracciones sociales que se dan a través de una computadora (Herring, 1996), sucediendo por medio del correo electrónico, aplicaciones de mensajería instantánea o redes sociales digitales.

Este tipo de análisis se desarrolla, comúnmente, sobre las redes sociales digitales, salud en general, así como en la psicología (Abualigah *et al.*, 2020; Hu & Flaxman, 2018; Singh *et al.*, 2020), haciendo del AS una parte importante en el ámbito social.

De esta manera, varios estudios se centran en desarrollar técnicas estadísticas y de Inteligencia Artificial para localizar el sentimiento predominante de los textos (ver Sección 3), haciendo uso, por ejemplo, de AP y otras técnicas tradicionales del PLN.

Por el lado de las técnicas de AP, varios estudios han intentado satisfactoriamente realizar AS sobre textos escritos en Inglés, sin embargo, el sentido de las opiniones puede cambiar de un idioma a otro. En el caso del Español Mexicano, es poco el trabajo realizado teniendo por objetivo llevar a cabo el AS usando AP.

En el AS, una de las técnicas más usadas son las RNC, que fueron creadas en el área de la Visión por Computadora (VC), que han sido utilizadas para resolver problemas en muchas áreas de estudio (Hatcher & Yu, 2018; LeCun *et al.*, 2015).

En el mismo sentido, en el AS para tweets escritos en Español Mexicano, se han encontrado estudios que analizan la agresividad

(Nina-Alcocer *et al.*, 2019; Penãloza, 2020), sin embargo, las arquitecturas de redes neuronales son buscadas en la literatura o son creadas a mano.

Por lo anterior, este trabajo se enfoca, primero que nada, en hacer una revisión de la literatura sobre las arquitecturas de RNC que existen para AS en otros idiomas, incluyendo el Español de España, y los algoritmos para buscar una red altamente competitiva, que muestre mejores resultados de clasificación en la detección del sentimiento predominante de tweets escritos en Español Mexicano (ver Sección 3).

1.2. Propuesta

La propuesta que presenta este trabajo de investigación toma como base dos técnicas tradicionales, seleccionadas por su uso recurrente en el Estado del Arte, la MSV y el IB, a los cuales se agrega una RNC creada a mano, con una estructura extraída de Kim (2014), para compararse con la RNC que será generada con las técnicas de NE, esperando que las técnicas de NE en combinación con AP mejoren los resultados en cuanto al desempeño de clasificación del AS de tweets escritos en Español Mexicano.

Por un lado, las técnicas tradicionales, MSV, IB, y la RNC, tanto la creada a mano como la generada por el algoritmo de NE harán uso de *Word2Vec* y *BERT* para la representación del texto, en combinación con la etiqueta previamente dada a cada *tweet*. Los algoritmos mencionados recibirán los datos procesados con la técnica *Word2Vec* o *BERT*.

Se demuestra que, como resultado final de este trabajo de investigación, se obtiene un mejor desempeño de clasificación con la técnica de NE para RNC, frente a las técnicas tradicionales, MSV e IB, además, se consigue una menor complejidad de la estructura de la red neuronal artificial con respecto a la creada a mano.

Se demuestra que, como resultado final de este trabajo de investigación, se logra superar por medio de la técnica de NE y las estructuras generadas mediante ello, en la mayoría de los casos, a los

demás modelos de clasificación que se reportan en mayor medida en la literatura, MSV e IB. Además, se consigue una menor complejidad de la estructura de la red generada mediante NE, con respecto a la extraída de la literatura.

1.3. Hipótesis

Una RNC creada con técnicas de NE, presenta una mejora en el desempeño de clasificación, estadísticamente superior, con respecto a las técnicas tradicionales de AS, potencialmente, con menor complejidad que una RNC creada a mano y extraída de la revisión de la literatura, para el AS de tweets escritos en Español Mexicano.

1.4. Justificación y Motivación

Los criterios que justifican este estudio, se describen enseguida:

1. En el Estado del Arte, uno de los estudios hace uso de NEAT para hacer AS, el cual mejora los resultados.
2. En previos experimentos realizados por el autor, el uso de NEAT mostró resultados similares con respecto a las técnicas tradicionales del AS (Hernández-Hernández *et al.*, 2021).
3. En el Estado del Arte, las RNC, superan en la mayoría de los casos a las técnicas tradicionales para AS.

Además, se cree que el uso de las técnicas de NE sobre una RNC pueden mejorar los resultados ya obtenidos. Esto favorece la inserción de la NE en el AS, ya que no se han encontrado estudios en el que se lleve a cabo la aplicación de NE sobre RNC para tweets escritos en Español Mexicano.

1.5. Objetivos

1.5.1. Objetivo General

Experimentar con algoritmos de neuroevolución para redes neuronales convolucionales y comparar con clasificadores tradicionales, buscando mejorar el desempeño de clasificación de AS en tweets escritos en Español Mexicano.

1.5.2. Objetivos Específicos

1. Revisar y mejorar la representación de los tweets, para su posterior clasificación con una RNC creada con NE.
2. Calibrar los algoritmos y clasificadores para el etiquetado de tweets en polaridades sentimentales: positiva, negativa y neutral.
3. Clasificar tweets en polaridades: positiva, negativa y neutral por medio de una RNC creada a mano, técnicas tradicionales del AS y una RNC creada con NE, para su comparación.
4. Comparar el desempeño con respecto a la precisión de las técnicas tradicionales y de NE para un veredicto final, con respecto a los resultados obtenidos por una prueba estadística.

2. Marco Teórico

En este capítulo se describen de manera breve algunos conceptos y formulaciones relacionadas a la propuesta y a los trabajos previos de AS.

2.1. Análisis de Sentimientos en Twitter

El AS, también llamado Minería de Opiniones (MO), ''es el campo de estudio que analiza las opiniones, sentimientos, evaluaciones, actitudes y emociones en un cierto lenguaje, a través de productos, servicios, organizaciones, individuos, eventos, tópicos y sus características" (Liu, 2012, p. 1), expresados a través de las redes sociales digitales, como es el caso de Twitter.

Twitter es una red social de tipo *microblogging*, destinada a escribir *tweets* a través de la pregunta *¿Qué estás pensando?* sobre cualquier tema (Kwak *et al*., 2010). Un *tweet* es un mensaje, que puede ser interpretado como una *opinión*, referente a un tema en particular, compartido a través de la Internet y otros dispositivos que tengan acceso a sus aplicaciones (Murthy, 2013).

2.1.1. La opinión a través de un tweet

Un *tweet* es un texto escrito en máximo 280 caracteres, que expresa una idea, pensamiento u opinión acerca de un tópico de interés para quien lo emite. Una opinión, por su parte, contiene los siguientes elementos, según Liu (2012):

1. Una entidad, e, es un producto, servicio, tópico, publicación, persona, organización o un evento, que contiene un conjunto de características o atributos.
2. Un aspecto, a, que puede ser una característica observada o situación, de la entidad e.
3. Un sentimiento, s, generado por una emoción, debido al aspecto a observado de la entidad e, que se manifiesta con diferentes intensidades, y puede ser etiquetado en *positivo*, *negativo* o *neutral*.

4. El propietario de la opinión, h, quien ha hecho la opinión.
5. El tiempo, t, en el que el propietario h realiza la opinión.

Las anteriores variables involucradas en la opinión, se pueden ver gráficamente en la Figura 2.1.

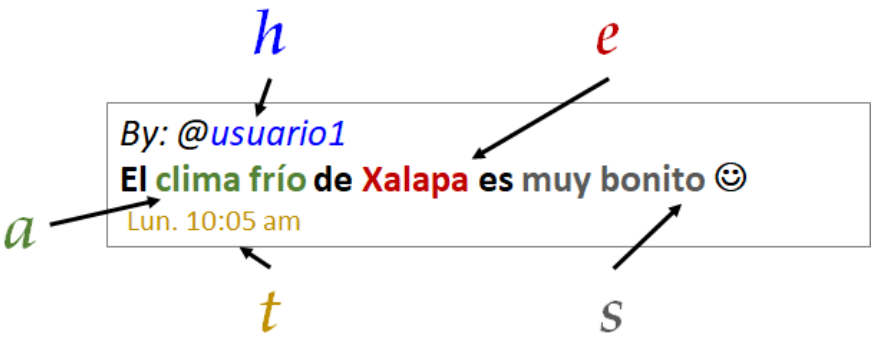


Figura 2.1: Ejemplo de una opinión con sus elementos

Aunque en las tareas del AS se encuentra que todos los elementos de la opinión pueden extraerse, en este trabajo nos enfocamos en el componente representado por s correspondiente al sentimiento de los tweets como opinión. Tomando en cuenta lo anterior, ya no es necesario hacer todas las etapas del pre-procesamiento de los datos, en particular la identificación de todos los elementos de la opinión, limitándonos a llevar a cabo la limpieza básica de los datos, y su posterior transformación, de manera que, los datos resultantes puedan ser procesados por algoritmos de AM o de AP. Esto tiene como resultado la simplificación de la tarea previa, agilizando el proceso en general.

2.2. Aprendizaje Automático para el Análisis de Sentimientos

El AM, es un campo perteneciente a la Inteligencia Artificial que estudia los algoritmos y técnicas para resolver de manera automática problemas complejos (Rebala *et al.*, 2019), a través de métodos estadísticos o matemáticos que simulan el aprendizaje.

Este trabajo se sitúa específicamente en el Aprendizaje Supervisado (ASu), el cual trabaja con un conjunto de datos que contienen

características y sus correspondientes clases. La idea parte de que, teniendo un vector de características X, una serie de funciones y decisiones lo transforme en un valor y, referente a la clase o etiqueta a la que pertenece dicho vector X. Dicha decisión o función se aprende de manera automática.

En AS, es común usar las técnicas de AM para realizar la tarea de descubrir el sentimiento predominante de los textos. Los algoritmos (q. e., clasificadores), más usados son la MSV y el IB.

2.2.1. Clasificador *Máquina de Soporte Vectorial*

Las MSV, introducidas en Cortes y Vapnik (1995), crean hiperplanos para poder hacer la separación de los datos en diferentes clases. MSV pueden manejar datos que no son *linealmente separables*, mapeando los puntos de un vector X en una dimensión más alta, usando funciones llamadas *kernels*.

El hiperplano creado es de $N - 1$ dimensiones para un vector X n-dimensional. La línea de decisión para clasificar o discriminar los datos, puede ser representada como en la Ecuación 2.1:

$$z = w * X + b \tag{2.1}$$

donde w y b, son el peso asignado a X y el término de *bias* o sesgo, respectivamente.

2.2.2. Clasificador *Ingenuo Bayesiano*

El clasificador IB, tal y como se explica en Bishop (2006), se basa en la Regla de Bayes, descrita en la Ecuación 2.2:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \tag{2.2}$$

donde $P(y|X)$ es la probabilidad de que X (vector de características) corresponda a la clase y, $P(X|y)$ es la probabilidad de, dada una clase y, exista una combinación específica de valores para cada atributo que conforma X (esto se conoce como la *verosimilitud*), y

$P(y)$, $P(X)$, denotan la probabilidad de ser y y de que exista X, respectivamente.

Por otra parte, y dado que algunas veces los datos no son discretos, la verosimilitud, i. e., $P(X|y)$, suele cambiar a la Ecuación 2.3:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \tag{2.3}$$

donde σ_y y μ_y, se calculan usando *máxima verosimilitud* (*Maximum Likelihood*), y $x_i \in X$, tal que $P(X|y)$, está definida en la Ecuación 2.4:

$$P(X|y) = \prod_{i=1}^{n} P(x_i \mid y) \tag{2.4}$$

Así, finalmente, se busca el mayor valor de $P(y|X)$, asignándose a los nuevos datos en X la clase con mayor probabilidad de ser y.

2.3. Aprendizaje Profundo en el Análisis de Sentimientos

El AP es una forma de ASu, que aplica sus técnicas haciendo uso de la jerarquía de conceptos, permitiendo a la computadora *aprender* conceptos complejos (Goodfellow *et al.*, 2016).

Las técnicas que se incluyen en el AP son: redes neuronales multicapa (RNM), LSTM, RNC, los *autoencoders* y *transformers*. Algunas de estas estructuras serán definidas en las subsecciones posteriores.

2.3.1. Redes Neuronales y la *retropropagación*

Las Redes Neuronales Artificiales (RNA), introducidas en Rumelhart *et al*. (1986), son un algoritmo computacional que trabaja tomando como inspiración las neuronas biológicas, pero que en vez de procesar señales, éstas procesan números. Son la base del AP.

El proceso básico de una neurona artificial se define en la Ecuación 2.5:

$$s = \sum_{i=1}^{N} x_i w_i + b \tag{2.5}$$

donde N es la cantidad de entradas a la neurona, w_i el peso correspondiente de la conexión de entrada, b el término de sesgo, s el valor de salida de la neurona, y x_i son las entradas de la neurona.

La suma y multiplicación pueden sustituirse por el *producto punto*, ya que se tienen el vector de entrada X y el vector de pesos W. Esto se expresa en la Ecuación 2.6.

$$s = XW + b \tag{2.6}$$

Además de que la neurona artificial ejecuta esta función, también computa una *no linealidad*, llamada *función de activación*. Algunas de éstas, son definidas en las Ecuaciones 2.7, 2.8, 2.9, 2.10.

- Sigmodal:

$$\frac{1}{1 + e^{-s}} \tag{2.7}$$

- Tangente Hiperbólica:

$$\frac{e^{s} - e^{-s}}{e^{s} + e^{-s}} \tag{2.8}$$

- Rectified Linear Unit (ReLU):

$$max(s, 0) \tag{2.9}$$

- Parametric ReLU (PReLU):

$$max(s, as) \tag{2.10}$$

- Softmax:

$$\frac{e^{s}}{\sum_{i=1}^{N} e^{s_i}} \tag{2.11}$$

Una función de activación se ejecuta después de la suma ponderada, permitiendo la no linealidad dentro de la neurona.

Varias neuronas pueden conectarse entre sí para crear una RNA. En la Figura 2.2, se muestra la estructura de una RNA.

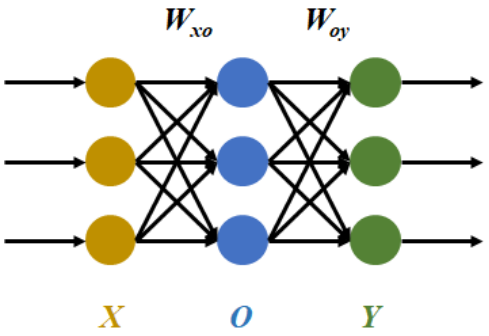


Figura 2.2: Estructura de una RNA, donde X es la capa de entrada, O representa la oculta, Y la de salida y W son los pesos que hay entre las capas

De la misma forma que en el producto punto, cuando se tiene una RNA de varias capas, se pueden utilizar matrices para el mejor manejo de los datos de entrada X, los pesos entre cada capa W y la salida Y, además se facilitan los cálculos.

Para que una RNA logre aprender, es necesario aplicar el algoritmo de *retropropagación*. Este algoritmo usa como salida esperada Y las etiquetas de los vectores en X, y en cada iteración calcula un error como el de la Ecuación 2.12:

$$E = \frac{1}{N}\sum_{i=1}^{N}(y_i' - y_i)^2 \tag{2.12}$$

siendo N la cantidad de neuronas de salida, y' la salida generada por la red y y la salida esperada. Esta función es llamada Error Cuadrático Medio (ECM).

De aquí en adelante, la RNA ahora tiene que calcular la derivada parcial de la Ecuación 2.13.

$$\frac{\partial E}{\partial W} \tag{2.13}$$

Dicha derivada parcial, para actualizar los pesos de la RNA, mediante el algoritmo del *Descenso del Gradiente*.

2.3.2. Redes Neuronales Convolucionales

Las RNC están basadas en la concepción de que el cerebro humano contiene neuronas que están acopladas en forma de capas, demostrado por el trabajo de Hubel y Wiesel (1959) sobre el córtex visual.

Las RNC están formadas de capas convolucionales, las cuales, a su vez, tienen un conjunto de filtros convolucionales (también llamados *kernels*), los cuales ejecutan la operación de convolución (S. Khan *et al.*, 2018).

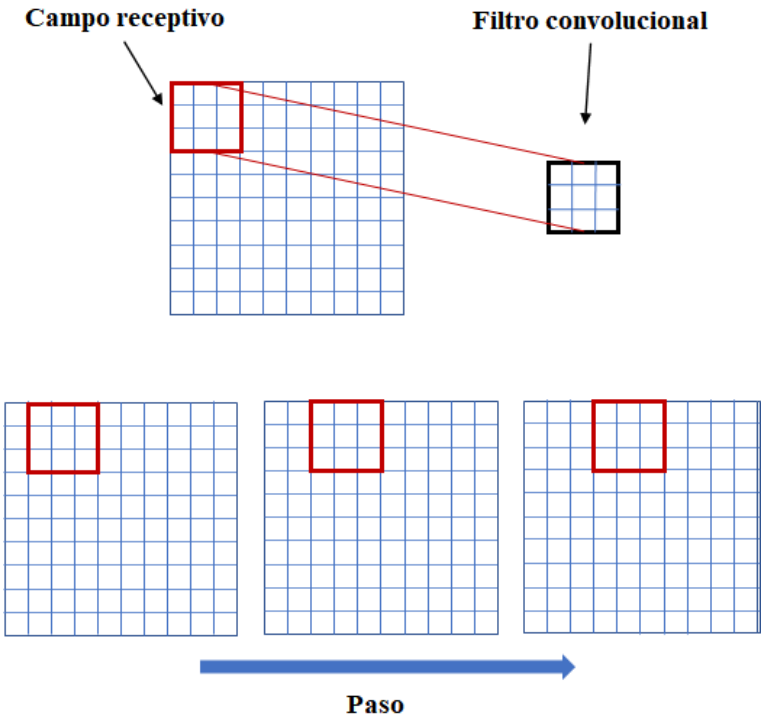


Figura 2.3: Partes y funcionamiento de la convolución

El filtro convolucional es una matriz de $n \times m$ llena de valores numéricos, que tiene un campo receptivo sobre otra matriz. El campo receptivo del filtro tiene que ver en qué parte de la matriz de entrada se va a realizar la operación de convolución, que es básicamente una suma ponderada e incluye un *sesgo*. Esta operación genera una *característica* y un conjunto de ellos es un *mapa de características*. Además, este filtro puede ir ''brincando" una cantidad de casillas hacia la derecha y hacia abajo, a este proceso se le llama *paso* (*stride*). En la Figura 2.3 se muestran las partes de una convolución.

Una RNC puede tener más de una matriz de entrada representando

el mismo objeto. Estas matrices son llamadas *canales*. Por el otro lado, la capa convolucional, puede dar como resultado más de una matriz de salida, estas matrices son llamadas *canales de salida*.

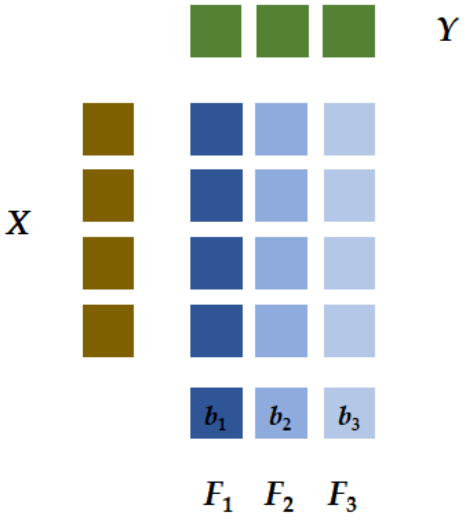


Figura 2.4: Capa convolucional con varios canales de entrada X y de salida Y

Tal y como se ve en la Figura 2.4, existen, ahora, varios filtros convolucionales (en azul), recibiendo varios canales de entrada X para generar varios canales de salida Y (llamados mapas de características), y sus respectivos valores de sesgo (denotados por b). Cada canal de salida F_i contiene tantos filtros convolucionales como la cantidad de canales de entrada, y su respectivo b_i. La operación a realizar para generar una característica se expresa en la Ecuación 2.14:

$$f = \sum_{i=1}^{N} F_{ci} * X_i + b \tag{2.14}$$

siendo $*$ la operación de convolución entre el filtro i del canal de salida c y la matriz X_i.

Luego de ejecutar la operación de convolución y tener los mapas de características listos, la RNC usa las funciones de activación. Posterior a esto, las operaciones de *pooling*, pueden ejecutar la función *max* para extraer el máximo valor o la función *avg* que es el promedio de los valores. El pooling también es un filtro y trabaja como el filtro convolucional, tiene un campo receptivo y un *paso*, pero diferente función.

Las RNC pueden tener múltiples capas convolucionales para mayor extracción de características. Al final de todas las capas convolucionales, usualmente se coloca una RNA para clasificar la entrada.

Como es costumbre, las RNC usan las llamadas funciones de pérdida, que calculan el error total de la RNC con respecto a una salida esperada, como la función de Pérdida de Probabilidad Logarítmica Negativa o Negative Log Likelihood Loss, entre otras. Normalmente las RNC son entrenadas mediante el algoritmo de *retropropagación*.

2.3.3. Modelo *Word2Vec*: *Continuous Bag-Of-Words* y *Skip-Gram*

El modelo *Word2Vec*, presentado en Mikolov, Chen *et al*. (2013) y en Mikolov, Sutskever *et al*. (2013), es un modelo de tipo autoencoder para transformar las palabras de un conjunto de datos de texto en vectores de valores continuos.

Este modelo incluye dos: *Continuous Bag-of-Words* (C-BOW) y *Skip-Gram* (SG). Ambos trabajan con el contexto de las palabras, que es meramente las palabras que están a cierta distancia de una en particular. Por ejemplo, si se tiene la frase: *La historia de esa película fue un desastre total*, se puede separar cada palabra por coma y así obtener: [la, historia, de, esa, película, fue, un, desastre, total]. El contexto funciona con un número definido, que para el ejemplo es 3. En la Tabla 2.1 se muestran las palabras y su correspondiente contexto.

Palabra	**Contexto**
la	historia
historia	la, de
de	historia, esa
esa	de, película
película	esa, fue
fue	película, un
un	fue, desastre
desastre	un, total
total	desastre

Tabla 2.1: Palabras y su correspondiente contexto en la frase de ejemplo

Estos modelos tienen una forma de *autoencoder* que reduce la dimensionalidad de la entrada y como salida esperada se tiene a la misma entrada (Baldi, 2012). En la Figura 2.5 se muestra la estructura de un autoencoder.

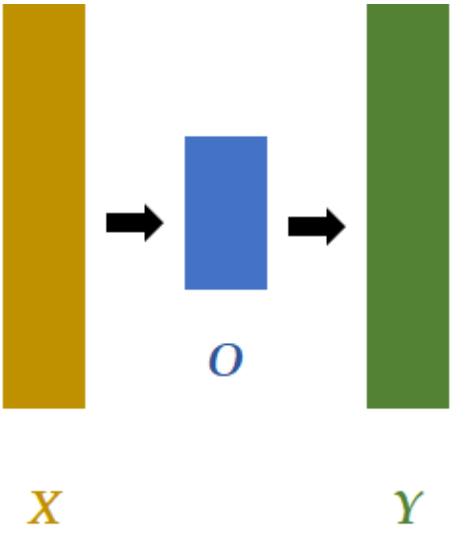


Figura 2.5: Estructura de un autoencoder

Este tipo de estructuras también son RNA, pero en el modelo *Word2Vec* funcionan de diferente manera y se usa el contexto de las palabras.

Las palabras, para que la red pueda leerlas, tienen que transformarse a *One-hot-vectors*. Uno de estos vectores tiene solo un 1 y el resto de posiciones tienen 0s, la cantidad de posiciones es igual a la cantidad de palabras en el *vocabulario* V que son todas las palabras en una base de datos. Cada vector, entonces, representa a una palabra.

Para que una palabra sea entrenada en el modelo C-BOW, del lado de la salida esperada Y se tiene que colocar dicha palabra, y del lado de la entrada X el contexto. En la capa oculta se encuentra la matriz de pesos que representa a las palabras después de terminar de entrenar.

2.3.4. Modelo *BERT*: *Bidirectional Encoder Representations from Transformers*

Al igual que *Word2Vec*, el modelo *BERT*, también se utiliza para transformar el texto en vectores de valores continuos. Dicho modelo fue presentado por Devlin *et al*. (2019), quienes a su vez, se basan en

Vaswani *et al*. (2017). *BERT* se describe, básicamente, como un modelo de lenguaje que pre-entrena, a través de texto no etiquetado, una representación bidireccional profunda que permiten las capas de neuronas de auto-atención (*self-attention*).

El entrenamiento de este modelo se basa en dos tareas no supervisadas: (1) predicción palabras de manera aleatoria, y (2) predicción de la siguiente sentencia en un texto dado. Cabe mencionar que este modelo originalmente fue entrenado para el Inglés, utilizando *WordPiece*, donde sus vectores no pertenecen a la palabra, sino, a la forma en cómo se dividen éstas (Wu *et al*., 2016).

Como resultado, este modelo arroja una matriz $e' \times m$ de valores continuos dado un texto con e' elementos, con respecto a *WordPiece*, siendo m un valor especificado por la cantidad de neuronas en cierta capa del modelo. Esta matriz, ya con la representación del texto, para el caso del AS, es la unidad principal que entrena los clasificadores.

2.4. Computación Evolutiva

La Computación Evolutiva (CE), tiene sus bases principalmente en el *Principio de la Evolución*, el cual se simula en computadoras para resolver una gama de problemas a través de la reproducción, variaciones aleatorias, competencias, y selección de individuos en una población de soluciones (Baeck *et al*., 2018), que están constantemente optimizándose.

Dentro de la CE existen varias técnicas las cuales son: algoritmos genéticos, estrategias evolutivas y programación evolutiva. En la Sub-sección 2.4.1, se describen los algoritmos genéticos.

2.4.1. Algoritmos Genéticos

Los Algoritmos Genéticos (AG), fueron introducidos en 1975 y descritos en Holland (1992). Las características de los AG son: (1) sujetos a una representación o codificación de las soluciones potenciales, se puede simular el espacio de búsqueda, (2) uso de una función de aptitud para evaluar el desempeño de las soluciones, (3) se usa selección para recolectar preferentemente a las mejores

soluciones, para aplicar algunos operadores de variación, (4) cruza y mutación para generar más soluciones, a las que se llaman $hijos$, y (5) la inclusión de las nuevas soluciones a la población original (Fogel, 2000). En la Figura 2.6, se muestra el procedimiento de un AG.

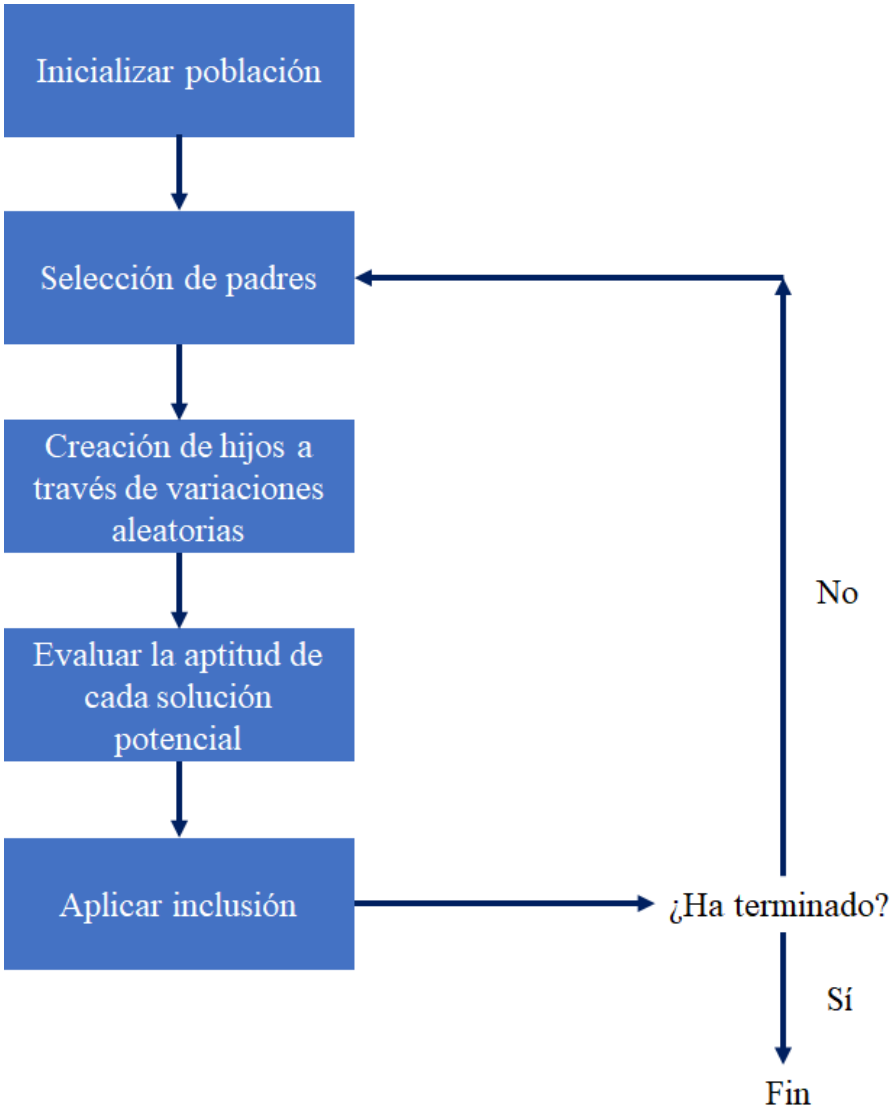


Figura 2.6: Esquema del funcionamiento de un AG (basado en Fogel (2000))

Los AG son utilizados, normalmente, para resolver problemas de optimización. Un problema de optimización se define como en Kochenderfer (2019), y que se muestra en la Ecuación 2.15:

$$\min f(\mathbf{x}) \tag{2.15}$$

donde $\mathbf{x}$ está sujeto a $\mathbf{x} \in X$, tal que X es el conjunto de soluciones

potenciales de ese problema, y $f(\mathbf{x})$ es la función que representa el problema a optimizar, que en este caso, consta de una minimización. Dicho problema también podría tratarse de una maximización.

Si $\mathbf{x} \in X$ entonces las soluciones tienen que estar representadas en alguna codificación. Dicha codificación representa el espacio de búsqueda. Por ejemplo, en el caso del *Problema del Vendedor Viajero*, una codificación puede ser a través de permutaciones, las cuales son listas o vectores de una dimensión especificada que contienen valores enteros únicos. Otras codificaciones utilizadas son la representación real, para espacios de búsqueda continuos, y la binaria, para espacios de búsqueda discretos. Todos estos valores estarán contenidos en un vector $\mathbf{x}$, el cual será evaluado a través de la función de aptitud $f(\mathbf{x})$.

Todas las soluciones pertenecientes a X deben ser inicializadas. Usualmente son creadas de manera aleatoria, y es por ello que se utilizan los operadores de variación, cruza y mutación, a través de los cuales se puede explotar y explorar, respectivamente, el espacio de búsqueda para generar nuevas soluciones. Una cruza de dos o varios individuos llamados *padres*, como en el ámbito natural, tiene la función de compartir *genes* para crear nuevos individuos a los que se les llaman *hijos*, estos últimos, considerados para mutarse usando una probabilidad.

Para que algunos individuos se crucen o recombinen entre sí, es necesario seleccionarlos mediante un mecanismo. Éste puede variar dependiendo del problema. Existe otro momento para escoger soluciones y es al seleccionar a los individuos que formarán parte de la siguiente generación.

Un AG, para aproximar una solución altamente competitiva itera una cantidad de generaciones, finalizando con la obtención de una solución $\mathbf{x}$ potencialmente cercana al óptimo.

2.4.2. Neuroevolución

La Neuroevolución (NE), está inspirada en el proceso biológico de la evolución en el cerebro humano. La NE utiliza las técnicas de CE para buscar las mejores estructuras, los pesos de las redes, algunos

hiperparámetros como la *tasa de aprendizaje* (usada para controlar el aprendizaje de la red), entre otros (K. O. Stanley, Clune *et al*., 2019). La NE, en algunos casos como en el algoritmo NEAT, sustituye a la regla principal de las RNA, el *Descenso del Gradiente*.

Los Algoritmos Evolutivos (AEs) más comunes para NE, son los AG. Y para este problema, los mismos mecanismos de selección, operadores de variación y la recombinación son utilizados para la búsqueda de RNA, con respecto a su estructura y algunas veces también con respecto a sus pesos. Sin embargo, un aspecto de particular interés con respecto a este problema, está relacionado con la codificación o representación de soluciones, las RNA (Vargas-Hákim *et al*., 2021).

En la mayoría de las veces, la función de aptitud se basa en la precisión que los individuos, las RNA, arrojan cuando son probadas. Otras funciones de aptitud agregan conocimiento sobre los elementos de las estructuras de RNA, ya que en algunos casos, como en Y. Sun *et al*. (2020), la optimización tiene que ver con reducir la cantidad de parámetros mientras se requiere de reducir el valor con respecto a una función de pérdida.

3. Marco Referencial

Existen diversas formas de hacer AS, siendo las más utilizadas: (1) de forma simbólica a través de diccionarios y palabras etiquetadas, y por otro lado, (2) a través de aprendizaje supervisado con técnicas tradicionales del AM, AP, CE y ensambles.

El AS parte, primero que nada, del uso de diccionarios o técnicas simbólicas para favorecer sus tareas. Una de estas técnicas es el uso de sinónimos de las palabras y los campos semánticos de las mismas que parte de un pre-etiquetado en diferentes emociones o sentimientos por parte de expertos. Éstas reciben, finalmente, un valor que las incluye en algún conjunto sentimental o emocional. En la Tabla 3.1 se describen de manera breve algunos trabajos de investigación que siguen estas técnicas.

Los diccionarios también llamados *lexicones*, son usados para etiquetar sentencias o documentos como críticas de películas y tweets. Además de esta forma de etiquetar, también se puede hacer, de manera bastante general, un etiquetado por parte de expertos a nivel de sentencia o documento. En la Tabla 3.2 se recopilan algunos de los estudios que hacen uso de técnicas tradicionales del AM para AS.

Continuando con el aprendizaje supervisado, en los últimos años se ha dado un incremento de las técnicas de AP para el AS, donde se remarca el uso de *Word2Vec*, *BERT* y las RNC. Se insiste en el uso de estas técnicas porque son la base principal de este trabajo, y se espera seleccionar alguna de las RNC para utilizarla sobre el conjunto de tweets escritos en Español Mexicano. Algunos de estos estudios se muestran en la Tabla 3.3 y 3.4.

En algunas ocasiones, se agregan ambas técnicas: la simbólica, utilizada de una manera más profunda, es decir, que se toman en cuenta con más importancia que los estudios de la Tabla 3.2, y la de aprendizaje. En este tipo de trabajos de investigación se agrega la combinación de técnicas simbólicas, AM y CE, por ejemplo, agregan el uso del diccionario *SentiWordNet* para comparar un ensamble basado en el clasificador MSV (F. H. Khan *et al.*, 2016), analizan

Autores	Tipo de Tarea	Contribución	Etiquetas y técnicas	Resultados generales
Esuli *et al*. (2006)	Simbólica	SENTIWORDNET	Uso de sinónimos	Diccionario en Inglés
Baccianella *et al*. (2010)	Simbólica	SENTIWORDNET 3.0	Uso de sinónimos	Creación del diccionario con una técnica diferente
Rangel *et al*. (2014)	Simbólica	Diccionario	Pre-etiquetado de palabras en diferentes niveles emocionales	Diccionario con palabras en Español con valores continuos
Rodríguez López y de Jesús Hoyos Rivera (2019)	Simbólica	Diccionario	Uso de campos semánticos	Diccionario con palabras del Español etiquetadas en valores del rango $[-1, 1]$

Tabla 3.1: Estudios en el ámbito simbólico

Autores	Tipo de Tarea	Contribución	Etiquetas y técnicas	Resultados generales
Neethu y Rajasree (2013)	Aprendizaje	AM sobre tweets de productos electrónicos	Etiquetas $+, -$. MSV, IB, ME, Ensamble	No hay diferencias entre clasificadores
Pang *et al*. (2002)	Aprendizaje	AM sobre críticas de películas	Etiquetas $+, -$. MSV, IB, ME	IB tiende a ser peor que MSV, pero no hay mucha diferencia
Read (2005)	Aprendizaje	Demostró que el dominio y el tiempo de los textos son importantes	Etiquetas $+, -$. MSV, IB	Los resultados varían entre clasificadores al cambiar el dominio de los textos
Pla y Hurtado (2013)	Aprendizaje	AM sobre tweets de diferentes tópicos, para el TASS 2013	Diversas polaridades. MSV	Mejores o similares frente a otros equipos del TASS

Tabla 3.2: Estudios en el ámbito del aprendizaje supervisado con técnicas tradicionales del AM; los símbolos $+$, $-$, son las polaridades sentimentales positiva y negativa

millones de tweets para crear sus propios diccionarios (Mozetič *et al*., 2016), y hacen uso del algoritmo evolutivo *Particle Swarm Optimization* para la extracción de características de los documentos o sentencias (Akhtar *et al*., 2017) para su posterior clasificación.

Por otra parte la CE también se ha utilizado para hacer clasificación con Programación Genética (PG) (Graff *et al*., 2020; Miranda-Jiménez *et al*., 2018), y el uso de NE, en un estudio donde se comparan redes neuronales *feed-forward*, IB, MSV, redes neuronales profundas de creencias y redes neuronales generadas por NEAT, siendo éste último mejor que el resto de clasificadores empleados para el AS de tweets en lengua Polaca (Sobkowicz, 2016). NEAT también fue

Autores	Tipo de Tarea	Contribución	Etiquetas y técnicas	Resultados generales
González *et al*. (2018)	Aprendizaje	AP sobre tweets, en el TASS 2018	Etiquetas $+$, $-$, $/$, ninguno. RNC, LSTM, (MSV + *Bag of Words*)	Arrojan resultados similares entre sí y mejores que otros equipos
Paredes-Valverde *et al*. (2017)	Aprendizaje	AP en tweets de servicios y productos en Español	Etiquetas $+$, $-$. (Word2Vec + RNC), IB, MSV	RNC mejor que el resto
Kim (2014)	Aprendizaje	RNC en diversos tipos de sentencias	RNC	RNC mejora con respecto a otros modelos del estado del arte
Ouyang *et al*. (2015)	Aprendizaje	Primera vez que se usa una RNC de 7 capas para críticas de películas	6 polaridades. (Word2Vec + RNC), IB, SVM	RNC mejor que el resto
Severyn y Moschitti (2015)	Aprendizaje	Proceso para iniciar parámetros de RNC	Etiquetas $+$, $-$. RNC	Mejores lugares en SemEval2015

Tabla 3.3: Estudios en el ámbito del aprendizaje profundo; los símbolos $+$, $-$ y $/$, son las polaridades sentimentales positiva, negativa y neutral

utilizado para hacer AS en tweets escritos en Español Mexicano, donde se utiliza una representación de tweets basada en la polaridad de la opinión y con respecto al campo semántico de las palabras; NEAT se comparó con una RNA, MSV y IB, donde se muestran resultados similares y mayores al 95 % de precisión (Hernández-Hernández *et al*., 2021).

Naturalmente, y debido a que el AP nació y fue usado para propósitos del área de Visión por Computadora como en la segmentación y clasificación de imágenes, técnicas de NE fueron creadas y utilizadas para resolver precisamente esos problemas. Así como los expertos en el área de PLN buscaron la mejora de sus estudios a través de las técnicas de AP como de las RNC y LSTM, en este trabajo se muestran las técnicas de NE que sirven de base para esta propuesta. Estos trabajos se pueden ver en la Tabla 3.5.

En esta revisión al Estado del Arte, se destaca el uso constante de técnicas tradicionales de AM como la MSV que, en varios de los estudios muestra resultados mejores en comparación con otros clasificadores como el IB y el EM, donde comúnmente, el uso de técnicas simbólicas predomina. Por otra parte, y debido a su uso recurrente en el área de Visión por Computadora, el uso de RNC en

Autores	Tipo de Tarea	Contribución	Etiquetas y técnicas	Resultados generales
Díaz-Galiano *et al*. (2019)	Revisión de literatura	AP sobre tweets, en el TASS 2019	Eso de variantes del Español y uso de *BERT* con RNR	Revisión sobre el concurso del TASS edición 2019
Godino y D'Haro (2019)	Aprendizaje	AP en tweets en diferentes variantes del Español	Etiquetas $+$, $-$, $/$, ninguno. *BERT* y otros clasificadores tradicionales	Mejor que otros trabajos en la literatura
Pastorini *et al*. (2019)	Aprendizaje	AP en tweets en diferentes variantes del Español	MSV, LSTM y RNA, y *BERT*	Las redes neuronales muestran buenos resultados
González *et al*. (2020)	Aprendizaje	Extensión del modelo del lenguaje *BERT* para el Español	Etiquetas $-$, $+$, $/$. *BERT* y RPP	Modelo de lenguaje TWilBERT extendido para el Español

Tabla 3.4: Estudios en el ámbito del aprendizaje profundo utilizando *BERT*; los símbolos $+$, $-$ y $/$, son las polaridades sentimentales positiva, negativa y neutral

Autores	Algoritmo	Codificación	Operadores usados	Tareas
K. O. Stanley y Miikkulainen (2002)	NEAT	Variable y directa	Selección, cruza y mutación	Aprendizaje por refuerzo y visión por computadora
Desell (2018)	Basado en NEAT	Variable y directa	Selección, cruza y mutación	Clasificación de imágenes
Xie y Yuille (2017)	Genético	Tamaño ajustado y binaria	Selección, cruza y mutación	Clasificación de imágenes
Y. Sun *et al*. (2020)	Genético	Variable y directa	Selección, cruza y mutación	Clasificación de imágenes
Dufourq y Bassett (2017)	Genético	Variable y directa	Selección y mutación	Clasificación de imágenes y AS en texto
Dahou *et al*. (2019)	Evolución Diferencial	Variable y directa	Operadores de Evolución Diferencial	AS en texto escrito en Árabe.

Tabla 3.5: Estudios en el ámbito de la Neuroevolución

el AS se nota más que el resto de técnicas de AP como las LSTM, que son generalmente más utilizadas para la predicción de, por ejemplo, la siguiente palabra en una sentencia. En este tipo de estudios, la representación de los datos se vuelve diferente, y de una manera no simbólica como los trabajos que se relacionan con las técnicas tradicionales del AM, en este caso, se suele agregar el uso de *Word2Vec* o *BERT* para la representación del texto. También se añade el uso de técnicas de CE, como la PG y NE, esta última la que más interesa y motiva en este estudio.

4. Propuesta

4.1. Metodología

La metodología incluye los objetivos específicos presentados con anterioridad. Aquí se describe cómo llevar a cabo esta serie de objetivos para lograr el fin. Dado que el objetivo principal de esta tesis es comprobar que la propuesta presentada tiene mejores resultados que las técnicas tradicionales, los siguientes son los pasos que se van a seguir:

1. Pre-procesamiento y transformación de los datos: como ya se tiene un conjunto de datos, éstos tienen que ser revisados, limpiados y transformados a representaciones numéricas, para que los clasificadores, tanto los tradicionales como los propuestos puedan operar sobre ellos.
2. Implementación y puesta a punto de una RNC: consiste en la implementación del algoritmo y la calibración de sus parámetros.
3. Puesta a punto de un algoritmo de NE para RNC: los algoritmos propuestos como NEAT y sus agregados, normalmente crean redes neuronales *feed-forward* más profundas, es posible que en ellos se puedan evolucionar las RNC. Sin embargo, en la revisión de la literatura, existen algunos algoritmos que han intentado hacer clasificación de sentimientos de texto con NE para RNC, como es el caso de EDEN (Dufourq & Bassett, 2017) y de DE-CNN (Dahou *et al.*, 2019).
4. Evaluación del desempeño y comparación de técnicas: los resultados de clasificación serán comparados con las pruebas estadísticas con respecto a la precisión que los algoritmos obtienen luego de entrenar.

4.2. Datos

Los tweets usados en este trabajo de investigación fueron recolectados y manualmente etiquetados en diferentes polaridades

sentimentales: *positiva*, *negativa* y *neutral*. Estos tweets forman parte del contexto político mexicano, y algunos de ellos fueron provistos por el Centro de Estudios de Análisis y Opinión de la Universidad Veracruzana (CEOA-UV). Los tweets fueron limpiados y posteriormente transformados, para que una RNC con el mecanismo de NE pueda leerlos. En la Tabla 4.1 se muestra la distribución original de los tweets en función de su polaridad.

Positivos	**Negativos**	**Neutrales**
3160	4579	17450

Tabla 4.1: Número de tweets por polaridad.

4.3. Limpieza de los datos

Con respecto a la limpieza de los tweets, se les extrajeron enlaces *https*, signos de puntuación, menciones en lo cual se incluye el símbolo '@', se mantuvieron las tendencias o *hashtags*, pero se elimina el símbolo '#', y todas las letras se cambiaron de mayúsculas a minúsculas. Los emojis también fueron recuperados.

Por otro lado, para evitar que la transformación de los datos a través del modelo *Word2Vec* tome un sesgo equivocado, valores numéricos que no pertenecían a ninguna palabra o tendencia, y tweets mayores a 60 palabras o con una sola, fueron eliminados.

4.4. Transformación de los datos

Para la transformación de los datos, se usó la biblioteca *Gensim*, de versión 3.8.1, implementada en Python con versión 3.7.4. Los parámetros utilizados en esta transformación, pertenecen al modelo *C-BOW*, y solo algunos fueron modificados para efectos de la tarea en cuestión. La dimensionalidad de los vectores de salida fue de 60; las palabras, tendencias o emojis que solo tuvieran *una* aparición en todos los datos, fueron descartados; se usó la semilla aleatoria 0, y el resto de parámetros se mantuvieron sin modificaciones. Los elementos correspondientes a un tweet fueron unidos en una matriz de $n \times m$, donde $m = 60$ y n el número de elementos máximo en un tweet.

En el caso del modelo *BERT*, se utilizó una red neuronal previamente entrenada con un tamaño reducido, propuesta en Abdaoui *et al.* (2020) y que incluye 15 idiomas, entre los cuales se encuentra el Español. Este modelo genera una matriz de $n \times m$, por lo tanto, no fue necesario hacer una unión de los vectores por cada palabra o elemento. Los valores resultantes corresponden a la última capa del modelo, donde $m = 768$ y n es el número de elementos. Dicho modelo fue entrenado con la biblioteca *PyTorch* versión 1.9.

En ambos casos, las matrices resultantes fueron modificadas centrando los vectores y agregando 0s como relleno o *padding*, haciendo una matriz de dimensiones más grandes con respecto a n para cada representación. Estos valores corresponden con el valor máximo de elementos en un tweet permitidos; en *Word2Vec* es 60, por la cantidad de palabras o elementos, y en *BERT* es 134 ya que este modelo toma en cuenta otros elementos más allá de las palabras y emojis. Suponiendo un tweet con 4 elementos, e', la matriz resultante, después de su modificación, se vería como la Figura 4.1.

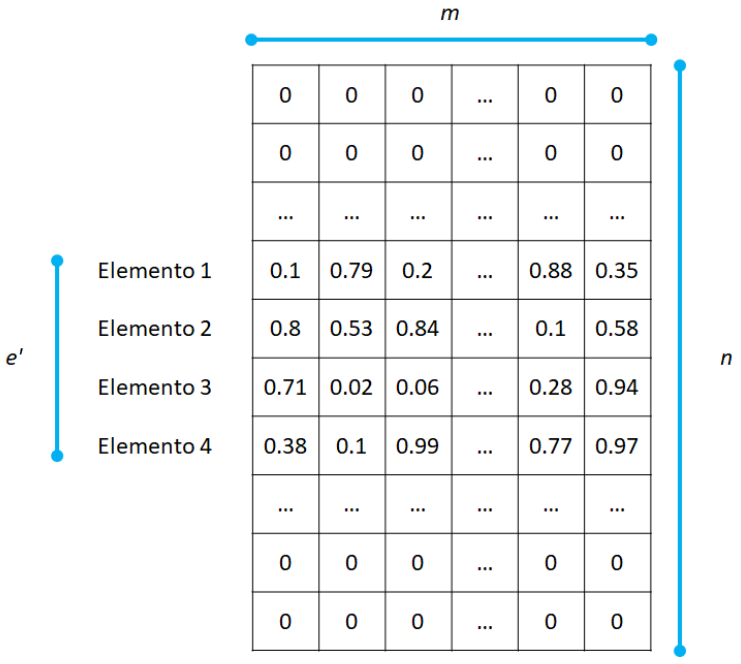


Figura 4.1: Estructura de un tweet después de su transformación a través de *Word2Vec* o *BERT*

Así, las matrices de *WordVec* tienen las dimensiones 60×60, y las de *BERT*, 134×768.

4.5. Red Neuronal Convolucional para el Análisis de Sentimientos

Dada la complejidad de la tarea relacionada con el análisis del lenguaje natural para la determinación de la polaridad, uno de los recursos que se utilizan son las RNC, pues permiten el análisis de estructuras complejas, como lo es la lengua, y con base en esto, logra llevar a cabo eficientemente procesos de clasificación de diferentes órdenes de complejidad.

La RNC tomada de la revisión de la literatura, pertenece a la creada por Kim (2014) y que ha sido usada en otros casos como en Severyn y Moschitti (2015) y Paredes-Valverde *et al*. (2017). La estructura de esta RNC se muestra en la Figura 4.2.

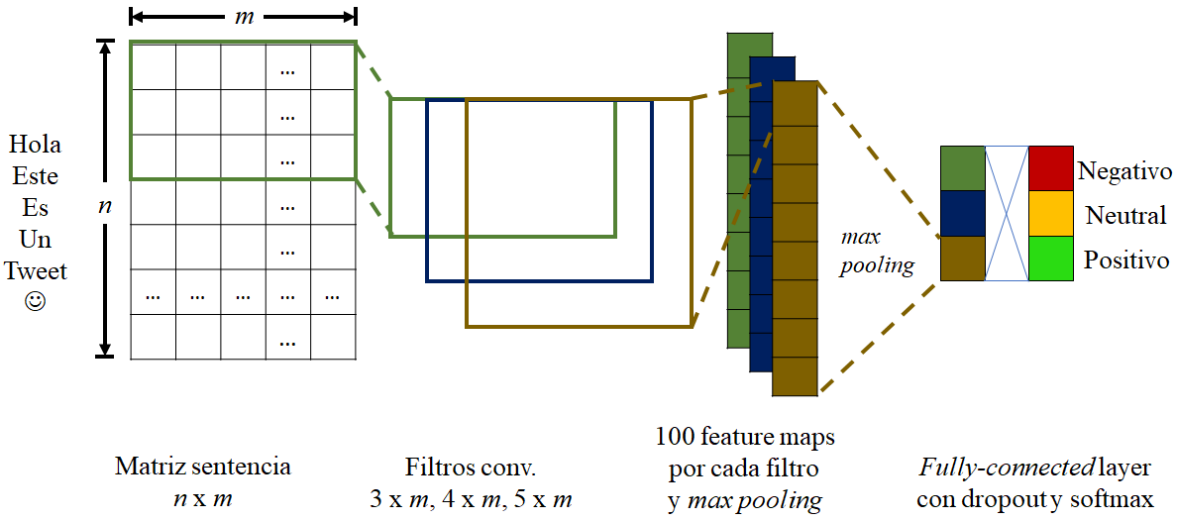


Figura 4.2: Estructura de la RNC utilizada para clasificación de texto

En la estructura de esta RNC, se puede apreciar, en primera instancia, la matriz sentencia que contiene los elementos del tweet a clasificar. Cada una de las filas corresponde a un elemento, representada en un vector de valores continuos de tamaño m (ver Sección 4.4). Un tweet, además, puede tener n elementos como máximo.

La siguiente parte de la estructura, corresponde con los filtros convolucionales. La RNC usa tres de diferentes tamaños (acoplados al tamaño de la entrada con respecto a las columnas): $3 \times m$, $4 \times m$, $5 \times m$, para extraer la mayor información posible del único canal de entrada.

Además, cada uno de estos filtros genera 100 *mapas de características*, los cuales, a su vez, pasan a través de una función de activación, que por defecto es la TanH, y una operación de *max pooling*, seleccionando el mayor valor de cada uno de los mapas de características generados. En la Figura 4.3 se aprecia un acercamiento a la parte correspondiente de la estructura de la RNC, tomando como ejemplo el filtro convolucional de $5 \times m$, representado con el color amarillo oscuro.

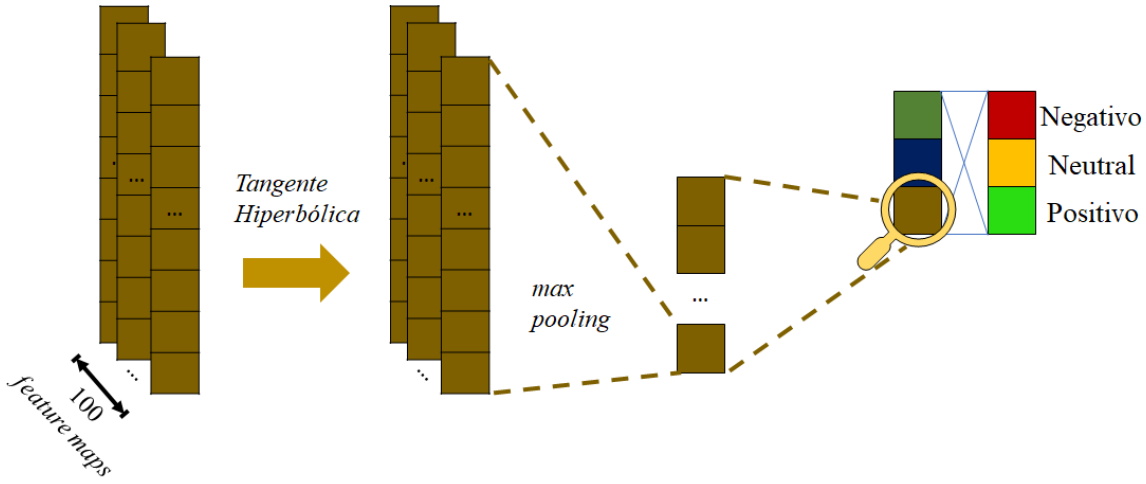


Figura 4.3: Estructura y flujo de la información a través del filtro convolucional de $5 \times m$

Al terminar el cómputo del *max pooling* se encuentra la capa *fully-connected*, capa que usa 300 neuronas de entrada, y 3 de salida, correspondientes a las 3 clases a clasificar. Esta capa tiene como función de activación a *softmax* y el mecanismo de *dropout*.

4.6. Algoritmo Genético para la Evolución de una Red Neuronal Convolucional para el Análisis de Sentimientos

En esta sección se presenta el mecanismo de NE para RNC, un algoritmo genético, que es una creación propia, fue llamado *Deep NeuroEvolution of Weights and Topologies* (DeepNEWT). Como idea fundamental de este algoritmo, se tiene que con mecanismos suficientes pueda buscar una estructura de RNC para clasificación de texto, mientras que se los pesos también están siendo entrenados, sin

Algoritmo 1: DeepNEWT

Datos: Límites de parámetros continuos y discretos
Resultado: Mejor RNC encontrada
Iniciar N RNC en la población P;
Inicializar ϕ_a y ϕ_b;
Calcular la aptitud de cada RNC;
mientras *máximo de generaciones no se ha alcanzado* **hacer**
  $P_s \leftarrow$ seleccionar elementos de P por Torneo;
  $P_c \leftarrow$ cruzar los elementos de P_s ;
  $P_a \leftarrow$ mutar los individuos de P_c con probabilidad ϕ_a ;
  $P_b \leftarrow$ mutar los individuos de P_a con probabilidad ϕ_b ;
  $P^{t+1} \leftarrow$ mejores de $P^t \cup P_c \cup P_a \cup P_b$ pasan a la siguiente generación;
fin

necesidad de entrenar a la red mediante *retropropagación* por cada generación.

Varios elementos fueron introducidos: (1) usar codificación directa de los individuos, (2) permitir cruzar y mutar individuos, (3) admitir el cambio de funciones no lineales y funciones de pooling, (4) buscar similitudes entre los individuos, (5) entrenar pesos, (6) permitir el usar la estructura de RNC de la literatura (ver Sección 4.5).

DeepNEWT se basa en ideas de otros trabajos. La codificación directa se basa en Desell (2018), Dufourq y Bassett (2017), K. O. Stanley y Miikkulainen (2002), Y. Sun *et al*. (2020), donde los bloques convolucionales, como los filtros de pooling son vistos por los algoritmos para sus propósitos. Además, permite la creación y combinación de las RNC creadas en Kim (2014) y Ouyang *et al*. (2015). A esta codificación, también se le agrega la capa *fully-connected* de perceptrones totalmente interconectados. *DeepNEWT* permite la cruza y mutación de los individuos como en Xie y Yuille (2017), pero en vez de utilizar codificación binaria, para este caso se utiliza una codificación directa como en Y. Sun *et al*. (2020).

Con respecto a la cruza y a la búsqueda de similitudes entre individuos, *DeepNEWT* está basado en la concepción de compartir elementos con la misma similitud como se hace en K. O. Stanley y Miikkulainen (2002) y Desell (2018), pero, en vez de hacerlo sobre los perceptrones (nodos) o mapas de características y filtros, se hace sobre bloques completos de filtros convolucionales, no linealidades y

filtros de pooling. Por su parte, la mutación de pesos y sesgo, es una simple adición de valores aleatorios como se propone en K. O. Stanley y Miikkulainen (2002). En el Algoritmo 1 se presenta el proceso completo de *DeepNEWT*.

En las siguientes sub-secciones se describen de manera detallada, la representación y operadores de variación utilizados por *DeepNEWT*.

4.6.1. Representación

La base de un buen desempeño del algoritmo es una buena representación o codificación adecuada. Esta codificación resguarda las estructuras de RNC en bloques *convolucionales + no linealidad + pooling*, llamados bloques *CNP* o *last-CNP* (que se refiere a la RNC de la Sección 4.5) y la capa *fully-connected* llamada *FC*. En la Figura 4.4 se muestra el flujo de la información desde la entrada hasta la salida a través de las RNC.

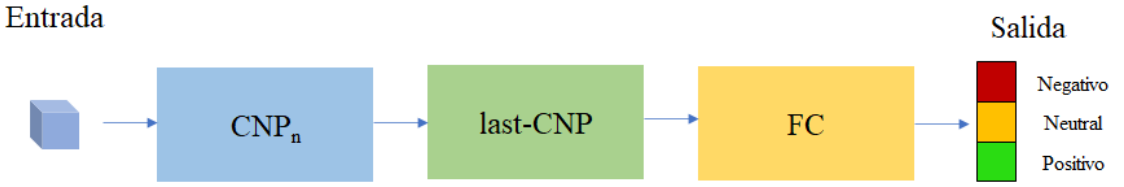


Figura 4.4: Estructura predeterminada de una RNC dentro de DeepNEWT

4.6.1.1. Capa de bloques *CNP*

La primera parte de estas estructuras contienen varios bloques CNP, los cuales pueden llegar a desaparecer incluso en su totalidad si, ante la ejecución de los operadores genéticos, se dan las condiciones para que esto suceda. La Figura 4.5 muestra el flujo de la información a través de estos bloques.

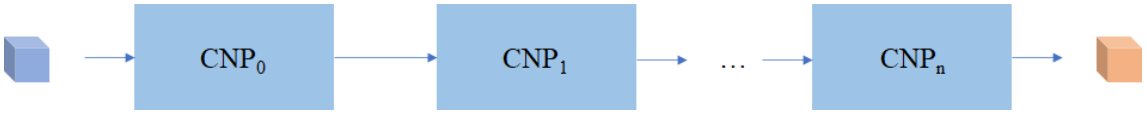


Figura 4.5: Estructura de una capa de bloques CNP

Cada CNP, por su parte, tiene $z \times z'$ filtros convolucionales, donde z es el número de canales de entrada y z' el número de mapas de características de salida, cada uno de tamaño $v \times w$, una función no

lineal y una operación de pooling que conlleva un filtro de $s \times t$ dimensiones. Asimismo, como en la descripción de las RNC (Sección 2.3.2), cada filtro, ya sea convolucional o de pooling, tienen un paso o *stride* que mantener; para el algoritmo, el paso predeterminado es de 1.

El cómputo de los filtros convolucionales en un tensor de $v \times w \times z \times z'$ genera una salida de $n' \times m' \times z'$, dada una entrada de $n \times m \times z$. Siendo que n y n', son las filas, m y m', las columnas de una matriz de entrada y otra de salida, respectivamente (ver *Matriz sentencia* en 4.2). Tomando en cuenta las dimensiones de los tensores y el paso predeterminados, en la Ecuación 4.1 se refleja el cómputo a seguir de n':

$$n' = n - v + 1 \tag{4.1}$$

y la Ecuación 4.2 calcula a m':

$$m' = m - w + 1 \tag{4.2}$$

En la Figura 4.6 se muestra gráficamente cómo se vería el funcionamiento de los filtros convolucionales.

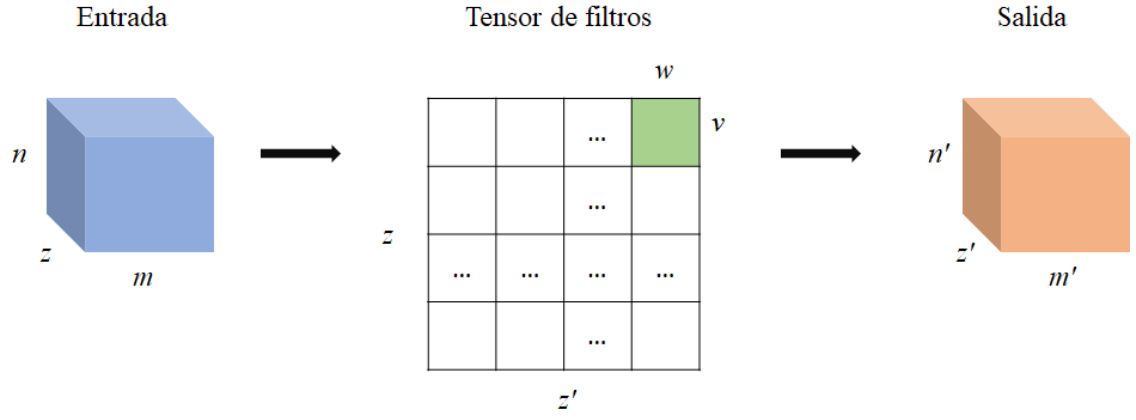


Figura 4.6: Cambio de la dimensionalidad después de la operación de convolución

Posterior al cálculo de la operación de convolución, una no linealidad calcula una siguiente salida. En este caso, las llamadas funciones de activación Sigm, TanH, ReLU y PReLU, son las utilizadas de manera predeterminada por el algoritmo en los bloques CNP. Este cálculo no

afecta las dimensiones de la salida de $n' \times m' \times z'$.

Después, la salida obtenida se ve modificada por un filtro de pooling de $s \times t$, que realiza el mismo camino que un filtro convolucional, igualmente con un paso predeterminado de 1, pero ejecutando la operación de pooling: máximo (Max) o promedio (Avg). Cuando una operación de pooling sobre una entrada de $n' \times m' \times z'$ se calcula, la salida generada cambia sus dimensiones a $n'' \times m'' \times z'$. Ahora la matriz resultante, tiene n'' filas y m'', columnas. En la Ecuación 4.3 se calcula n'':

$$n'' = n' - s + 1 \tag{4.3}$$

y en la Ecuación 4.4, m'':

$$m'' = m' - t + 1 \tag{4.4}$$

En la Figura 4.7 se ve de manera gráfica el cambio en dimensiones de la salida con respecto a la entrada generada por el filtro de pooling.

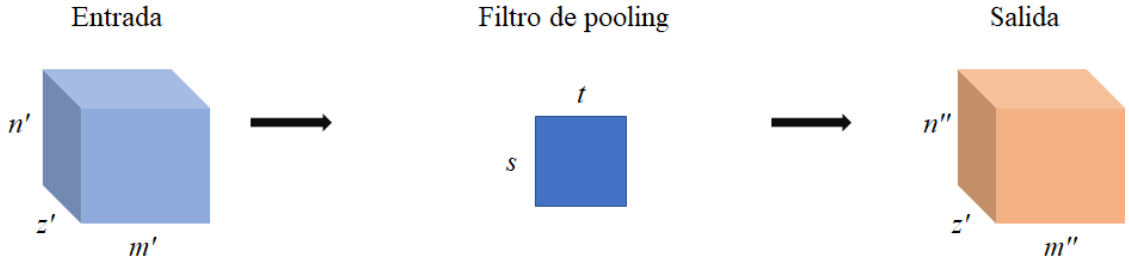


Figura 4.7: Cambio de la dimensionalidad después de la operación de pooling

Cada bloque CNP, además de resguardar información acerca de los filtros tanto convolucionales como de pooling, también guarda información acerca del tipo de capa uniendo el pooling y la no linealidad.

4.6.1.2. Similitud de RNCs

Uno de los nuevos elementos que tiene este algoritmo, es la facilidad de recombinar los individuos a través de la similitud de la topología. Basado en las *marcas históricas* de K. O. Stanley y Miikkulainen (2002) y el operador de cruza de EvoCNN en Y. Sun *et al*. (2020).

DeepNEWT tiene por objetivo encontrar cuál CNP comparada con otra tiene no linealidades y operaciones de pooling similares. En la Tabla 4.2, se muestran los números de similitud entre cada no linealidad y operación de pooling.

Número	No linealidad	Pooling
1	Sigm	Max
2	Sigm	Avg
3	Tanh	Max
4	Tanh	Avg
5	ReLU	Max
6	ReLU	Avg
7	PReLU	Max
8	PReLU	Avg

Tabla 4.2: Números de similitud y sus correspondiente no linealidad y operación de pooling

Cada CNP guarda un número de similitud.

4.6.1.3. Bloque *last-CNP*

El bloque last-CNP se refiere a la estructura de RNC utilizada de manera predeterminada (ver Sección 4.5). Es el único bloque que no va a desaparecer (junto con la capa FC), pero puede ser modificado con los operadores de mutación. Además, puede almacenar varios filtros convolucionales con diferentes tamaños en v y w. Cabe mencionar que cada CNP solo tiene un tamaño de filtro convolucional a diferencia de los bloques last-CNP.

Como ya se mencionó, el bloque last-CNP contiene uno o más filtros convolucionales de tamaño $v \times w$, de esta manera el bloque genera más de una salida, cada una con respecto a un filtro convolucional. En la Figura 4.8 se observa el flujo de la información.

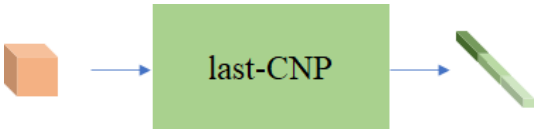


Figura 4.8: Flujo de la información dentro del bloque last-CNP

De esta manera, se tiene que el conjunto $F = \{f_i | f_i$ es un tensor de $z \times z'$ con filtros convolucionales de $v_i \times w$ e $i = 1...N\}$, siendo N el número de filtros diferentes. Por lo tanto, la salida, después de la operación convolucional, se define como el conjunto $Y = \{y_i | y_i$ es un

tensor de tamaño $n' \times 1 \times z\}$, donde n' se define en la Ecuación 4.1. El filtro convolucional i tiene una dimensión de $v_i \times m$, donde $w = m$, con respecto a una entrada $n \times m \times z$. Visualmente, en la Figura 4.9 se muestra el cambio de la dimensionalidad de la salida con respecto a la operación de convolución realizada por el filtro i.

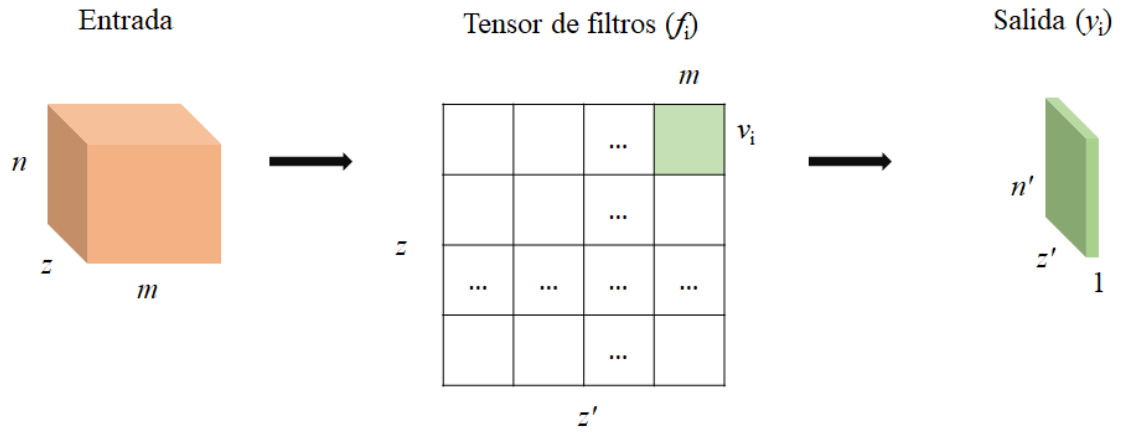


Figura 4.9: Cambio de la dimensionalidad después de la operación de convolución en el bloque last-CNP

De igual manera como en la RNC predeterminada y en los bloques CNP, en el bloque last-CNP se ejecuta una no linealidad. Las mismas funciones de activación de los bloques CNP se utilizan dentro del bloque last-CNP.

Los filtros de pooling tienen las mismas características que en la RNC predeterminada, i. e., que después de la operación convolucional y el cómputo de la no linealidad, el filtro tendrá la misma cantidad de filas, s, y columnas, t, que la salida con dimensiones n' y m, respectivamente. Por lo tanto, una operación de pooling sobre una entrada de $n' \times 1 \times z'$, dado que $m = 1$, genera una salida de $1 \times 1 \times z'$, es decir, que en cada z'_i hay un escalar. En la Figura 4.10 se muestra el cambio de dimensionalidad con respecto a la operación de pooling en el bloque last-CNP.

Posterior al cálculo del bloque last-CNP, las diferentes salidas y_i, ya con respecto a la operación de pooling, se concatenan una tras otra para ser enviadas a la última capa de esta RNC generadas por DeepNEWT. De manera que, el vector resultante del bloque last-CNP tendrá una dimensión de $|Y|z'$ misma que el número de neuronas de entrada a la capa FC.

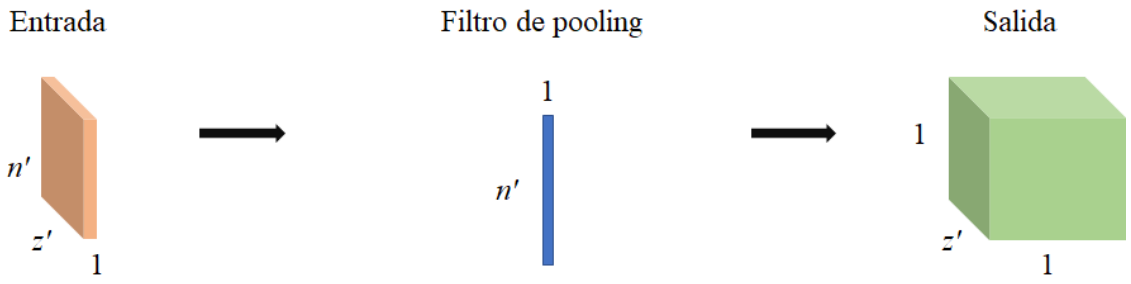


Figura 4.10: Cambio de la dimensionalidad después de la operación de pooling en el bloque last-CNP

4.6.1.4. Capa *FC*

Finalmente, la capa FC. Incluye el clasificador neuronal con una sola capa de entrada y salida. Las neuronas se encuentran interconectadas entre sí. El número de neuronas de entrada varía entre cada RNC dentro de DeepNEWT ya que tiene que ver con la salida del bloque last-CNP, como ya se mencionó en la sección anterior. El número de neuronas en la capa de salida es la misma que el número de clases a etiquetar, es decir, 3, una neurona por polaridad sentimental, positiva, negativa o neutral. En la Figura 4.11 se muestra el flujo de la información a través de la capa FC.

Figura 4.11: Flujo de la información dentro de la capa FC

Las neuronas calculan el proceso básico tal y como se describe en la Sección 2.3.1. Luego de ello, una función de activación (no linealidad) calcula una salida. Para esto, las funciones disponibles son: Sigm, Tanh y Softmax.

4.6.1.5. Objetivo de la representación

El objetivo principal de esta codificación es poder mantener los bloques CNP, last-CNP y FC de forma explícita, es decir, que toda información referente a los pesos de los filtros y las conexiones de neuronas en la capa FC, las funciones de activación (no linealidades), pueden ser vistas, en algún momento por el algoritmo, para ejecutar alguna función de mutación o cruza.

Esta codificación también permite el entrenamiento de los pesos de la red durante la búsqueda de la estructura, situación que no sucede en otras representaciones incluyendo la binaria, donde las RNC se entrenan mediante el Descenso del Gradiente.

4.6.2. Operador de selección y cruza

El procedimiento del operador de selección se hace a través de un *Torneo Determinista*, en donde, de la población original P se seleccionan p individuos sin reemplazo, de los cuales, solo uno pasa a formar parte del conjunto de los padres a cruzar. Este procedimiento se lleva a cabo tantas veces sean necesarias para elegir un número de padres T. Y así, como van siendo seleccionados los pares de padres, así serán sometidos al operador de cruza. Todos los padres serán cruzados y si no son cruzados, por la naturaleza de los operadores, pasarán a la siguiente etapa como hijos. Este operador genera el conjunto P_c de individuos cruzados.

4.6.2.1. Operador de cruza del bloque *CNP*

Se tienen dos padres $P1$ y $P2$, que son los que se van a cruzar en las capas CNP. Como resultado del operador, se generan dos hijos $H1$ y $H2$.

El funcionamiento de este operador se muestra en la Figura 4.12. Cada bloque CNP se divide por la línea roja, cada no linealidad **N** tiene su propio índice, al igual que el pooling **Po** (filtro y operación). A la derecha de cada **N** y **Po** está el número de similitud.

Para realizar la cruza, primero se identifican los números de similitud que existen en ambos padres, si hubiera más de una ocurrencia en un padre, se seleccionan de manera aleatoria. Solo se van a cruzar los filtros de pooling y su respectiva operación, manteniendo así, los números de similitud sin modificarse. En caso de no haber mismos números de similitud en ambos, los mismos padres pasan directamente a la mutación.

4.6.2.2. Operadores de cruza del bloque *last-CNP* y de la capa *FC*

Estos operadores parten con dos padres $P1$ y $P2$, y resultan en dos hijos $H1$ y $H2$.

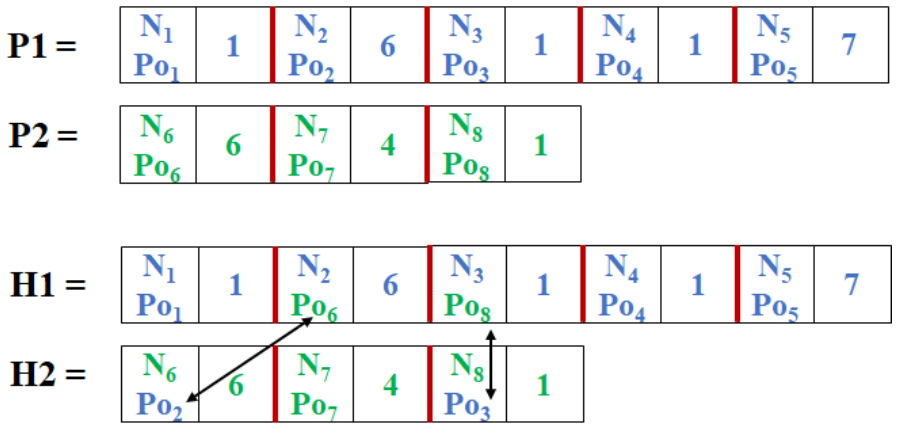


Figura 4.12: Cruza de elementos en el bloque CNP

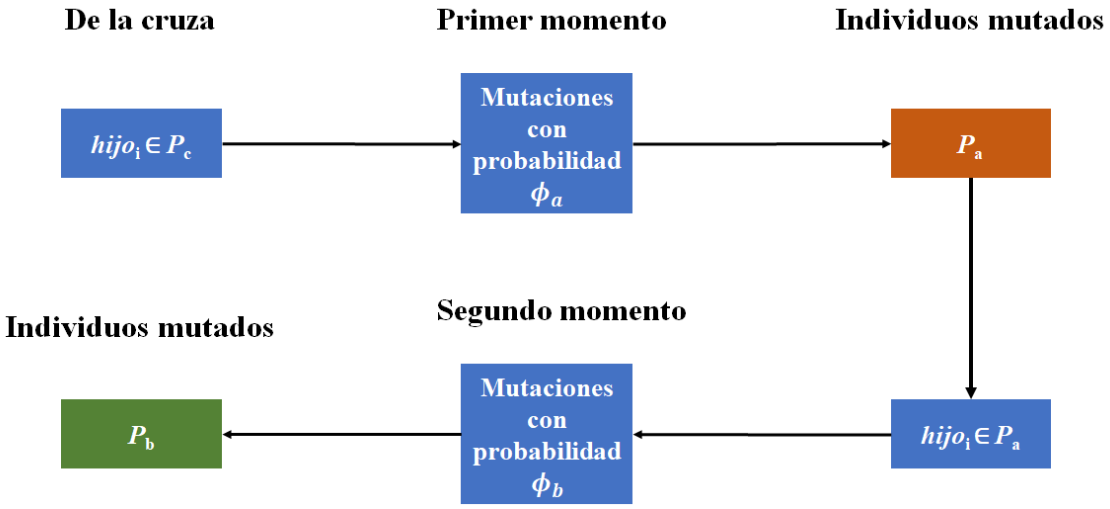


Figura 4.13: Esquema de momentos en el operador de mutación

En el caso del bloque last-CNP, la operación de pooling es la que va a ser compartida, sin el filtro.

Y en el caso de la capa FC, la no linealidad o función de activación de la capa de salida es la compartida.

4.6.3. Operador de mutación

En el operador de mutación para los bloques CNP, last-CNP y FC, se involucran dos momentos. El primero de ellos con una probabilidad ϕ_a y el segundo con otra probabilidad ϕ_b, sujetos a $0 < \phi_b < \phi_a < 1$, siendo que, entonces, no siempre los individuos podrán mutarse en ambos momentos, solo en el primer momento o en ninguno de ellos.

El operador de mutación genera dos conjuntos de individuos mutados, tanto para el primer momento, P_a, como para el segundo,

P_b. Cada uno de estos momentos contienen una secuencia de varias mutaciones. En la Figura 4.13 se muestra el esquema con cada momento y sus correspondientes probabilidades de mutación.

4.6.3.1. Secuencias de mutaciones

Para crear una solución después de una mutación, se siguen las siguientes secuencias de mutaciones de manera que la más disruptiva suceda primero, es decir, la que más modifique al individuo y genere un procedimiento de recomposición. Para el primer momento de mutaciones:

1. Número de filtros convolucionales
2. Número de canales de salida (o mapas de características)
3. Función no lineal
4. Tamaño de los filtros convolucionales
5. Modificación de los pesos W y b de los filtros convolucionales

El Algoritmo 2 muestra el procedimiento a seguir dentro del primer momento de mutaciones. En este algoritmo se involucran los hijos resultantes de la cruza en el conjunto P_c. Cada hijo puede ser mutado con probabilidad ϕ_a por cada mutación i dentro del conjunto de mutaciones M_a del primer momento.

El segundo momento, por otro lado, tiene la siguiente secuencia:

1. Número de filtros convolucionales
2. Número de canales de salida (o mapas de características)
3. Número de bloques CNP
4. Operación de pooling
5. Tamaño del filtro de pooling
6. Función no lineal
7. Tamaño de los filtros convolucionales
8. Modificación de los pesos W y b de los filtros convolucionales

Algoritmo 2: Primer momento de mutaciones

Datos: Hijos resultantes de la cruza P_c
Resultado: Hijos modificados P_a
$P_a \leftarrow \emptyset$;
para $hijo \in P_c$ **hacer**
  $i \leftarrow 1$;
  mientras $i \leq |M_a|$ **hacer**
    si $r < \phi_a$ **entonces**
      $hijo \leftarrow \text{mutaciónA}_i(hijo)$;
    fin
    $i \leftarrow i + 1$;
  fin
  si *hijo fue modificado* **entonces**
    $P_a \leftarrow P_a \cup hijo$;
  fin
fin

Algoritmo 3: Segundo momento de mutaciones

Datos: Hijos resultantes de la mutación P_a
Resultado: Hijos modificados P_b
$P_b \leftarrow \emptyset$;
para $hijo \in P_a$ **hacer**
  $i \leftarrow 1$;
  mientras $i \leq |M_b|$ **hacer**
    si $r < \phi_b$ **entonces**
      $hijo \leftarrow \text{mutaciónB}_i(hijo)$;
    fin
    $i \leftarrow i + 1$;
  fin
  si *hijo fue modificado* **entonces**
    $P_b \leftarrow P_b \cup hijo$;
  fin
fin

A diferencia del primer momento, el segundo toma en cuenta los hijos modificados del primer momento en el conjunto P_a, la probabilidad ϕ_b y las mutaciones correspondientes al segundo momento M_b. El Algoritmo 3 evidencia la secuencia de mutaciones de este momento.

4.6.3.2. Operador de mutación de los bloques *CNP* y *last-CNP*

En la Tabla 4.3, se muestran las mutaciones que el algoritmo puede hacer, con sus respectivos bloques, tipos y movimientos.

Las mutaciones sobre la capa CNP, siguen este procedimiento: (1) seleccionar un bloque, (2) del bloque seleccionado, tomar los tensores

Mutación	CNP	last-CNP	Tipo
W y b de los filtros conv.	Sí	Sí	Suma de valores aleatorios
Tamaño de los filtros conv.	Filas y Columnas	Solo Filas	$+/-$ en uno
Función no lineal	Sí	Sí	Aleatorio
Tamaño del filtro de pooling	2do. mov.	-	$+/-$ en uno
Operación de pooling	2do. mov.	2do. mov.	Cambio
Número de bloques CNP	2do. mov.	-	$+/-$ en uno
Número de canales de salida	Sí	Sí	$+/-$ de forma aleatoria
Número de filtros conv.	-	Sí	$+/-$ en uno

Tabla 4.3: Esquema de mutaciones a ejecutar en los movimientos y en cada bloque, se agrega el tipo de mutación; el símbolo $+/-$ significa incrementar o reducir

de pesos W y sesgo b de los filtros convolucionales, (3) modificar el tensor dependiendo de la mutación. Las mutaciones de W y b suceden secuencialmente, es decir, ya que se modifica W con respecto a sus dimensiones, por ejemplo, necesariamente tendrá que ser modificado el tensor b.

Con respecto al bloque last-CNP, el procedimiento a seguir es: (1) seleccionar un tensor de filtros del conjunto F, (2) del tensor f_i seleccionado aplicar la mutación sobre W y b. Igualmente, las mutaciones de W y b suceden de manera secuencial.

Ya que se tienen los tensores seleccionados, ya sea de un bloque CNP o de un filtro dentro del bloque last-CNP, hay varias mutaciones a realizar. Cada una sucede si un valor aleatorio r es menor que ϕ_a o ϕ_b, dependiendo del momento en el que se encuentre, ya sea el primero o el segundo, respectivamente.

Es menester mencionar que la selección, o creación de valores aleatorios viene a partir de una distribución discreta o distribución continua, ambas uniformes, dependiendo del caso.

La mutación propuesta sobre los pesos W y sesgo b de los filtros convolucionales se hace a partir de una suma de valores aleatorios. Se crea un tensor S con valores aleatorios en un rango especificado R de las mismas dimensiones que el tensor seleccionado W o b (dependiendo el caso), y solo a algunos de los valores, también seleccionados de manera aleatoria, son modificados sumando S. Las Ecuaciones 4.5 y 4.6, muestran la suma a realizar sobre los tensores a modificar para el siguiente tiempo $t + 1$.

$$W_{t+1} = W_t + S_W \tag{4.5}$$

$$b_{t+1} = b_t + S_b \tag{4.6}$$

Las mutaciones del tamaño de los filtros, ya sean convolucionales o de pooling, se hace con respecto a las dimensiones $v \times w$ y $s \times t$, respectivamente.

Para el caso de las funciones no lineales se cambian de manera aleatoria por otras, y para la operación de pooling, cambia de Max a Avg y vicerversa, ya que no hay más operaciones.

Las últimas tres mutaciones: (1) número de bloques CNP, (2) número de canales de salida z' (o mapas de características) y (3) número de filtros convolucionales, funcionan de la misma misma manera.

Como se puede ver en las mutaciones que modifican las dimensiones de algunos elementos dentro de la RNC, se incluye implícitamente una selección discreta de un valor entero e. Dicho valor e sirve para seleccionar un elemento a eliminar o insertar uno nuevo en ese lugar. Cuando un elemento se inserta, las cualidades y caraterísticas del elemento también son generadas de manera aleatoria con respecto a su contexto (tensor, valor, bloque o capa), de manera que las soluciones se mantengan congruentes y evitar generar inconsistencias.

4.6.3.3. Operador de mutación de la capa *FC*

En esta capa, pueden ser modificados los tensores W y b con respecto a sus valores, así como se definió en las Ecuaciones 4.5 y 4.6.

4.6.4. Recomposición de las soluciones

Las mutaciones y las cruzas pueden generar soluciones inconsistentes dentro de las RNC, por lo tanto, es necesario recomponer y mantener la coherencia dentro de éstas. Por ejemplo, si un bloque CNP es eliminado de una RNC, los bloques o capas (last-CNP o FC) adyacentes tendrán que modificarse para mantener

las dimensiones de los tensores consistentes de acuerdo a la modificación realizada. Si este paso no se realiza, las RNC generadas estarán incompletas o mal generadas.

5. Experimentos y resultados

5.1. Datos utilizados

Debido al desbalanceo de clases en la distribución original de los tweets, se decidió llevar a cabo el balanceo de las mismas a través de un submuestreo. Se eliminaron de manera aleatoria datos de aquellas clases con mayor número de ejemplos, para balancearse frente a las clases que menos contuvieran, todo esto garantizando el mantenimiento de las características principales de los datos. En la Tabla 5.1, se muestran la cantidad de tweets resultantes de este pre-procesamiento.

Sentimiento	Originales	Selección Aleatoria
Negativos	4582	3161
Neutrales	17450	3161
Positivos	3161	3161

Tabla 5.1: Número de tweets en la base de datos original, después de seleccionar aleatoriamente datos, para lidiar con el desbalanceo de clases

Después del pre-procesamiento, los tweets fueron transformados con el modelo *Word2Vec* y *BERT*, para posteriormente ejecutar los algoritmos.

5.2. Ambiente de Ejecución

La limpieza y transformación de los datos fueron ejecutadas en un equipo de cómputo AMD Ryzen 5 3500 con 4 núcleos a 2.10 GHz de CPU, 12 GB de RAM y Windows 10 de 64 bits.

Los experimentos de entrenamiento de los clasificadores y el algoritmo *DeepNEWT* fueron ejecutados en un ambiente con una CPU Intel®Xeon(R) con 12 núcleos a 1.90 GHz, 62.8 GB de RAM y Ubuntu 20.04 LTS.

Algunos experimentos, fueron ejecutados en una GPU de 12 GB NVIDIA Tesla K80 con 2496 núcleos CUDA GDDR5, provista por Google Colaboratory, y otros más en una GPU de 6 GB NVIDIA GeForce GTX 980 Ti con 2816 núcleos CUDA.

5.3. Forma de evaluar los resultados

Al finalizar las ejecuciones de los algoritmos, y éstos habiendo generado los resultados pertinentes, van a ser evaluados con técnicas estadísticas. Partiendo del uso de la prueba no-paramétrica *Kolmogorov-Smirnov*, se puede saber si la distribución de cada conjunto de datos, se ajusta o no a una *distribución normal*, a partir de aquí, y dependiendo del resultado de la prueba, hay dos variantes para cada caso: (1) cuando los resultados no vienen de una distribución normal, se usará la prueba *no-paramétrica Friedman aligned ranks*; por el contrario (2) si la prueba muestra que la distrubución de los resultados se ajusta a una normal, la prueba paramétrica *ANOVA* (Análisis de Varianza) se aplicaría sobre estas muestras.

Dada la naturaleza de estas pruebas multivariables, es necesario realizar un análisis de comparación *post-hoc*, que involucran, nuevamente, el tipo de las muestras obtenidas. Para el caso de que los datos no pertenezcan a una distribución normal, la prueba *Friedman aligned* con la corrección de *Bonferroni-Dunn* sobre el $p\text{-}value$ será aplicada, por otro lado, si los datos se ajustan a una distribución normal, la prueba que se aplicará será la prueba *Tukey*, respectivamente.

5.4. Resultados

Diversos experimentos fueron ejecutados para analizar y evaluar el desempeño del algoritmo *DeepNEWT*: (1) búsqueda preliminar de RNCs utilizando el conjunto de datos representados con *Word2Vec*, (2) búsqueda de RNCs utilizando variaciones en los parámetros del algoritmo a través de un conjunto reducido de datos representado con *Word2Vec*, (3) calibración, ajuste y entrenamiento de las estructuras encontradas en los experimentos preliminares utilizando los datos representados con *BERT* frente a los clasificadores tradicionales, y (4) ejecución de los algoritmos tradicionales y *DeepNEWT* utilizando los datos representados con *Word2Vec*. En la Tabla 5.4 se muestra un esquema donde se visualizan los

	Experimento	Proceso	Datos	Resultados	Sección
1	Búsqueda de la arquitectura y pesos de una RNC	Ejecutar *DeepNEWT* para generar estructuras preliminares	*Word2Vec*	Dos RNC con estructura y pesos inicializados	5.4.1
2	Búsqueda de los mejores parámetros para *DeepNEWT*	Ejecutar *DeepNEWT* con diferentes valores de parámetros utilizando menos datos	*Word2Vec* (150 sentencias)	Precisión por cada combinación de parámetros	5.4.2
3	Comparación de las estructuras preliminares frente a los clasificadores de AM tradicional	Calibrar y ejecutar las estructuras preliminares para *BERT*	*BERT* (10-fold)	Precisión por cada partición de los datos	5.4.3
4	Comparación de *DeepNEWT* frente a los clasificadores de AM tradicional	Calibrar y ejecutar *DeepNEWT* con los parámetros utilizados en (1)	*Word2Vec* (5-fold)	Precisión por cada partición de los datos	5.4.4

Tabla 5.2: Esquema de experimentos realizados y sus correspondientes resultados

experimentos llevados a cabo, el proceso, los datos utilizados y los resultados generados por éstos.

Para evaluar el desempeño de clasificación de los algoritmos se realizaron diferentes experimentos a través de una validación cruzada (CV, por sus siglas en Inglés), con k particiones con las clases bien balanceadas, es decir, una validación cruzada *estratificada*.

En esta sección, se presentan los resultados de las CV de todos los algoritmos y su clasificación. Se muestran, por otro lado, los resultados de los parámetros resultantes de las RNC generadas por los experimentos preliminares de *DeepNEWT* en comparación con la RNC expuesta en la Sección 4.5; y finalmente, la evaluación de los resultados.

Las distintas CV ejecutadas, respetan las particiones de los tweets, es decir, los mismos tweets ya sea representados por *Word2Vec* o *BERT* serán los mismos para cada una de las *partes*.

Antes de comparar los resultados, se tienen que verificar a qué tipo de distribución pertenecen. Para este caso, se hace uso de la prueba *Kolmogorov-Smirnov*. Su hipótesis nula H_0 se acepta cuando la distribución de los resultados se ajusta a la distribución normal, por el contrario, se rechaza H_0 y se acepta la hipótesis alternativa indicando que la distribución de los resultados no se ajusta a una

distribución normal. Los resultados de la CV de los clasificadores y el algoritmo *DeepNEWT* en todas las etapas de los experimentos, fueron evaluados y la prueba indicó que las muestras de resultados no se ajustan a una distribución normal.

Por lo anterior, las pruebas a utilizar, son: (1) la prueba *Friedman aligned ranks*, para verificar diferencias entre varios conjuntos de datos, y (2) el análisis de comparación por pares, mediante la prueba *post-hoc Friedman aligned* con la corrección de *Bonferroni-Dunn* sobre el $p\text{-}value$, para ubicar las diferencias significativas entre los conjuntos de resultados.

La prueba *post-hoc Friedman aligned* determina una conclusión a través de un $p\text{-}value$; si $p\text{-}value < \alpha$, las muestras contienen una diferencia significativa; si *p-value* $\geq \alpha$, las muestras son iguales y se demuestra que no se encontraron diferencias significativas, siendo $\alpha = 0{,}05$.

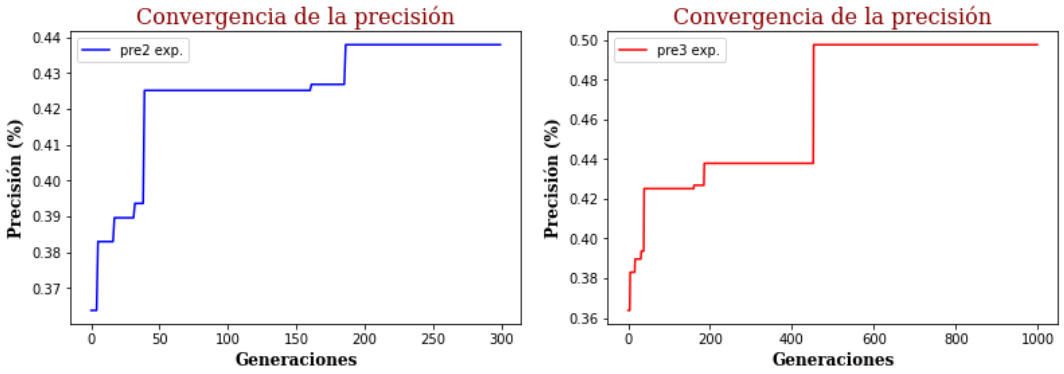
5.4.1. Experimentos preliminares con *DeepNEWT*

El primer conjunto de experimentos tiene que ver con la ejecución de *DeepNEWT*. El algoritmo fue ejecutado, de manera preliminar, dos veces con diferentes parámetros: $pre2$ y $pre3$, generando una RNC por cada ejecución. Dichas RNC contienen una estructura neuronal y pesos ya entrenados dentro de *DeepNEWT*. Para la ejecución del algoritmo *DeepNEWT* y la búsqueda de estas redes fue utilizado el conjunto de datos representado con *Word2Vec*. Los datos se utilizaron en su totalidad, generando una precisión del conjunto que se tomó como función de aptitud del algoritmo.

En la Tabla 5.3 se muestran los parámetros utilizados en los experimentos preliminares de *DeepNEWT*, los cuales fueron encontrados de manera empírica. Se aprecian distintos valores de parámetros: (1) T, es el número de padres a seleccionar, se mantiene con los mismos valores para ambos experimentos, (2) p, es el número de individuos a seleccionar por cada torneo, (3) ϕ_a, que es la probabilidad de mutar un individuo en el primer momento, y (4) ϕ_b que es la probabilidad de mutar un individuo en el segundo momento.

Parámetro	*pre2*	*pre3*
Generaciones	300	1000
Población	20	20
T	3	3
p	2	2
ϕ_a	0,8	0,8
ϕ_b	0,7	0,7
R	$[-2, 2]$	$[-2, 2]$

Tabla 5.3: Parámetros utilizados en la ejecución de *DeepNEWT*



(a) (b)

Figura 5.1: Convergencia de la precisión en los experimentos preliminares $pre2$ y $pre3$

El parámetro $Generaciones$ es el único que cambia en estos dos conjuntos, se esperaba observar cambios en el experimento con los parámetros de $pre3$, aumentando a 1000 el número de generaciones. El resto de parámetros, fueron seleccionados por las siguientes razones: (1) el valor del parámetro T, para generar el mínimo de individuos al finalizar la ejecución de los operadores de cruza y muta, y mantener, de esta manera, características de los individuos creados en generaciones anteriores que pudieran aportar mejoras a las futuras estructuras; (2) parámetro p, normalmente el valor asignado es el predeterminado para este tipo de torneo; (3) parámetros ϕ_a y ϕ_b, mantienen la condición $0 < \phi_b < \phi_a < 1$, y sus valores, para estos experimentos, son mayores a $0{,}5$, para generar más mutaciones sobre un individuo en ambos momentos, de manera que se puedan generar nuevas características que podrían ser útiles en nuevas generaciones; (4) el valor de R, asignado así, para generar valores aleatorios uniformemente distribuidos en un rango reducido, y realizar mutaciones, sobre W ó b, que mantengan características anteriores de los individuos.

Además se definen algunos valores mínimos y máximos de los elementos de las RNC, que son utilizados para ambos experimentos preliminares. El número de bloques CNP permitidos se encuentra en el rango $[0, 5]$, el valor z', correspondiente al número de mapas de características, para bloques CNP y last-CNP se extrae del rango $[10, 50]$, y, por último, para los tamaños de filtros convolucionales y de pooling, el rango es $[3, 6]$.

Se graficaron las convergencias de la ejecución de los experimentos, las cuales se muestran en la Figura 5.1. Se puede observar que la precisión máxima lograda es de casi el 50 %. Además, se puede notar que el experimento $pre2$ genera una precisión más baja que el $pre3$. Asimismo, el experimento $pre3$ genera una mejora en la precisión de casi 6 puntos porcentuales con respecto a $pre2$, y después de la generación 420, la convergencia se mantuvo igual hasta finalizar la ejecución del algoritmo. Esto es debido a que los operadores de cruza y mutación encontraron una estructura y pesos en dicha generación que logró mantenerse como la mejor hasta finalizar la búsqueda.

5.4.2. Búsqueda de RNC para la mejora de la precisión con *DeepNEWT*

Con el fin de mejorar la precisión de los experimentos preliminares y de encontrar un conjunto de parámetros efectivo, un segundo conjunto de experimentos fue ejecutado utilizando variaciones en los parámetros del algoritmo *DeepNEWT*. Para estos experimentos se utilizó un conjunto de datos representados con *Word2Vec* que contiene 150 tweets con las tres clases (positivos, negativos y neutrales), bien balanceadas.

Diferentes experimentos se realizaron para medir el desempeño del algoritmo *DeepNEWT* usando combinaciones de parámetros que difieren de los experimentos preliminares, con respecto al número T de padres seleccionados para cruzar y la probabilidad de mutación en el primer y segundo momento, ϕ_a y ϕ_b, respectivamente. Debido a que el costo computacional es excesivo (alrededor de 12 horas utilizando todos los tweets), una ejecución por configuración fue realizada. Como parámetros predeterminados, se establecieron 100

Exp.	T	ϕ_a	ϕ_b	**%**	**Exp.**	T	ϕ_a	ϕ_b	**%**	**Exp.**	T	ϕ_a	ϕ_b	**%**
$A1$	6	0,2	0,1	47,33	$B1$	12	0,2	0,1	50,67	$C1$	18	0,2	0,1	48,67
$A2$	6	0,4	0,1	49,33	$B2$	12	0,4	0,1	47,33	$C2$	18	0,4	0,1	51,33
$A3$	6	0,4	0,2	46,67	$B3$	12	0,4	0,2	47,33	$C3$	18	0,4	0,2	52
$A4$	6	0,6	0,1	48	$B4$	12	0,6	0,1	46,67	$C4$	18	0,6	0,1	50,67
$A5$	6	0,6	0,2	46,67	$B5$	12	0,6	0,2	48,67	$C5$	18	0,6	0,2	50,67
$A6$	6	0,6	0,4	49,33	$B6$	12	0,6	0,4	45,33	$C6$	18	0,6	0,4	47,33

Tabla 5.4: Parámetros de los experimentos y sus respectivos porcentajes de precisión finales

generaciones y 20 individuos para correr los experimentos. Nuevamente, la función de aptitud del algoritmo es la precisión obtenida por los datos representados con *Word2Vec*.

La Tabla 5.4 contiene los resultados obtenidos y sus parámetros asociados a cada experimento. En la Figura 5.2 se encuentran las gráficas de convergencia para cada experimento (A, B y C), en las cuales se agrega una gráfica de convergencia con respecto a la media de las precisiones obtenidas.

Con respecto a la precisión lograda, en los experimentos con 18 padres seleccionados (experimentos C), en 4 de 6 experimentos se llegó a más del 50 % de precisión, donde la precisión más alta es de 52 % con $\phi_a = 40\,\%$ y $\phi_b = 20\,\%$. Todos los experimentos con 6 padres seleccionados (experimentos A), se mantienen por debajo del 50 % de precisión, pero la gráfica de convergencia con respecto a la media (Figura 5.2(d)), indica que los experimentos con 6 padres seleccionados son mejores que los experimentos con 12 padres seleccionados (experimentos B).

Los experimentos con 12 padres seleccionados tienen una convergencia más lenta con respecto a los experimentos con 6 y 18 padres seleccionados. Por otro lado, en los experimentos con 18 padres seleccionados, el algoritmo produce, con respecto a la convergencia de la precisión media, mejor precisión que le resto de los experimentos. En tres casos (experimentos $C2$, $C3$ y $C5$), la convergencia fue mejor hasta alcanzar una precisión mayor que el 50 % después de 80 generaciones.

Como conclusión de estos experimentos, la búsqueda de *DeepNEWT* se beneficia cuando el número de padres seleccionados se incrementa, particularmente cuando los valores de la mutación se

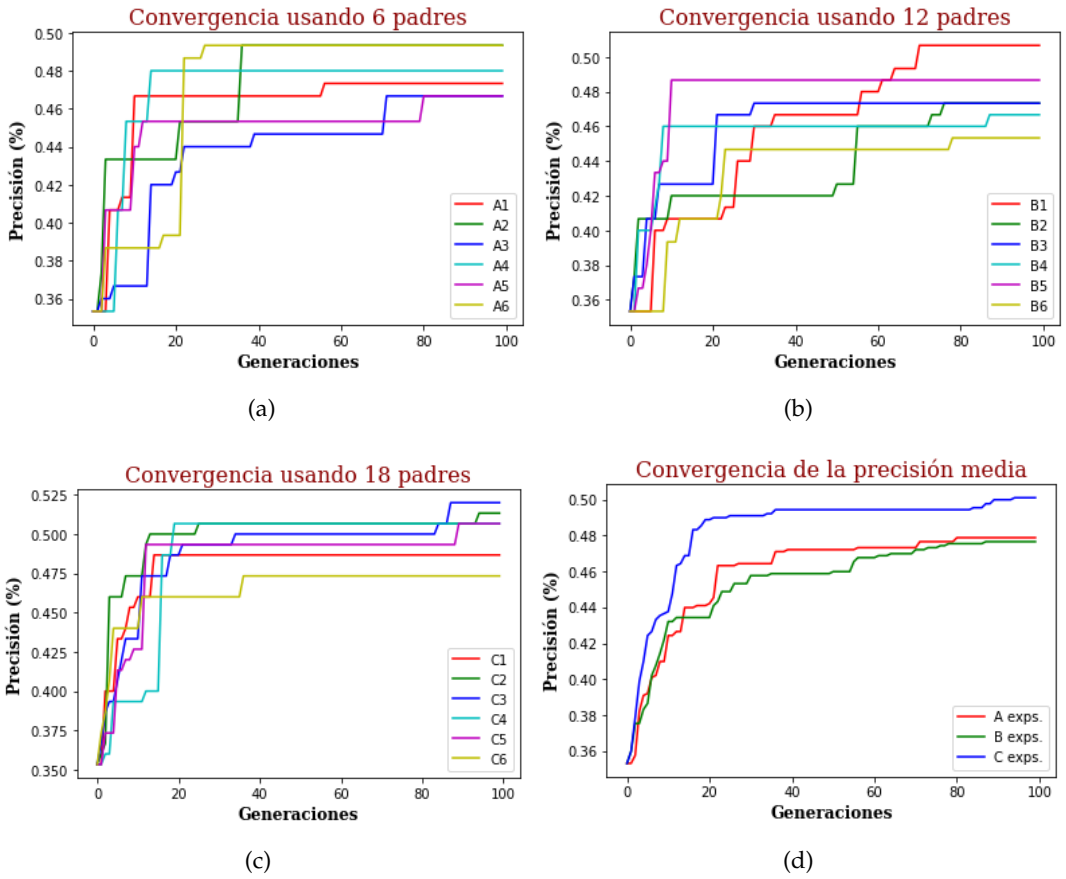


Figura 5.2: Convergencia de precisión con diferente número de padres seleccionados, y su gráfica de convergencia con respecto a la precisión media

encuentran colocados en medio de los rangos probados (*i. e.*, $0{,}4$ Y $0{,}2$ por cada probabilidad de mutación).

5.4.3. AM tradicional y entrenamiento de las estructuras preliminares encontradas por *DeepNEWT* para *BERT*

En esta etapa de experimentación, se realizaron los entrenamientos de los clasificadores de AM tradicional, la RNC extraída de la literatura y las estructuras encontradas en los experimentos preliminares de *DeepNEWT*. Todas las RNC fueron calibradas y ajustadas para ser utilizadas con los datos representados con *BERT*. El objetivo de estos experimentos es evaluar la precisión en una CV con un $k = 10$.

Los clasificadores de AM tradicional: IB y MSV fueron tomados de la biblioteca *Scikit-Learn* con versión $0{,}22$. Los clasificadores fueron

Experimento	Iteraciones	Taza de aprendizaje
es1	200	0.001
es2	1000	0.001
es3	5000	0.001
es4	10000	0.001

Tabla 5.5: Parámetros del experimeto es de $pre2$ y $pre3$ con *BERT*

ejecutados con los parámetros predeterminados de la biblioteca. MSV usa la función *Radial Basis* como kernel, e IB, para el caso de la verosimilitud, ya que se trabaja con datos continuos, se calcula a partir de una distribución Gaussiana. La RNC, expuesta en la Sección 4.5, fue entrenada durante 200 iteraciones con el optimizador *AdaDelta*, un *dropout* de $0{,}5$ para la capa FC y una taza de aprendizaje de $0{,}0001$.

Por otro lado, las estructuras preliminares generadas por *DeepNEWT* fueron calibradas y ajustadas para ser entrenados con los datos representados con *BERT*. El número de filtros, las funciones de activación y las operaciones de pooling, se mantuvieron para este entrenamiento. Por el lado de los pesos encontrados por *DeepNEWT*, fueron reiniciados de manera aleatoria y nuevamente aprendidos mediante el optimizador *AdaDelta* y un *dropout* de $0{,}5$ en la capa FC. Para facilitar la lectura, el prefijo es se refiere al entrenamiento de las RNC encontradas en los experimentos $pre2$ y $pre3$. En la Tabla 5.5, se muestran los parámetros utilizados para los experimentos es. Estos parámetros fueron seleccionados de manera empírica.

En cada *parte* dentro de la CV, fue entrenado un nuevo modelo con el 90 % de datos y el 10 % para prueba. Esto quiere decir que los modelos reiniciaron sus parámetros a un estado inicial, pasada la etapa de prueba de cada modelo con el 10 % de datos. En el caso de las RNC, se reiniciaron los pesos y sesgos previamente entrenados.

En la Tabla 5.6 se muestran los resultados de la ejecución de MSV, IB y RNC junto con el entrenamiento de la estructura $pre2$, mientras que en la Tabla 5.7 se muestran los mismos resultados de los clasificadores tradicionales y la RNC en conjunto con el entrenamiento de la estructura $pre3$.

En primera instancia, los resultados de la precisión mejoran con respecto a los previamente obtenidos por *DeepNEWT* utilizando

Parte	MSV	IB	RNC	*es1-pre2*	*es2-pre2*	*es3-pre2*	*es4-pre2*
1	**65.54**	33.40	62.17	62.28	63.65	64.49	64.59
2	65.75	33.72	62.59	64.27	65.65	66.81	**67.23**
3	63.65	33.61	62.49	62.80	**65.44**	65.12	64.49
4	**64.45**	33.86	61.71	61.81	64.03	**64.45**	64.24
5	**67.19**	33.86	62.66	62.97	65.61	66.66	67.09
6	66.77	33.65	62.03	65.40	**66.88**	66.77	66.67
7	**64.87**	33.86	62.45	63.61	64.66	**64.87**	64.03
8	64.87	33.54	63.71	63.71	64.56	64.56	**65.29**
9	65.30	33.44	61.92	61.29	63.50	**65.40**	64.03
10	62.13	33.76	60.02	61.39	**64.87**	64.03	63.40
Mejor	67.19	33.86	63.71	65.40	66.88	66.81	67.23
Mediana	65.09	33.69	62.31	62.89	64.77	65.00	64.54
Peor	62.13	33.40	60.02	61.29	63.50	64.03	63.40
Media	65.05	33.67	62.18	62.95	64.89	65.32	65.11
Desv. Est.	1.46	0.17	0.94	1.32	1.04	1.06	1.40

Tabla 5.6: Porcentajes de precisión de los datos correctamente clasificados de las 10 particiones de prueba de los experimentos de AM tradicional y la estructura preliminar $pre2$ de RNC ajustada para *BERT*

Word2Vec. Lo anterior puede deberse a que dicho modelo fue entrenado únicamente con los tweets, mientras que, en este caso, *BERT* es un modelo previamente entrenado con una base de datos mucho más grande y que ofrece más información acerca de los datos.

La CV con los datos de *BERT*, muestra resultados cercanos entre sí con respecto a MSV y RNC, mientras que los de IB se encuentran por debajo de los anteriores, llegando a obtener alrededor de 33 % de precisión. Con respecto a los mejores resultados, la MSV supera en 3.48 puntos porcentuales a RNC. En el entrenamiento de los estructuras preliminares, $pre2$ y $pre3$, generadas por *DeepNEWT*, muestran resultados competitivos frente a la MSV.

En los experimentos es para $pre2$, se puede observar una tendencia a mejorar los resultados en la medida que el número de iteraciones aumenta. El penúltimo de estos experimentos, $es3$-$pre2$, presenta cuatro $partes$ ganadas ($partes$ 2, 3, 9, 10) y dos empatadas ($partes$ 4 y 7) con respecto a MSV. También se pueden observar medianas bastante similares y competitivas entre sí. La tendencia hacia la mejora termina cuando las iteraciones de entrenamiento de la estructura $pre2$ son 10000, ya que en este caso, la mediana es menor que en $es3$ y los resultados de la MSV, sin embargo, mantiene cuatro partes ganadas ($partes$ 2, 4, 8 y 10).

Parte	MSV	IB	RNC	*es1-pre3*	*es2-pre3*	*es3-pre3*	*es4-pre3*
1	**65.54**	33.40	62.17	60.80	63.33	65.01	64.28
2	65.75	33.72	62.59	61.85	65.75	66.17	**66.91**
3	63.65	33.61	62.49	60.59	64.38	**65.44**	65.12
4	64.45	33.86	61.71	61.39	63.71	64.35	**65.51**
5	67.19	33.86	62.66	60.97	65.08	66.98	**68.04**
6	**66.77**	33.65	62.03	63.19	65.51	**66.77**	66.35
7	64.87	33.86	62.45	62.13	64.77	64.56	**66.14**
8	64.87	33.54	63.71	61.92	65.08	**66.46**	64.98
9	**65.30**	33.44	61.92	57.91	64.45	64.87	64.45
10	62.13	33.76	60.02	60.34	63.19	**64.87**	62.66
Mejor	67.19	33.86	63.71	63.19	65.75	66.98	68.04
Mediana	65.09	33.69	62.31	61.18	64.61	65.23	65.32
Peor	62.13	33.40	60.02	57.91	63.19	64.35	62.66
Media	65.05	33.67	62.18	61.11	64.52	65.55	65.44
Desv. Est.	1.46	0.17	0.94	1.41	0.89	0.97	1.51

Tabla 5.7: Porcentajes de precisión de los datos correctamente clasificados de las 10 particiones de prueba de los experimentos de AM tradicional y la estructura preliminar $pre3$ de RNC ajustada para *BERT*

Por otro lado, el experimento es para $pre3$, da como resultado una tendencia a la alza, semejante a la del experimento es de $pre2$. Además, el experimento $es3$-$pre3$ obtiene una mediana y media más altas que la MSV para *BERT*. El mejor experimento es $es4$, donde se obtienen siete partes a favor, mediana y media de 65.32 y 65.44, respectivamente. Todo esto, favorable frente a la MSV, la cual tiene, en contraste, 3 partes ganadas y mediana y media menores.

Los resultados de estos experimentos fueron comparados. La prueba *Friedman aligned ranks* fue ejecutada realizando las siguientes comparaciones, tomando como conjunto de datos *control* a los resultados generados por los experimentos es-pre:

- Clasificadores tradicionales de AM (MSV, IB) y la RNC *versus* experimentos es de $pre2$ (de la Tabla 5.6).
- Clasificadores tradicionales de AM (MSV, IB) y la RNC *versus* experimentos es de $pre3$ (de la Tabla 5.7).

Tal prueba da como resultado, por cada comparación, un $p\text{-}value < \alpha$, respectivamente. Esto significa que existen diferencias entre los conjuntos de datos comparados. La prueba *Friedman aligned* se ejecuta para realizar el análisis *post-hoc*, agregando la corrección de *Bonferroni-Dunn* sobre el $p\text{-}value$:

Prueba	**Friedman Aligned ranks, α = 0.05 ($pre2$)**					
Comparación 1	**MSV**	**IB**	**RNC**	***es1-pre2***	***p-value***	**Conclusión**
rangos	32.5	5.5	17.55	23.75	9.20E-06	Diferentes
Post-hoc (p-$value$ no ajustado)	0.0285	0.0005	0.2357	-	-	
Bonferroni-Dunn (p-$value$ ajustado)	0.0855	**0.0015**	0.7071	-	-	
Comparación 2	**MSV**	**IB**	**RNC**	***es2-pre2***	***p-value***	**Conclusión**
rangos	32.2	5.5	15.6	29.7	1.35E-05	Diferentes
Post-hoc (p-$value$ no ajustado)	0.7742	3.67E-06	0.007	-	-	
Bonferroni-Dunn (p-$value$ ajustado)	1	**1.10E-05**	**0.021**	-	-	
Comparación 3	**MSV**	**IB**	**RNC**	***es3-pre2***	***p-value***	**Conclusión**
rangos	29.75	5.5	15.5	31.25	1.11E-05	Diferentes
Post-hoc (p-$value$ no ajustado)	0.7742	8.42E-07	0.0026	-	-	
Bonferroni-Dunn (p-$value$ ajustado)	1	**2.53E-06**	**0.0078**	-	-	
Comparación 4	**MSV**	**IB**	**RNC**	***es4-pre2***	***p-value***	**Conclusión**
rangos	30.3	5.5	15.5	30.7	1.21E-05	Diferentes
Post-hoc (p-$value$ no ajustado)	0.939	1.44E-06	0.0036	-	-	
Bonferroni-Dunn (p-$value$ ajustado)	1	**4.31E-06**	**0.0108**	-	-	

Tabla 5.8: Comparación de los clasificadores de AM tradicional con los experimentos es de $pre2$; los valores en negritas muestran diferencias significativas cuando se compara la estructura $pre2$ contra un clasificador tradicional (MSV, IB o RNC)

- MSV, IB y RNC *versus* es con $pre2$
- MSV, IB y RNC *versus* es con $pre3$

Los resultados de estas pruebas, tomando como datos de *control* a los resultados generados por las estructuras $pre2$ y $pre3$, se muestran en las Tablas 5.8 y 5.9, respectivamente. El p-$value$ ajustado por la prueba *Bonferroni-Dunn* se toma en cuenta.

Por un lado, los resultados de las comparaciones con respecto a los datos generados por $pre2$ muestran, en su totalidad, similitudes entre los experimentos es-$pre2$ frente a MSV. Mientras que, cuando es-$pre2$ se compara contra RNC e IB, los resultados son, en el caso de la *comparación* 1, contra RNC, similares, y el resto de pruebas, son todas con diferencias significativas, donde los $rangos$ se mantienen por debajo del $rango$ de es-$pre2$.

Por el lado de los resultados de las comparaciones con respecto a los datos generados por $pre3$ muestran, nuevamente, diferencias significativas en las $comparaciones$ 2, 3 y 4, con respecto a IB y RNC, los cuales tienen $rangos$ menores. Y con respecto a MSV, el

Prueba	Friedman Aligned ranks, α = 0.05 ($pre3$)					
Comparación 1	**MSV**	**IB**	**RNC**	***es1-pre3***	***p-value***	**Conclusión**
rangos	35.5	5.5	24	17	6.48E-06	Diferentes
Post-hoc (p-$value$ no ajustado)	0.0004	0.0278	0.1806	-	-	
Bonferroni-Dunn (p-$value$ ajustado)	**0.0012**	0.0834	0.5418	-	-	
Comparación 2	**MSV**	**IB**	**RNC**	***es2-pre3***	***p-value***	**Conclusión**
rangos	32.45	5.5	15.5	28.55	9.90E-06	Diferentes
Post-hoc (p-$value$ no ajustado)	0.4557	1.04E-05	0.0126	-	-	
Bonferroni-Dunn (p-$value$ ajustado)	1	**3.12E-05**	**0.0378**	-	-	
Comparación 3	**MSV**	**IB**	**RNC**	***es3-pre3***	***p-value***	**Conclusión**
rangos	28.85	5.5	15.5	32.15	1.05E-05	Diferentes
Post-hoc (p-$value$ no ajustado)	0.5279	3.44E-07	0.0014	-	-	
Bonferroni-Dunn (p-$value$ ajustado)	1	**1.03E-06**	**0.0042**	-	-	
Comparación 4	**MSV**	**IB**	**RNC**	***es4-pre3***	***p-value***	**Conclusión**
rangos	29.2	5.5	15.5	31.8	1.07E-05	Diferentes
Post-hoc (p-$value$ no ajustado)	0.619	4.89E-07	0.0018	-	-	
Bonferroni-Dunn (p-$value$ ajustado)	1	**1.47E-06**	**0.0054**	-	-	

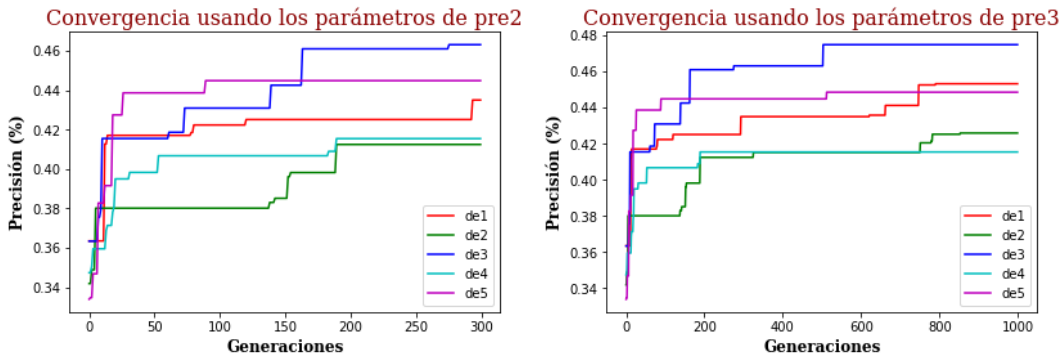
Tabla 5.9: Comparación de los clasificadores de AM tradicional con los experimentos es de $pre3$; los valores en negritas muestran diferencias significativas cuando se compara la estructura $pre3$ contra un clasificador tradicional (MSV, IB o RNC)

experimento $es1$-$pre3$ generó diferencias significativas, esta vez favorable para MSV debido a su $rango$ ($33{,}5$ contra 17).

5.4.4. AM tradicional y *DeepNEWT* para *Word2Vec*

Esta segunda etapa de experimentos que involucran AM tradicional, la RNC extraída de la literatura y el algoritmo *DeepNEWT*, utilizan todos los tweets representados con *Word2Vec*. Esto con la finalidad de evaluar el desempeño de *DeepNEWT* frente al resto de clasificadores. Una CV de $k = 5$ se empleó para evaluar la precisión. Y un análisis de la búsqueda de las RNC mediante *DeepNEWT* es agregado.

Los clasificadores fueron tomados de la biblioteca *Scikit-Learn* con versión $0{,}22$. Los clasificadores fueron ejecutados con los parámetros predeterminados de la biblioteca. MSV usa la función *Radial Basis* como kernel, e IB, para el caso de la verosimilitud, ya que se trabaja con datos continuos, se calcula a partir de una distribución Gaussiana. Se agrega a esta serie de experimentos la ejecución de la RNC, expuesta en la Sección 4.5. Esta RNC fue entrenada durante



(a) (b)

Figura 5.3: Convergencia de precisión de *DeepNEWT* con respecto al conjunto de entrenamiento utilizando los parámetros de los experimentos preliminares, $pre2$ y $pre3$

Parte	DeepNEWT (*de-pre2*)	DeepNEWT (*de-pre3*)
1	43.5	**45.32**
2	41.23	**42.59**
3	46.31	**47.48**
4	**41.54**	**41.54**
5	44.48	**44.85**
Mejor	46.31	47.48
Mediana	43.5	44.85
Peor	41.23	41.54
Media	43.41	44.36
Desv. Est.	2.11	2.34

Tabla 5.10: Porcentajes de precisión de los datos correctamente clasificados de las 5 particiones de entrenamiento durante la búsqueda de RNC mediante *DeepNEWT* en la CV

200 iteraciones con el optimizador *AdaDelta*, *dropout* de $0{,}5$ para la capa FC y una taza de aprendizaje de $0{,}0001$. Para el caso de los parámetros de *DeepNEWT*, fueron utilizados los mismos que en los experimentos preliminares $pre2$ y $pre3$. Los tweets transformados por *Word2Vec* fueron utilizados para correr los algoritmos, incluyendo *DeepNEWT*.

En cada parte dentro de esta CV, fue entrenado un nuevo modelo con el 80 % de los datos y el 20 % para prueba. Esto quiere decir que los modelos reiniciaron sus valores y parámetros a un estado inicial.

Con respecto al entrenamiento y búsqueda de las RNC dentro de *DeepNEWT*, se extrajeron las diferentes gráficas de convergencia para ambos conjuntos de parámetros, $pre2$ y $pre3$, utilizando únicamente el conjunto de entrenamiento. Se ejecutaron 5 entrenamientos (de),

Parte	IB	MSV	RNC	DeepNEWT (*de-pre2*)	DeepNEWT (*de-pre3*)
1	39.33	**54.4**	38.06	44.44	45.39
2	33.42	**53.66**	43.75	41.7	43.23
3	33.9	**54.61**	36.32	47.92	48.37
4	41.09	**53.06**	36.18	42.41	42.41
5	36.45	**52.69**	36.66	42.3	42.14
Mejor	41.09	54.61	43.75	47.92	48.37
Mediana	36.45	53.66	36.66	42.41	43.23
Peor	33.42	52.69	36.18	41.7	42.14
Media	36.84	53.68	38.19	43.75	44.31
Desv. Est.	3.34	0.83	3.19	2.55	2.60

Tabla 5.11: Porcentajes de precisión de los datos correctamente clasificados de las 5 particiones de prueba

uno por cada partición de la CV. En la Figura 5.3 se muestran las gráficas de convergencia correspondientes a los entrenamientos de. En la Tabla 5.10 se muestran los resultados de la precisión obtenida en el conjunto de entrenamiento en cada $parte$ de la CV.

A diferencia de los experimentos previos, los de mejora (Sección 5.4.2), el entrenamiento de en todas las fases se mantiene por debajo del 50 % de precisión. En ambos experimentos con los parámetros de $pre2$ y $pre3$, el entrenamiento $de3$ gana, y se mantiene arriba desde antes de la generación 200 hasta terminar, contrario al resto. El entrenamiento $de2$ y $de4$ fueron los que menos precisión obtuvieron luego de 1000 generaciones, con $42{,}59\,\%$ y $41{,}54\,\%$, respectivamente. El entrenamiento con los parámetros $pre3$ obtiene una mejora, en promedio, de casi un punto porcentual con respecto al entrenamiento utilizando los parámetros de $pre2$.

En la Tabla 5.11 se muestran los resultados de precisión luego de evaluar los modelos generados utilizando el conjunto de prueba de cada $parte$ de la CV. Los resultados favorecen a la MSV en todas sus partes, obteniendo en promedio un $53{,}58\,\%$ de precisión. Esto representa casi 10 puntos porcentuales con respecto a de-$pre3$, que obtuvo mejor promedio y mediana que de-$pre2$, IB y la RNC tomada de la literatura.

Con respecto a los resultados generados por *DeepNEWT*, entre sí, tienen una diferencia de poco más de medio punto porcentual, lo que apunta a que el conjunto $pre3$ de parámetros tiene diferencias que lo hacen ganar en todos las $partes$ de la CV con respecto a IB, de-$pre2$ y

Prueba	**Friedman Aligned ranks, α = 0.05 (*DeepNEWT*)**					
Comparación 1	**MSV**	**IB**	**RNC**	***de-pre2***	***p-value***	**Conclusión**
rangos	18	5.2	6.6	12.2	0.0088	Diferentes
Post-hoc (p-$value$ no ajustado)	0.1211	0.0613	0.1344	-	-	-
Bonferroni-Dunn (p-$value$ ajustado)	0.3633	0.1839	0.4032	-	-	-
Comparación 2	**MSV**	**IB**	**RNC**	***de-pre3***	***p-value***	**Conclusión**
rangos	18	5.2	6.4	12.4	0.0076	Diferentes
Post-hoc (p-$value$ no ajustado)	0.1345	0.0543	0.1089	-	-	-
Bonferroni-Dunn (p-$value$ ajustado)	0.4035	0.1629	0.3267	-	-	-

Tabla 5.12: Comparación de los resultados de clasificación en el conjunto de prueba de los clasificadores de AM tradicional, la RNC y los experimentos de

la RNC.

Para evaluar las diferencias significativas, se ejecutó la prueba *Friedman aligned ranks* realizando una comparación entre MSV, IB, RNC y los resultados generados por los experimentos de-$pre2$, de-$pre3$. Ambas pruebas, tienen como resultado un p-$value < \alpha$, lo cual muestra que existen diferencias entre los conjuntos de datos. La prueba *post-hoc Friedman aligned* con la corrección de *Bonferroni-Dunn* del p-$value$, se ejecuta para llevar a cabo las siguientes comparaciones:

- IB, MSV y RNC *versus* de con $pre2$
- IB, MSV y RNC *versus* de con $pre3$

En la Tabla 5.12 se muestran los resultados del análisis *post-hoc*. Se puede observar que las diferencias signficativas que se previeron, son correctas, solo con respecto a los $rangos$ obtenidos, donde los resultados de los experimentos $pre2$ y $pre3$ se mantienen superiores a los de IB y RNC, pero no es el caso frente a MSV. Además, puede apreciarse que no existen diferencias significativas entre los resultados obtenidos por los algoritmos en cuestión, a través del análisis *post-hoc* y la corrección del p-$value$, tomando como $control$ los experimentos pre.

5.4.5. Estructuras de RNC obtenidas por *DeepNEWT*

Las estructuras resultantes que fueron producidas por *DeepNEWT* en los experimentos preliminares, $pre2$ y $pre3$, se describen de manera detallada en esta sección. Estas estructuras son menos complejas, con respecto a la RNC propuesta originalmente, y coincide con lo esperado en la *Hipótesis* de esta tesis. A esto se le suma que dichas RNC superan a la RNC original usando *BERT* y *Word2Vec*, habiendo generado las estructuras y pesos utilizando *DeepNEWT*.

Después de ejecutado *DeepNEWT*, las RNC resultantes de $pre2$ y $pre3$ fueron extraídas y utilizadas para los experimentos es. Estas estructuras de RNC se muestran visualmente en la Figura 5.4 y en la Figura 5.5. Cada una de éstas finalizó dentro del algoritmo como una estructura similar a la RNC original (ver Sección 4.5). Únicamente, el bloque last-CNP y la capa FC fueron utilizadas por estas estructuras.

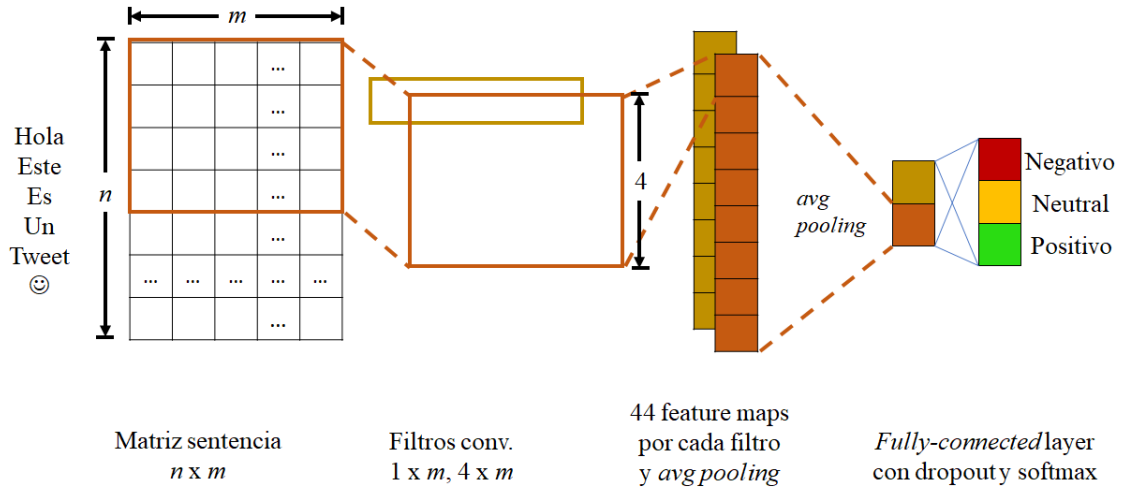


Figura 5.4: Estructura de RNC encontrada por *DeepNEWT* en el experimento $pre2$

Cada una de estas estructuras contiene 2 filtros convolucionales con dimensiones $1 \times m$, $pre2$ por un lado tiene un filtro de $4 \times m$ y, por el otro, $pre3$ tiene un filtro de $3 \times m$. El algoritmo encontró, después de 1000 generaciones, una estructura menos compleja, incluso, que la de $pre2$. Los filtros convolucionales generan, además, 44 y 12 mapas de características, z', en $pre2$ y $pre3$, respectivamente.

La no linealidad utilizada después de la operación de convolución es, para ambos casos, la PReLU cuyo parámetro a es el predeterminado por la biblioteca *PyTorch* en $0{,}25$.

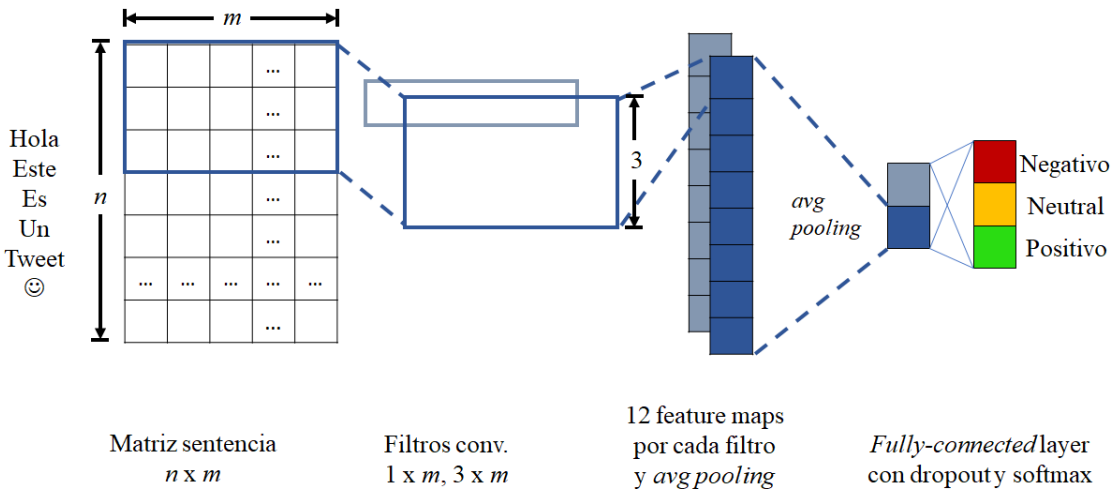


Figura 5.5: Estructura de RNC encontrada por *DeepNEWT* en el experimento $pre3$

Elemento	**RNC original**	***pre2***	***pre3***
Número de parámetros	73203	13555	2979
Número de filtros last-CNP	3	2	2
Número de canales de salida	100	44	12
Operación de pooling	Max	Avg	Avg
Función no lineal en last-CNP	TanH	PReLU	PReLU
Función no lineal en capa FC	Softmax	Softmax	Softmax

Tabla 5.13: Estructuras resultantes con sus respectivos elementos y sus correspondientes valores de los experimentos $pre2$ y $pre3$

El filtro de pooling cambió de Max a Avg, de manera que, ahora, valores negativos dentro de las representaciones son tomados en cuenta a diferencia de Max que solo toma el valor mayor de un conjunto de datos.

Para el caso de las neuronas de entrada en la capa FC, $pre2$ tiene 88 y $pre3$, 24. Ambas estructuras terminan con 3 neuronas de salida y una función de activación *Softmax*.

La Tabla 5.13 muestra el resumen de las estructuras de $pre2$ y $pre3$ tomando en cuenta los datos transformados con *Word2Vec*. El número de parámetros se calcula a partir de: (1) el número de filtros $|F|$, (2) las dimensiones de cada uno $v_i \times w \times z \times z'$, (3) número de neuronas de entrada en la capa FC x, (4) número de neuronas de salida en la capa FC y, (5) número de valores sesgo en los filtros convolucionales b_c y neuronas en la capa FC b_y. El número de sesgo en los filtros convolucionales, es igual al número de mapas de características z'. Así, para cada estructura, tomando en cuenta una

entrada de $60 \times 60 \times 1 \times 1$ de *Word2Vec*, el número de parámetros es igual a la Ecuación 5.1:

$$params = (\sum_{i=1}^{|F|} v_i wzz') + xy + b_c|F| + b_y \tag{5.1}$$

Y sustituyendo, la Ecuación 5.2 calcula el número de parámetros para $pre2$:

$$\begin{aligned} params &= (1 * 60 * 1 * 44 + 4 * 60 * 1 * 44) + 88 * 3 + 44 * 2 + 3 \\ &= (13200) + 264 + 88 + 3 \\ &= 13555 \end{aligned} \tag{5.2}$$

la Ecuación 5.3 calcula el número de parámetros para $pre3$:

$$\begin{aligned} params &= (1 * 60 * 1 * 12 + 3 * 60 * 1 * 12) + 24 * 3 + 12 * 2 + 3 \\ &= (2880) + 72 + 24 + 3 \\ &= 2979 \end{aligned} \tag{5.3}$$

Dicho número de parámetros obtenido por cada estructura, $pre2$ y $pre3$, representa el 18.5 % y 4 % de la RNC original, respectivamente. Esto quiere decir, que se logró reducir la complejidad de una RNC hasta en un 96 %.

6. Conclusiones y Trabajo futuro

En este documento se presentó el trabajo de AS con técnicas de NE frente a enfoques tradicionales del AA y AP. La propuesta de NE, *DeepNEWT*, busca mejoras en las estructuras de RNC, al mismo tiempo, que se van ajustando los pesos de los filtros convolucionales y los de la capa clasificadora (capa FC). También se llevó a cabo el entrenamiento de clasificadores tradicionales, MSV e IB, agregando una RNC que se extrajo de la revisión de la literatura.

En primera instancia, el algoritmo *DeepNEWT* fue utilizado para encontrar estructuras y pesos de RNC completamente fabricadas por las técnicas agregadas a este AG, que involucra variaciones aleatorias dentro de la cruza y las mutaciones. Dos conjuntos de experimentos fueron ejecutados para observar el comportamiento del algoritmo: (1) de manera preliminar se generaron RNC de menor tamaño, y (2) para mejorar la precisión en un conjunto reducido de datos. Todo esto utilizando el conjunto de tweets transformados con *Word2Vec*. En los experimentos preliminares, $pre2$ y $pre3$, se encontraron RNC que redujeron el tamaño de la RNC original hasta en un 96 %. Y, por otro lado, el siguiente conjunto de experimentos, referente a la mejora de la precisión, logró obtener varios resultados mayores al 50 % de precisión.

Después de ejecutado *DeepNEWT*, las RNC resultantes de $pre2$ y $pre3$ fueron extraídas y utilizadas para los experimentos es. Los clasificadores MSV, IB y el modelo de RNC fueron ejecutados en una CV de 10 $partes$ utlizando el conjunto de tweets transformado con *BERT*. La prueba *Friedman aligned ranks* indica la presencia de diferencias significativas con respecto a la precisión encontrada por los modelos $pre2$ y $pre3$. En varios de los resultados, las $partes$ de la CV fueron ganadas por los modelos generados por *DeepNEWT*. Para esta serie de experimentos se decidió ajustar la taza de aprendizaje a $0{,}001$ para ambas estructuras. Las pruebas estadísticas muestran que los experimentos es-$pre3$ tienen diferencias significativas frente a la RNC tomada de la literatura ajustada para *BERT* e IB. Y con respecto a la MSV, algunas de las estructuras de $pre3$, específicamente la $es3$ y

$es4$ muestran una mejora, ya que obtienen una mediana y media mejores, la cantidad de $partes$ ganadas es mayor, y la prueba estadística muestra un *p-value* de 1, en ambos casos, favorables para estas estructuras, dado que los $rangos$ son mayores ($32{,}15$ y $31{,}8$, respectivamente).

Finalmente, los parámetros utilizados para los experimentos $pre2$ y $pre3$ fueron asignados para la tarea de entrenar RNC mediante *DeepNEWT*, comparándose, en una etapa de prueba, contra los clasificadores tradicionales: IB, MSV, y la RNC extraída de la literatura. Estos experimentos se llevaron a cabo utilizando los tweets transformados con *Word2Vec*. La evaluación realizada fue hecha a través de una CV de 5 $partes$. En esta serie de experimentos, las pruebas estadísticas muestran una diferencia signficativa a favor del entrenamiento realizado por *DeepNEWT* sobre los clasificadores IB y RNC. Sin embargo, este no es el caso de la MSV, donde los $rangos$ la favorecen. Además, el análisis *post-hoc* genera un $p\text{-}value \geq \alpha$, lo cual demuestra que no existen diferencias significativas entre estos conjuntos de datos.

DeepNEWT fue utilizado para buscar una mejor estructura y mejores pesos para inicializarlos a priori antes de un entrenamiento exahustivo mediante el algoritmo de *retropropagación* (experimento es). Además, antes de hacer tal entrenamiento, *DeepNEWT* por sí solo logró reducir la complejidad de la estructura y aumentó la precisión frente a la RNC tomada de la literatura (experimento $pre2$ y $pre3$). Por lo anterior, y en conclusión, los objetivos fueron alcanzados en esta investigación. Asimismo, la hipótesis planteada en esta tesis es aceptada gracias a los resultados obtenidos durante los experimentos, estadísticos y pruebas de evaluación realizadas.

El trabajo futuro se divide en dos grupos: (1) mejoras sobre el algoritmo *DeepNEWT*, y (2) más experimentación.

Se proponen modificaciones a *DeepNEWT*: (1) mejorar la mutación de pesos W y sesgo b, (2) permitir que se combinen los elementos de un bloque CNP para crear bloques NPC o PNC, y (3) permitir la cruza del filtro convolucional. La primera mejora, se centraría en la actualización de W y b para tener el control de cuáles valores modificar si éstos han sido menos actualizados que el resto o si es

necesario hacer más actualizaciones sobre los mismos. Así, se espera, que en un futuro, debido a que *DeepNEWT* tiene acceso al interior de las RNC, pueda generar una explicación, de cierta manera, a través de este mecanismo. Por otro lado, la segunda y tercera propuesta de mejora, puede tener implicaciones en cuanto a la efectividad del algoritmo y la precisión.

Las siguientes propuestas de experimentación pueden tomar en cuenta las mejoras anteriores a *DeepNEWT*. (1) Realizar experimentación sobre otros conjuntos de parámetros como los realizados en la Sección 5.4.2 y analizar las RNC resultantes, con el fin de comparar los resultados y realizar pruebas estadísticas para mostrar una posible diferencia significativa. (2) Lo mismo que el punto anterior, pero utilizando los tweets transformados con *BERT*, de esta manera, es posible que se generen RNC distintas entre sí que pueden tener ventajas unas sobre otras. (3) Utilizar como función de aptitud el error generado por las RNC para evaluar los individuos. (4) Realizar experimentación como la de la Sección 5.4.4, agregando las mejoras al algoritmo expresadas con anterioridad. (5) Nuevamente, sobre los experimentos de la Sección 5.4.4, es posible tomar las RNC generadas durante el entrenamiento y entrenarlas, desde ese estado en el que se encuentran, para mejorar la clasificación. El *post*-entrenamiento se llevaría a cabo a través de *retropropagación*.

DeepNEWT al ser un algoritmo de NE, puede ser aplicado a otras tareas. Por ejemplo, en la VC donde se encuentran los trabajos más importantes de AP. Además, se da pie para que dentro del área de AS surjan más investigaciones de RNA, en vista de que la NE no es muy utilizada. Dentro del PNL, existen muchos modelos de AP, como los *transformers*, como es el caso de *BERT*, que aumentan el número de parámetros para obtener mejoras. La necesidad de ello tiene que ver con que las RNA aprenden mucho mejor con grandes cantidades de datos. Sin embargo, un algoritmo como *DeepNEWT* puede demostrar que se pueden reducir los parámetros y obtener resultados similares o mejores que los obtenidos por propuestas del Estado del Arte.

Bibliografía

Abdaoui, A., Pradel, C. & Sigel, G. (2020). Load what you need: Smaller versions of mutililingual BERT, En *Proceedings of SustaiNLP: Workshop on simple and efficient natural language processing*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.sustainlp-1.16

Abualigah, L., Alfar, H. E., Shehab, M. & Hussein, A. M. A. (2020). Sentiment Analysis in Healthcare: A Brief Review. *Studies in Computational Intelligence*, *874*(December 2019), 129-141. https://doi.org/10.1007/978-3-030-34614-0_7

Ain, Q. T., Ali, M., Riaz, A., Noureen, A., Kamran, M., Hayat, B. & Rehman, A. (2017). Sentiment Analysis Using Deep Learning Techniques: A Review. *International Journal of Advanced Computer Science and Applications*, *8*(6), 424-433. https://doi.org/10.14569/IJACSA.2017.080657

Akhtar, M. S., Gupta, D., Ekbal, A. & Bhattacharyya, P. (2017). Feature Selection and ensemble construction: A two-step method for aspect based sentiment analysis. *Knowledge-Based Systems*, *125*, 116-135. https://doi.org/10.1016/j.knosys.2017.03.020

Baccianella, S., Esuli, A. & Sebastiani, F. (2010). SENTIWORDNET 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. *Proceedings of the 7th International Conference on Language Resources and Evaluation, LREC 2010*, *0*, 2200-2204.

Baeck, T., Fogel, D. & Michalewicz, Z. (2018). *Evolutionary computation 1: Basic algorithms and operators*. Boca Raton, FL, CRC Press.

Baldi, P. (2012). Autoencoders, Unsupervised Learning, and Deep Architectures. *ICML Unsupervised and Transfer Learning*, 37-50. https://doi.org/10.1561/2200000006

Bishop, C. (2006). *Pattern recognition and machine learning*. New York, Springer. https://www.microsoft.com/en-us/research/publication/pattern-recognition-machine-learning/

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273-297. https://doi.org/10.1007/BF00994018

Dahou, A., Elaziz, M. A., Zhou, J. & Xiong, S. (2019). Arabic Sentiment Classification Using Convolutional Neural Network and Differential Evolution Algorithm. *Computational Intelligence and Neuroscience*, *2019*. https://doi.org/10.1155/2019/2537689

Desell, T. (2018). Accelerating the Evolution of Convolutional Neural Networks with Node-Level Mutations and Epigenetic Weight Initialization. *Proceedings of the*

Genetic and Evolutionary Computation Conference Companion, 157-158. https://doi.org/10.1145/3205651.3205792

Devlin, J., Chang, M. W., Lee, K. & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, *abs/1810.04805*(1950), arXiv 1810.04805, 4171-4186. http://arxiv.org/abs/1810.04805

Díaz-Galiano, M. C., García-Vega, M., Casasola, E., Chiruzzo, L., García-Cumbreras, M., Cámara, E. M., Moctezuma, D., Ráez, A. M., Cabezudo, M. A. S., Tellez, E., Graff, M. & Miranda, S. (2019). Overview of TASS 2019: One more further for the global Spanish sentiment analysis corpus. *CEUR Workshop Proceedings*, *2421*, 550-560.

Dufourq, E. & Bassett, B. A. (2017). EDEN: Evolutionary deep networks for efficient machine learning. *2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference, PRASA-RobMech 2017*, *2018-Janua*, 110-115. https://doi.org/10.1109/RoboMech.2017.8261132

Esuli, A., Sebastiani, F. & Moruzzi, V. G. (2006). SENTIWORDNET: A Publicly Available Lexical Resource for Opinion Mining. *Language*, *0*, 417-422.

Fogel, D. B. (2000). What is evolutionary computation? *IEEE Spectrum*, *37*(2), 26-32. https://doi.org/10.1109/6.819926

Galván, E. & Mooney, P. (2020). Neuroevolution in Deep Neural Networks: Current Trends and Future Challenges, arXiv, 1-20. http://arxiv.org/abs/2006.05415

Gauci, J. & Stanley, K. (2007). Generating large-scale neural networks through discovering geometric regularities. *Proceedings of GECCO 2007: Genetic and Evolutionary Computation Conference*, 997-1004. https://doi.org/10.1145/1276958.1277158

Godino, I. G. & D'Haro, L. F. (2019). GTH-UPM at TASS 2019: Sentiment analysis of tweets for Spanish variants. *CEUR Workshop Proceedings*, *2421*, 579-588.

González, J.-A., Hurtado, L.-F. & Pla, F. (2018). Elirf-upv at tass 2018: Sentiment analysis in tweitter based on deep learning. *CEUR Workshop Proc*, 37-44.

González, J. Á., Moncho, J. A., Hurtado, L. F. & Pla, F. (2020). Elirf-upv at tass 2020: Twilbert for sentiment analysis and emotion detection in Spanish tweets. *CEUR Workshop Proceedings*, *2664*, 179-186.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning*. Cambridge, Massachusetts, The MIT Press.

Graff, M., Miranda-Jiménez, S., Tellez, E. S. & Moctezuma, D. (2020). EvoMSA: A Multilingual Evolutionary Approach for Sentiment Analysis. *IEEE Computational Intelligence Magazine*, 76-88. https://doi.org/10.1109/MCI.2019.2954668

Hatcher, W. G. & Yu, W. (2018). A Survey of Deep Learning: Platforms, Applications and Emerging Research Trends. *IEEE Access*, *6*, 24411-24432. https://doi.org/10.1109/ACCESS.2018.2830661

Hernández-Hernández, J.-C., Mezura-Montes, E., Hoyos-Rivera, G.-J. & Rodríguez-López, O. (2021). Neuroevolution for Sentiment Analysis in Tweets

Written in Mexican Spanish (E. Roman-Rangel, Á. F. Kuri-Morales, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa & J. A. Olvera-López, Eds.). *Pattern Recognition MCPR 2021. Lecture Notes in Computer Science*, 101-110. https://doi.org/10.1007/978-3-030-77004-4_10

Herring, S. C. (1996). *Computer-Mediated Communication: Linguistic, social, and cross-cultural perspectives (Pragmatics & Beyond New Series)*. John Benjamins Publishing Company. https://doi.org/10.1075/pbns.39

Holland, J. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. MIT Press.

Hu, A. & Flaxman, S. (2018). Multimodal sentiment analysis to explore the structure of emotions. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 350-358. https://doi.org/10.1145/3219819.3219853

Hubel, D. H. & Wiesel, T. N. (1959). Receptive fields of single neurones in the cats striate cortex. *The Journal of Physiology*, *148*(3), 574-591. https://doi.org/10.1113/jphysiol.1959.sp006308

Khan, F. H., Qamar, U. & Bashir, S. (2016). eSAP: A decision support framework for enhanced sentiment analysis and polarity classification. *Information Sciences*, *367-368*, 862-873. https://doi.org/10.1016/j.ins.2016.07.028

Khan, S., Rahmani, H., Shah, S. A. A. & Bennamoun, M. (2018). *A Guide to Convolutional Neural Networks for Computer Vision* (Vol. 8). https://doi.org/10.2200/s00822ed1v01y201712cov015

Kim, Y. (2014). Convolutional neural networks for sentence classification. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 1746-1751. https://doi.org/10.3115/v1/d14-1181

Kochenderfer, M. (2019). *Algorithms for optimization*. Cambridge, Massachusetts, The MIT Press.

Kwak, H., Lee, C., Park, H. & Moon, S. (2010). What is twitter, a social network or a news media? *Proceedings of the 19th International Conference on World Wide Web*, 591-600. https://doi.org/10.1145/1772690.1772751

LeCun, Y., Bengio, Y. & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436-444. https://doi.org/10.1038/nature14539

Liu, B. (2012). *Sentiment Analysis and Opinion Mining* (Vol. 5). Morgan & Claypool publishers. https://doi.org/10.2200/S00416ED1V01Y201204HLT016

Martinez, A. D., Del Ser, J., Villar-Rodriguez, E., Osaba, E., Poyatos, J., Tabik, S., Molina, D. & Herrera, F. (2021). Lights and shadows in Evolutionary Deep Learning: Taxonomy, critical methodological analysis, cases of study, learned lessons, recommendations and challenges. *Information Fusion*, *67*, 161-194. https://doi.org/10.1016/j.inffus.2020.10.014

Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013). Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, 1-12.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. & Dean, J. (2013). Distributed representations ofwords and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 1-9.

Miranda-Jiménez, S., Graff, M., Tellez, E. S. & Moctezuma, D. (2018). INGEOTEC at SemEval 2017 Task 4: A B4MSA Ensemble based on Genetic Programming for Twitter Sentiment Analysis, 771-776. https://doi.org/10.18653/v1/s17-2130

Mozetič, I., Grčar, M. & Smailović, J. (2016). Multilingual Twitter Sentiment Classification: The Role of Human Annotators. *PloS one*, *11*(5), 1602.07563, e0155036. https://doi.org/10.1371/journal.pone.0155036

Murthy, D. (2013). *Twitter: Social Communication in the Twitter Age (Digital Media and Society)*. Polity.

Neethu, M. S. & Rajasree, R. (2013). Sentiment analysis in twitter using machine learning techniques. *2013 4th International Conference on Computing, Communications and Networking Technologies, ICCCNT 2013*, 1-5. https://doi.org/10.1109/ICCCNT.2013.6726818

Nina-Alcocer, V., González, J. Á., Hurtado, L. F. & Pla, F. (2019). Aggressiveness detection through deep learning approaches. *CEUR Workshop Proceedings*, *2421*, 544-549.

Ouyang, X., Zhou, P., Li, C. H. & Liu, L. (2015). Sentiment analysis using convolutional neural network. *Proceedings - 15th IEEE International Conference on Computer and Information Technology, CIT 2015, 14th IEEE International Conference on Ubiquitous Computing and Communications, IUCC 2015, 13th IEEE International Conference on Dependable, Autonomic and Se*, 2359-2364. https://doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.349

Pang, B., Lee, L. & Vaithyanathan, S. (2002). Thumbs up? Sentiment Classification using Machine Learning Techniques. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (July), 79-86. https://doi.org/10.3115/1118693.1118704

Paredes-Valverde, M. A., Colomo-Palacios, R., Salas-Zárate, M. D. P. & Valencia-García, R. (2017). Sentiment Analysis in Spanish for Improvement of Products and Services: A Deep Learning Approach. *Scientific Programming*, *2017*. https://doi.org/10.1155/2017/1329281

Pastorini, M., Pereira, M., Zeballos, N., Chiruzzo, L., Rosá, A. & Etcheverry, M. (2019). Retuyt-INCO at TASS 2019: Sentiment analysis in Spanish tweets. *CEUR Workshop Proceedings*, *2421*(task 1), 605-610.

Penãloza, V. (2020). Detecting aggressiveness in mexican Spanish tweets with lstm + gru and lstm + cnn architectures. *CEUR Workshop Proceedings*, *2664*, 280-286.

Pla, F. & Hurtado, L. F. (2013). Elirf-upv en tass 2013: Análisis de sentimientos en twitter. *Congreso de la Sociedad Española para el Procesamiento del Lenguaje Natural (SEPLN 2013)*, 220-227. https://documat.unirioja.es/servlet/articulo?codigo=7339856

Rangel, I. D., Sidorov, G. & Guerra, S. S. (2014). Creación y evaluación de un diccionario marcado con emociones y ponderado para el español. *Onomazein*, *29*(1), 31-46. https://doi.org/10.7764/onomazein.29.5

Read, J. (2005). Using emoticons to reduce dependency in machine learning techniques for sentiment classification. *ACL-05 - 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, (June), 43-48. https://doi.org/10.3115/1628960.1628969

Rebala, G., Ajay, R. & Churiwala, S. (2019). *Introduction to machine learning*. Springer, Cham. https://doi.org/10.4018/978-1-7998-0414-7.ch003

Risi, S. & Stanley, K. O. (2012). An enhanced hypercube-based encoding for evolving the placement, density, and connectivity of neurons. *Artificial Life*, *18*(4), 331-363. https://doi.org/10.1162/ARTL_a_00071

Rodríguez López, O. & de Jesús Hoyos Rivera, G. (2019). A Simple but Powerful Word Polarity Classification Model (L. Martínez-Villaseñor, I. Batyrshin & A. Marín-Hernández, Eds.). *Advances in Soft Computing*, 51-62. https://doi.org/10.1007/978-3-030-33749-0_5

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533-536. https://doi.org/10.1038/323533a0

Severyn, A. & Moschitti, A. (2015). UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification, (SemEval), 464-469. https://doi.org/10.18653/v1/s15-2079

Singh, N. K., Tomar, D. S. & Sangaiah, A. K. (2020). Sentiment analysis: a review and comparative analysis over social media. *Journal of Ambient Intelligence and Humanized Computing*, *11*(1), 97-117. https://doi.org/10.1007/s12652-018-0862-8

Sobkowicz, A. (2016). Automatic Sentiment Analysis in Polish Language (D. Ryżko, P. Gawrysiak, M. Kryszkiewicz & H. Rybiński, Eds.). En D. Ryżko, P. Gawrysiak, M. Kryszkiewicz & H. Rybiński (Eds.), *Machine intelligence and big data in industry*. Springer International Publishing. https://doi.org/10.1007/978-3-319-30315-4_1

Stanley, K. O. & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, *10*(2), 99-127. https://doi.org/10.1162/106365602320169811

Stanley, K. O., Clune, J., Lehman, J. & Miikkulainen, R. (2019). Designing neural networks through neuroevolution. *Nature Machine Intelligence*, *1*(1), 24-35. https://doi.org/10.1038/s42256-018-0006-z

Sun, S., Luo, C. & Chen, J. (2017). *A review of natural language processing techniques for opinion mining systems* (Vol. 36). Elsevier B.V. https://doi.org/10.1016/j.inffus.2016.10.004

Sun, Y., Xue, B., Zhang, M. & Yen, G. G. (2020). Evolving Deep Convolutional Neural Networks for Image Classification. *IEEE Transactions on Evolutionary Computation*, *24*(2), 394-407. https://doi.org/10.1109/TEVC.2019.2916183

Vargas-Hákim, G. A., Mezura-Montes, E. & Acosta-Mesa, H.-G. (2021). A Review on Convolutional Neural Networks Encodings for Neuroevolution. *IEEE Transactions on Evolutionary Computation*. https://doi.org/10.1109/TEVC.2021.3088631

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *2017-Decem*(Nips), 5999-6009.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., … Dean, J. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, arXiv, 1-23. http://arxiv.org/abs/1609.08144

Xie, L. & Yuille, A. (2017). Genetic cnn. *2017 IEEE International Conference on Computer Vision (ICCV)*, 1388-1397. https://doi.org/10.1109/ICCV.2017.154

Yadav, A. & Vishwakarma, D. K. (2020). Sentiment analysis using deep learning architectures: a review. *Artificial Intelligence Review*, *53*(6), 4335-4385. https://doi.org/10.1007/s10462-019-09794-5

Young, T., Hazarika, D., Poria, S. & Cambria, E. (2018). Recent trends in deep learning based natural language processing [Review Article]. *IEEE Computational Intelligence Magazine*, *13*(3), 55-75. https://doi.org/10.1109/MCI.2018.2840738

Zhang, L., Wang, S. & Liu, B. (2018). Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *8*(4), 1-25. https://doi.org/10.1002/widm.1253

...