Un Algoritmo Genético para la Selección de Características de las Representaciones Vectoriales de Texto dadas por BETO

Maestría en Inteligencia Artificial

Juan José Guzmán Landa
S21000453

Instituto de Investigaciones en Inteligencia Artificial
Universidad Veracruzana

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

31.01.2024

Agradecimientos

Nadie está preparado para la vida, una llena de eventos caóticos, sucesos donde muchas veces ni si quiera están sujetos a tus decisiones, y aun a pesar de ello, existe una constante, una con la capacidad de estar en todos los eventos, fuerte para cualquier circunstancia, esto es algo que todos conocemos como familia. Aquella donde comienzas tu trayectoria en la vida, pero también es esa que se forma durante el mismo andar, muchos se vuelven parte de la constante, algunos pasarán a ser solo algo transitorio, pero siempre hay mucho que aprender de todos, por esta misma razón quiero agradecerles.

Principalmente agradezco a mi Madre, ella es mi fuente más grande de impulso, es la persona que me ha enseñado a levantarme siempre sin importar cuantas veces la vida me azote contra el suelo. También es la primera en escuchar y dar el mejor consejo, es alguien que no conoce límites, siempre da más de sí tanto al actuar como al dar amor. Gracias por tanto a cambio de tan poco querida Madre.

A mi mentor, el Dr. Guillermo de Jesús Hoyos Rivera, es la persona con más energía que he conocido, siempre listo para todo con la mejor actitud. No existe ni la más mínima duda, sin usted nada de esto se hubiera logrado, y no tengo las palabras para expresar la magnitud de mi agradecimiento. Al final, solo puedo esperar haber cumplido con todas sus expectativas.

Al Dr. Efrén Mezura Montes, le agradezco su disposición siempre ante cualquier duda, sobre todo por sembrar en mi la curiosidad hacía el computo evolutivo, sin sus clases no hubiera logrado

entender tanto, es un gran maestro y una gran persona.

Al Mtro. José Clemente Hernández Hernández, la persona quien me mostró todo lo que podía hacer, sin sus ideas este trabajo no hubiera sido el mismo, gracias por resolver todas mis dudas sin importar la hora, pero en especial por todas esas preguntas que siempre me hicieron dudar.

A mi Padre por enseñarme el valor de la constancia y disciplina, siempre dispuesto a reparar todo hasta lo que no conoce, además de tener una habilidad y tenacidad sorprendentes a la hora de conducir, así como también su gran habilidad para contar buenos chistes. También agradezco a mi Hermana por alegrar mis días con todas sus ocurrencias, siempre me ha hecho saber lo orgullosa que está de mí y espero poder seguir haciéndolo.

A mi mejor amiga y compañera de mis travesías, gracias por escuchar una y otra vez todo lo que alberga mi mente, por siempre estar dispuesta a despegar y volar alto aun si la caída pueda ser dura. Nunca dejes de contagiar tu agradable risada.

A la persona que mas ha entrado y salido de mi constante, gracias hermano, fuiste parte fundamental de mis resultados. Al grupo de trabajo Syntagma, a mis compañeros de maestría, amigos y más que amigos, a todos les agradezco cada momento compartido, sin esos pequeños instantes no sería ni una fracción de lo que soy hoy.

También agradezco al Consejo Nacional de Humanidades, Ciencias y Tecnologías, CONACyT, por su apoyo a través de una beca, fue pieza indispensable para la realización de estos estudios de maestría en la Universidad Veracruzana.

Resumen

El tratamiento de datos alfanuméricos en computadoras no es natural para éstas, pues la forma en que representan todos los datos es a través de valores numéricos. En consecuencia, al trabajar con procesamiento de material en lenguaje humano (lenguaje natural), es necesario crear representaciones numéricas equivalentes para que éstas sean procesadas por las computadoras.
En la actualidad existen diversas técnicas para representar el texto de manera numérica, siendo una de las de mayor éxito la conocida como modelo de Representación de Codificador Bidireccional de Transformadores, BERT, por sus siglas en inglés (Devlin *et al*., 2019). La clave del éxito en esta representación viene dada porque los tokens del texto tienen un conjunto de características diferente dependiendo de su contexto, lo que permite tener información más detallada a la hora de abordar diversas tareas dentro del Procesamiento del Lenguaje Natural, NLP, por sus siglas en inglés. La selección de características dentro de esta nueva representación es un enfoque importante para conocer si todas las características son necesariamente relevantes o pueden existir algunas que destaquen por su importancia de las demás, y por ende filtrar las que no. Existen diversos métodos de selección de características, por filtro, embebido, y de envoltura, mejor conocidos en el inglés como *Filter*, *Wrapper* y *Embedded*. Dada la naturaleza de este trabajo el método de *Wrapper* es el más adecuado para buscar el rendimiento de diferentes subconjuntos de características dado un clasificador. Por otro lado, tomando en cuenta el espacio del problema que se genera con la representación actual, el enfoque de envoltura en la literatura es habitual encontrarlo con el uso de un algoritmo genético para la selección de características. De esta manera tenemos un número

finito de individuos que ponen a prueba su rendimiento en el clasificador con las diferentes representaciones generadas. El propósito de este trabajo es hacer la búsqueda bajo un entorno nativo del español, y encontrar diferentes representaciones que puedan tener un rendimiento competitivo o incluso mejor a la representación actual para una tarea en específico. Todo con el fin de dar pauta a seguir haciendo una búsqueda más profunda de las características relevantes, además de proporcionar las nuevas representaciones para que sean utilizadas en otros modelos con tareas diferentes a las presentadas en este trabajo. En los experimentos y resultados se podrá encontrar el comportamiento que tienen las diferentes representaciones, los rangos en los que se van comportando mejor y si existen características que prevalecen por destacar entre todas las representaciones generadas. Por último, se debe de destacar que en todos los casos hubo una reducción en el número de características original de 768, el GA logró encontrar representaciones con un buen rendimiento y con un número de características de 212, 251, 300 y 578. Por lo tanto, tenemos representaciones más pequeñas que a su vez también reducen el número de parámetros necesarios para entrenar un modelo, un beneficio que puede ser muy relevante hablando en términos de tiempo y costo computacional.

Índice general

Índice de Figuras

Índice de Tablas

Lista de Abreviaciones

AI	Artificial Intelligence
ANN	Artificial Neural Network
BERT	Bidirectional Encoder Representations from Transformer
BoW	Bag Of Words
CNN	Convolutional Neural Network
DL	Deep Learning
FS	Feature Selection
GA	Genetic Algorithm
IG	Information Gain
ML	Machine Learning
NB	Naive Bayes
ReLU	Rectified Linear Units
SVM	Support Vector Machine
TF-IDF	Term Frequency - Inverse Document Frequency

1. Introducción

La representación numérica del texto es parte fundamental de modelos que tienen la capacidad de realizar tareas como la clasificación de la polaridad, traducción automática, generación automática de texto, transformación de texto a imagen, entre otros. La forma más conocida actualmente es la representación del texto a través de vectores numéricos, es decir, por cada una de las palabras o fragmentos de palabras, mejor conocidos como tokens, deberá de existir un vector de 768 valores de punto flotante. La representación vectorial viene proporcionada por un modelo del lenguaje llamado BERT, el cual fue entrenado con la base de datos de Wikipedia (en inglés) y BookCorpus, por lo tanto, es un modelo que está diseñado para representar texto en inglés. En el caso de la representación de texto para el español existe otro modelo del lenguaje nombrado BETO (Cañete *et al.*, 2020). Tanto BERT como BETO tienen un vocabulario de 31,002 tokens, y cada uno de los elementos que conforman su vocabulario tienen asociado un vector de 768 valores de números reales. Se debe de resaltar que en todo el trabajo se puede hablar de palabras pero también de tokens (palabras o fragmentos de palabras), haciendo referencia a la forma en que distintos modelos procesaron el texto.

La forma más habitual actualmente para conocer el rendimiento de estas representaciones consiste en utilizar Redes Neuronales Artificiales, ANN, así como de Redes Neuronales Convolucionales, CNN, en ambos casos por sus siglas en inglés, entre otras. Estos modelos tienen la capacidad de abordar diferentes tipos de tareas, por ejemplo, la clasificación de texto, algo que se puede extender a una amplia cantidad de dominios y clases. Si tomamos el caso con respecto a la clasificación de una sentencia dada su polaridad, se ha logrado alcanzar un buen rendimiento con respecto a la exactitud (o *accuracy* por su traducción más conocida en el inglés). Gran parte del éxito en la tarea anterior, consiste en la nueva representación del texto que se obtiene a través del modelo, y aquí es donde la tarea puede escalar fácilmente a niveles vertiginosos, dado que la representación equivale a 768 características por cada uno de los

tokens (unidades sintácticas) que conforman la sentencia. Por ejemplo, para una sentencia de tan solo 50 tokens, tenemos una entrada de 50 x 768 = 38,400 características, es decir que el modelo debe de tener una entrada y arquitectura con respecto a todas esas características, y esa misma situación se cumple con todas las demás sentencias. También es posible obtener la representación de un solo vector por toda la sentencia, pero existen limitaciones a la hora de hacer la selección de esas características, algo que se verá más a detalle en secciones posteriores.

Hacer una selección de características, mejor conocido en el inglés como Feature Selection (FS), tiene diversos métodos y enfoques, pero su propósito general es encontrar, convertir, o extraer de todo un conjunto de características aquellas donde la información sea más relevante. En el enfoque de envoltura se menciona para la clasificación de texto un uso inadecuado por su alto número de características (Deng *et al*., 2019). Por lo regular se refieren a esto cuando se trata de clasificar un texto utilizando métodos de representación de texto como una bolsa de palabras, BoW, y un clasificador como la Máquina de Soporte Vectorial, SVM, o un Ingenuo Bayesiano, NB, entre otros, dado que la representación debe de generar un vector para cada uno de los textos, el cual toma en cuenta la frecuencia de las palabras. Las características resultantes de cada vector solo expresan un número para cada una de las palabras donde se puede ver la cantidad de existencias. Dado lo anterior, realmente no se está trabajando con una representación con la capacidad de almacenar información más especializada para el texto, por lo tanto, en el caso de BoW la selección de características se debería realizar en cada nueva tarea asignada y solo sería útil para ese momento. En este trabajo se está realizando una selección de características para buscar si existe una **nueva representación** con el poder de mejorar o igualar el rendimiento en comparación con la representación original, pero con menor costo computacional.

El espacio de búsqueda dado el problema, se vuelve lo suficientemente grande como para involucrar un Algoritmo

Genético, GA, por sus siglas en inglés, además la representación binaria de cada individuo se adapta apropiadamente para la búsqueda de las características, por lo tanto, la selección de características con un enfoque de envoltura es el que se utilizará en este trabajo.

1.1. Antecedentes

Una FS para la clasificación de textos, por lo regular es mediante el enfoque de filtro, por su simplicidad y poca complejidad de implementación. Es decir, en este enfoque se trabaja directamente con métodos donde se asigna algún puntaje a las palabras, y partir de ahí hacer la selección. En otros casos el enfoque no está necesariamente en la clasificación de texto, pero sí abordan una selección de características mediante otras técnicas, incluso mencionan trabajos donde utilizan el método de envoltura por sí solo, y combinado con el de filtro.

En todos los casos para una FS es necesario tomar en cuenta la representación de los datos, en el caso del texto, hay una gran variedad de representaciones, cada una con diferentes aportes, y solo pocas con la capacidad de ser adaptables a diferentes dominios. Considerando una FS en trabajos con una representación más cercana a la del presente, es decir, la representación dada por BERT, se encuentran trabajos donde incluso llegan a mencionar a BERT como un tipo de selector de características (Garcia-Silva & Gomez-Perez, 2021), esto por su mecanismo de *auto-atención*, el cual a grandes rasgos ajusta los pesos de su perceptrón para cada token de entrada, orientando esa atención en los tokens con mayor aproximación a un dominio en específico. Por otro lado, bajo la misma representación vectorial por palabra, los trabajos hacen una FS pero no necesariamente de una manera directa sobre las propias características, al menos no en BERT, a diferencia de otro donde sí trabajan con las características pero su representación es con el modelo *Word2Vec*.

En este punto los trabajos solo se centran en el enfoque de filtro o en la representación de texto, además los resultados de la búsqueda se van limitando conforme se hace más precisa la consulta, esto es, considerar el enfoque de envoltura con un GA para una FS en la clasificación de texto. A pesar de ello sí hay trabajos con una cierta aproximación, donde implementan una FS usando un GA para la clasificación de texto, pero la representación vuelve a ser como la de *BoW*. Ahora, agregando el idioma principal de este trabajo de investigación, el español mexicano, los trabajos resultantes tienen su base en el modelo BETO desde el aspecto del ajuste fino, mejor conocido en el inglés como *fine-tuning*, pero ya no comparten la línea de investigación en los demás objetivos, como en la apicación de una FS. Todo lo descrito hasta el momento, se profundiza a más detalle en la Sección 3.

En este trabajo de investigación la selección de características bajo el enfoque de envoltura utilizando un GA tiene el objetivo de generar una nueva representación con características más relevantes para la tarea propuesta, pero a la vez que estas nuevas representaciones puedan ser utilizadas en otros conjuntos de datos para saber si el rendimiento se mantiene, de esta manera también se puede justificar el hecho de la existencia de características con mayor importancia, así como de otras no tan relevantes.

1.2. Propuesta

La idea general de este trabajo es diseñar e implementar una estrategia de selección de características bajo el enfoque de envoltura, es decir, utilizar un GA para la búsqueda del mejor subconjunto de características, determinando este punto a través de la aptitud obtenida de los diferentes individuos en la tarea de clasificación del texto. La propuesta también tiene como propósito plantear todo este enfoque desde el modelo especializado para el español, BETO, y a su vez utilizar conjuntos de datos nativos del mismo idioma, algo ausente hasta ahora en el estado del arte. En

especial se plantea trabajar con un conjunto de datos en específico y aplicar diferentes filtros con base en las sentencias más relevantes para así hacer un tipo de aprendizaje de pocos intentos, mejor conocido en el ingles como *few-shot learning*. La propia naturaleza del GA nos permitirá observar cómo se pueden comportar las diferentes representaciones generadas, para poder comprobar si la famosa representación original realmente es la óptima en todos los diferentes conjuntos de datos restantes. Así mismo tanto las nuevas representaciones como la original tendrán una comparación indirecta contra diferentes técnicas propias de cada conjunto de datos.

1.3. Hipótesis

Es posible generar nuevas representaciones con una longitud de características dependiente del número de sentencias utilizadas, las cuales se espera que tengan un rendimiento al menos similar, probablemente mejor al de la original. Además, se espera que la calidad de la clasificación de las nuevas características sea mejor en diferentes conjuntos de datos, con el beneficio adicional de un menor costo computacional. Todo lo anterior tomando como base el modelo del lenguaje BETO.

1.4. Justificación y Motivación

Una de las razones de este trabajo se encuentra en la ausencia de trabajos relacionados a la selección de características con el enfoque de envoltura para una representación dada por BETO. Así como la falta de información con respecto a la representación de 768 características el cual indique algún significado de relevancia entre las diferentes características. Sabemos de forma general como cada vector tiene capturada información según el contexto entre todas sus características, pero no conocemos cuales pueden ser más relevantes, y tampoco hay información al respecto, por eso mismo se vuelve importante la selección de características.

La última motivación de este trabajo recae en el poder de cómputo necesario actualmente para re-entrenar modelos como BERT o BETO,

si se desea trabajar con un modelo como BERT es necesario re-entrenar todos y cada unos de sus parámetros para adaptar la representación a una nueva tarea. No hay una aproximación para saber que tan necesarias son todas esas características, sin embargo, si existe una mejor representación el poder de cómputo necesario se puede ver reducido, permitiendo la implementación de modelos del lenguaje con dimensiones más eficientes. Esto último mencionado es un tema de suma importancia debido a la consecuencia, por un lado, se puede aplicar lo anterior a su contraparte BERT y en general modelos que tengan una representación vector-palabra, por otro lado, también hablamos directamente de una reducción en la huella de carbono generada por todos estos modelos de DL, algo que tiene un gran impacto según los descrito en (Anthony *et al.*, 2020).

1.5. Objetivos

1.5.1. Objetivo General

Hacer una selección de características con el enfoque de envoltura a través de un algoritmo genético para encontrar diversas representaciones con un conjunto de datos específico, comparar el rendimiento de las nuevas representaciones contra la representación original en los demás conjuntos de datos nativos del español, así como contra otras técnicas.

1.5.2. Objetivos Específicos

1. Analizar el proceso de implementación que puede tomar el algoritmo genético para realizar la selección de características. Tomando en cuenta para la primera instancia una prueba de concepto.
2. Hacer el estudio pertinente de cada uno de los conjuntos de datos nativos del español, para trabajar con el más representativo.
3. Aplicar *few-shot learning* con diferentes subconjuntos de los datos más relevantes para obtener diferentes representaciones.

4. Dada todas las nuevas representaciones, evaluar si existe el incremento o disminución del número de características con respecto al original.
5. Comprobar el rendimiento de todas las representaciones incluyendo la original, en los diferentes dominios de cada conjunto de datos, así como una comparación indirecta contra otras técnicas.

2. Marco Teórico

2.1. Clasificación de la polaridad de un texto

La tarea más conocida a la hora de trabajar con el texto es la clasificación del mismo entre diferentes tipos, algo determinado a partir de la información contenida en el texto. En tiempos más recientes se ha vuelto famoso el análisis de sentimientos, en esencia sigue siendo una clasificación de texto, pero ahora en sentido de la polaridad, es decir, se busca poder clasificar un conjunto de textos a partir de la carga positiva o negativa reflejada en cada uno.

En el trabajo de (Dalal & Zaveri, 2011) se exponen cuales son los pasos fundamentales para llevar a cabo una clasificación de texto adecuada. El primer paso es procesar el texto, esto incluye tanto la limpieza, como la extracción de características, o también, la transformación del texto a una representación dada a partir de alguna técnica. Para la limpieza del texto no existe un seguimiento de pasos estándar, en algunos trabajos tenemos ciertas formas de limpieza, dado a como algunos modelos pueden considerar información valiosa lo que algunos otros no. De la misma forma no existe una representación estándar para atender todas las tareas de clasificación, de hecho, existe una interesante evolución con respecto a las diferentes representaciones y como ha recibido el texto a lo largo del tiempo. Todo apunta a mejorar el rendimiento con cada nueva representación, no solo para la tarea de clasificación, si no también, para tareas de otra índole como la generación de texto a partir de una entrada, la traducción automática, entre otras.

En el caso de la lengua inglesa, no existe la acentuación, por lo tanto, una limpieza de ello no se considera necesaria, pero para el caso del español en muchas situaciones sí es necesario eliminar los acentos de todas las palabras, evitando de esta forma que exista la misma palabra más de una vez por el mal uso del acento. Así mismo también se le llama limpieza al proceso de convertir todo el texto a

minúsculas, eliminar las palabras vacías (o mejor conocido por su traducción al inglés como *stopwords*), hacer una lematización de las palabras (o mejor conocido por su traducción al inglés como *stemming*), incluso por la forma en la como ahora se puede enriquecer el texto a partir de emojis, también se puede hacer una limpieza de ellos.

Las palabras vacías se determinan como aquellas palabras que no aportan información relevante al texto, muchas de ellas ya están previamente cargadas en algún diccionario y solo basta con recuperarlas para hacer la limpieza correspondiente. Por ejemplo, en la siguiente lista se muestran algunas palabras vacías conocidas comúnmente para el idioma español, extraídas desde Python, de una biblioteca nombrada *nltk* (Natural language toolkit) y la más completa *spacy*:

- A, acá, ahí, el, la, con, como, de, del, en, es, ni, me, por, sin, tan, y, ya, yo, tu, esas, eran, eras, ese, eso, esta, ex, fin, fue, ha, he, mi, o, sido, te, tengas, sentido, teniendo, habrás, estuvierais, tus, tienen, vuestras, habiendo, muchos, para, algunas, desde, serían, ti, sois, estaraís, habré, porque, cuando, tenía, tengamos, estados, contra, estad, estada, estuvierais, habida, habidas, haya, hayan, tuvo, tuya, un, éramos, cuantas, realizó, manifestó, ciertos, señaló, tercero, siguiente, total, alrededor, verdadera, informó, agregó, expresó, hago, dan, dicho

Las palabras vacías son propias de cada lenguaje, por eso será necesario evaluar para cada idioma cuáles son sus correspondientes stopwords. Para el caso de la lematización de palabras se puede ver más como el proceso donde una palabra conjugada vuelve a su forma más básica, por ejemplo, podemos tener las palabras duermo, duermes, dormiré, dormimos, duermen, dormía, dormiremos, dormís, dormirán, y otras más, pero al final todas provienen del mismo verbo, dormir, entonces, ese es el propósito de la lematización, convertir todas esas conjugaciones a su verbo base para evitar que existan diferentes características a partir de una misma palabra.

Puede existir un paso intermedio entre la extracción de características o representación del texto y el clasificador, algo mencionado por los autores en el estudio de (Kowsari *et al*., 2019). Dicho paso consiste en una selección o reducción de características con el fin de reducir el tiempo de procesamiento, y por ende, el costo computacional; además los autores también mencionan cómo es mejor realizar dicha reducción en lugar de solo buscar clasificadores baratos. El último paso precisamente entra en materia de los clasificadores, dado que existe una gran variedad y entre ellos también existe una gran diferencia de como procesan los datos, es importante tomar en cuenta la representación del texto para determinar como el clasificador podría manejar mejor las características, y obtener un mejor rendimiento.

Considerando la relevancia de las técnicas de aprendizaje profundo, y entrando más a detalle en los distintos clasificadores que existen, el trabajo de (Minaee *et al*., 2021) describe varias de ellas en el ámbito de la clasificación de texto, comienza desde los modelos más simples como una red neuronal recurrente o una CNN, hasta modelos más complejos como aquellos donde involucran un transformer o modelos híbridos.

2.2. Representación del texto

Las máquinas están diseñadas para operar números, incluso el texto escrito por uno en las máquinas está siendo procesado a través de una codificación numérica. Pero si deseamos procesar el texto de una manera más especializada y crear modelos capaces de trabajar directamente con él, lo primero forzosamente es obtener una nueva representación. Es decir, convertir el texto de su representación actual, a una que tenga la capacidad de ser numérica, donde todos esos nuevos valores numéricos deben de tener el propósito de contener la información necesaria del texto para resolver alguna tarea asignada. Las representaciones generadas para procesar el texto a lo largo del tiempo han sido varias, muchas con el fin de cumplir una tarea en específico, y las más actuales han tenido el objetivo de

contener información más general con la capacidad de resolver tareas en diferentes áreas y dominios.

2.2.1. Bolsa de palabras

En el libro de (Liu *et al.*, 2020) se realizó una extensa investigación acerca de las representaciones dadas al texto para tener distinta información encapsulada. Empezando con una de las representaciones ampliamente utilizada en diversos trabajos tenemos a BoW, un método que busca contabilizar la frecuencia de las palabras en un documento de texto. El primer paso para obtener nuestra bolsa de palabras es sacar el vocabulario a partir de todos los textos (también se le llama documentos), después cada texto tendrá asociado un vector del mismo tamaño como del vocabulario, por último, los valores dentro de cada vector serán la frecuencia de cada palabra dentro de cada texto, por ejemplo, si tenemos lo siguientes textos (sacados del conjunto de datos generado por Keung *et al.*, 2020) (sic):

1. "es fantastico y en amazon lo encontre al mejor precio,me encanta como te deja las ojeras y todo en general"
2. "Por el precio, este reloj es inmejorable. Estilo clásico y mantiene el tiempo bien. Estoy muy satisfecho. . . ."
3. "Estoy encantada va muy bien tanto manicura como pedicura nuen precio buena calidad"
4. "Estoy super contento con la pizarra. Ademas tiene una calidad muy buena. Recomendable."
5. "el producto es como se describe. Ahora lo importante es que la duración sea la indicada, pero ... no suele ser."

En algunos casos al usar esta técnica, se pueden eliminar las stopwords, dado que la tarea más común es la clasificación del texto, se busca tener las palabras más relevantes de cada dominio, y las palabras vacías solo agregan ruido en general a todos los textos en todos los dominios. Por lo tanto el vocabulario del texto sería el siguiente:

- además, ahora, amazon, bien, buena, calidad, clásico, contento, deja, describe, duración, encanta, encantada, encontré, estilo, fantástico, general, importante, indicada, inmejorable, manicura, mantiene, mejor, nuen, ojeras, pedicura, pizarra, precio, producto, recomendable, reloj, satisfecho, ser, suele, super, tiempo, va.

Para el ejemplo, las palabras vacías se consideraron a partir de la biblioteca, nltk, propia del lenguaje de programación Python. El resultado fue un vocabulario 37 palabras, así que los vectores de cada uno de los textos en el ejemplo también serán del mismo tamaño. Para generar los vectores debemos de considerar las palabras en el orden como está el vocabulario, es decir, la posición 1 del vector es la palabra "además", la posición 2 es la palabra "ahora", la posición 3 es "amazon", la 4 es "bien", la 5 es "buena" y así sucesivamente. De esta manera y con ese orden se debe de ir haciendo el conteo de palabras para cada texto, por lo tanto, los vectores resultantes serían los siguientes (La tabla solo muestra las primeras 20 palabras del vocabulario):

texto 1	0	0	1	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	0
texto 2	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1
texto 3	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
texto 4	1	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
texto 5	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0

Dado que solo fue un ejemplo de pocos textos con pocas palabras, las ocurrencias de las palabras no son mayor a 1, de hecho, este BoW se puede ver como una de las alternativas donde solo reflejan la existencia y/o ausencia de una palabra independientemente de sus ocurrencias. Analizando lo expuesto a una escala más grande, existen textos con una mayor longitud de palabras, el resultado serían vectores más grandes, con un mayor número de ocurrencias por cada palabra.

Esta representación difiere en muchos sentidos con la de BETO, por un lado, tenemos información la cual solo refleja existencia u ocurrencias, además las stopwords se vuelven completamente irrelevantes, algo que no sucede en la representación dada por BETO.

También existe diferencia en la dependencia del tamaño de los vectores con respecto a la longitud del vocabulario, pero sobre todo, BoW no tiene un orden en las palabras, algo por el cual hace perder información valiosa propia de un texto, es decir, cualquier texto es una secuencia de palabras, la cual proporciona un orden y sentido al texto, algo donde BoW se queda fuera. Por último, esta técnica debe de ser aplicada nuevamente para cada conjunto y tarea a resolver, los vectores están ligados completamente al conjunto de datos procesado en cada momento.

2.2.2. N-gramas

Considerando el orden cronológico propuesto en (Liu *et al.*, 2021), el primer modelo de representación de texto fue el de N-gram, el cual nace con el propósito de capturar la información secuencial inherente del texto, es decir, la naturaleza de cualquier sentencia está en contener un palabra seguida de otra, así sucesivamente hasta llegar al número de palabras necesario para dar sentido a la sentencia como tal. A manera de ver un ejemplo del funcionamiento de *N-gram* en su aplicación *bi-grama*, *tri-grama* y *cuatri-grama*, tenemos la siguiente oración, "El producto que pedí me llegó en muy mal estado", por lo tanto, los n-grams quedarían así:

bi-gramas	**tri-gramas**	**cuatri-gramas**
El producto	El producto que	El producto que pedí
Producto que	Producto que pedí	Producto que pedí me
Que pedí	Que pedí me	Que pedí me llegó
Pedí me	Pedí me llegó	Pedí me llego en
Me llegó	Me llegó en	Me llegó en muy
Llegó en	Llegó en muy	Llegó en muy mal
En muy	En muy mal	En muy mal estado
Muy mal	Muy mal estado	-
Mal estado	-	-

Es importante mencionar como cada *n-gram* se vuelve un dato atómico, esto es, considerar como una sola unidad cada *n-gram* a pesar de estar conformados por más de una palabra. Una vez se haya generado este mismo proceso para todo un conjunto de sentencias de diferentes dominios, podemos hacer exactamente lo mismo como en *BoW*, un conteo de frecuencias de nuestros *n-grams*, esto con el fin de ver cuál secuencia de palabras tiene una mayor presencia en los

diferentes textos. Entre los beneficios de esta técnica, también se puede notar algo importante con la nueva información, brindar la posibilidad de calcular la probabilidad de una palabra a partir de las palabras previas que son parte de los *n-grams*, incluso se puede aplicar el caso inverso, de toda una secuencia de palabras, cuál palabra va después, algo que se puede ver en esencia dentro del modelo *Word2Vec*.

Algunas otras consideraciones de la técnica anterior es la forma en la cual incrementa el tamaño de su vocabulario, dado que primero se debe generar un vocabulario inicial con palabras únicas, para después, al ir agregando los diferentes *n-grams*, cada uno va generando demasiada información y es bastante habitual encontrar ruido por las ocurrencias de 1 en la mayoría de los n-grams generados. Por último, las técnicas anteriores como *BoW* y *N-grams*, tienen un enfoque demasiado apegado a las ocurrencias de elementos, por lo tanto, se vuelven modelos probabilísticos. Además, solo funcionan para representar un texto completo en un solo vector o varios vectores, pero no se está logrando una representación propia para cada una de las palabras.

Hasta este punto la forma de capturar la información valiosa del texto a través de las técnicas previamente descritas, ha demostrado tener un buen rendimiento, son sencillas, prácticas e incluso se complementan, pero a pesar de ello, aun queda mucho por mejorar. Es por eso que surge una nueva técnica, más bien, un modelo con la capacidad de capturar información a un nivel significativamente superior, e incluso da inicio a una nueva forma de representar al texto por palabras.

2.2.3. Word2Vec

El modelo fue presentado en el 2013, con el artículo de (Mikolov *et al.*, 2013), su propósito fue generar vectores de números reales para un conjunto amplio de palabras. La información capturada por cada

uno de los vectores hizo que las palabras similares gramaticalmente estuvieran cerca y lo contrario para palabras distintas, todo esto dentro de un espacio de 300 dimensiones. De manera general *Word2Vec* está conformado por 2 modelos, el primero su función consiste en predecir una palabra a partir de las palabras a su alrededor, es decir, Bolsa continua de palabras, *CBoW* por sus siglas en inglés, y el segundo tiene como objetivo hacer lo contrario, a partir de una palabra generar las palabras que deberían de estar en su contexto, es decir, *Skip-Gram*.

Primero para ambos modelos necesitamos darle una representación vectorial inicial a cada palabra del vocabulario, y para ello se puede utilizar una técnica conocida como codificación one-hot, la cual consiste en asignar a cada palabra un vector de la longitud del vocabulario con todos su valores a 0 menos la posición de la palabra dentro del vocabulario, ese será el único valor en 1 dentro de su vector one-hot, por ejemplo, para la sentencia "El lobo alfa siempre vigila la manada desde atrás", considerando su vocabulario, su representación *one-hot* sería la siguiente:

El	1	0	0	0	0	0	0	0	0
Lobo	0	1	0	0	0	0	0	0	0
Alfa	0	0	1	0	0	0	0	0	0
Siempre	0	0	0	1	0	0	0	0	0
Vigila	0	0	0	0	1	0	0	0	0
La	0	0	0	0	0	1	0	0	0
Manada	0	0	0	0	0	0	1	0	0
desde	0	0	0	0	0	0	0	1	0
Atrás	0	0	0	0	0	0	0	0	1

Es una forma bastante simple de generar una representación numérica del texto a través de vectores por cada palabra, aunque la información aportada con esa representación solo está capturando su posición dentro del vocabulario, algo con poca utilidad, además cuando el vocabulario es demasiado grande, la representación de una sola palabra tendrá un vector de la misma longitud. Otra representación también se puede dar a partir de una generación de valores aleatorios, según sea el interés y propósito de cada autor, después se determina una longitud fija para todos los vectores de cada palabra y la aleatoriedad de los valores de cada vector podrían

estar dentro de un rango establecido, por ejemplo, entre 0 y 1.

Asumiendo que las dos técnicas de *Word2Vec* están basadas en un modelo ampliamente utilizado dentro del NLP, nombrado Modelo del Lenguaje de Redes Neuronales, por sus siglas en ingles NNLM, la representación inicial de las palabras es a través de la codificación *one-hot*, seguido hay una proyección para reducir la dimensionalidad, por último, una red neuronal tipo *feed-forward* con el fin de resolver alguna tarea en particular. Con la finalidad de comprender más a detalle el funcionamiento de *CBoW* y *Skip-gram*, vamos a proyectar la siguiente sentencia en dos tipos de componentes:

Las palabras resaltadas con el recuadro verde indican palabras de contexto, y para el caso de la palabra resaltada en el recuadro rojo, hace referencia a la palabra objetivo. Es importante mencionar que lo anterior se conoce como ventana, y dicha ventana puede ser más grande o incluso más pequeña, pero siempre debe de existir una palabra objetivo y sus correspondientes palabras de contexto. A partir de aquí es donde se destaca el propósito de cada técnica, con *CBoW* vamos a trabajar las palabras de contexto para poder encontrar la palabra objetivo, por lo tanto, primero se debe de convertir cada palabra de contexto en su respectiva codificación one-hot:

El	1	0	0	0	0	0	0	0	0
Lobo	0	1	0	0	0	0	0	0	0
Siempre	0	0	0	1	0	0	0	0	0
Vigila	0	0	0	0	1	0	0	0	0

El proceso continúa haciendo una suma de los vectores para generar un único vector por todas las palabras:

El lobo siempre vigila	1	1	0	1	1	0	0	0	0

Una vez que se tiene el vector se debe de generar la proyección del mismo, en otras palabras, se debe de hacer una multiplicación entre el vector anterior y una matriz, de esta forma cada palabra tendrá

una representación vectorial diferente con valores en constante adaptación durante todo el proceso de entrenamiento. Las dimensiones de la matriz dependen de dos criterios, el tamaño del vocabulario corresponde al número de filas, y el número de las columnas es un parámetro ajustable dependiente del criterio de cada autor, para el caso de *Word2Vec* los autores definieron ese criterio en 300 características por palabra. Para fines del ejemplo, la matriz será de 9 filas (el tamaño del vocabulario) y 5 características, el resultado de la operación anterior se puede ver de la siguiente manera:

1		0.31	0.07	0.30	0.86	0.85
1		0.18	0.04	0.48	0.52	0.10
0		0.49	0.15	0.20	0.90	0.08
1		0.78	0.19	0.80	0.41	0.33
1	✖	0.71	0.06	0.01	0.33	0.84
0		0.35	0.81	0.77	0.41	0.33
0		0.18	0.71	0.84	0.20	0.19
0		0.55	0.76	0.07	0.86	0.24
0		0.49	0.80	0.95	0.63	0.29

El resultado de la operación será una matriz de las mismas dimensiones, pero con valores diferentes a cero en aquellas posiciones donde existía el 1 de cada palabra, por lo tanto, esos valores corresponden ahora a la nueva representación de cada palabra dentro del vocabulario. El resultado se puede de la siguiente manera:

0.31	0.07	0.3	0.86	0.85
0.18	0.04	0.48	0.52	0.1
0	0	0	0	0
0.78	0.19	0.8	0.41	0.33
0.71	0.06	0.01	0.33	0.84
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

El proceso restante consiste en tomar las nuevas representaciones y pasarlas a través de una red neuronal, donde cada representación será una entrada distinta para la red, pero en el proceso de entrenamiento la retropropagación del error no solo se queda en las capas ocultas de la red, sino que también se ajustan los valores de la matriz, por lo tanto, la representación de cada palabra se verá ajustada, todo con el fin de irse acercando en el espacio n-dimensional los valores de las

diferentes representaciones. Los últimos pasos de CBoW se pueden observar en la Figura 2.1.

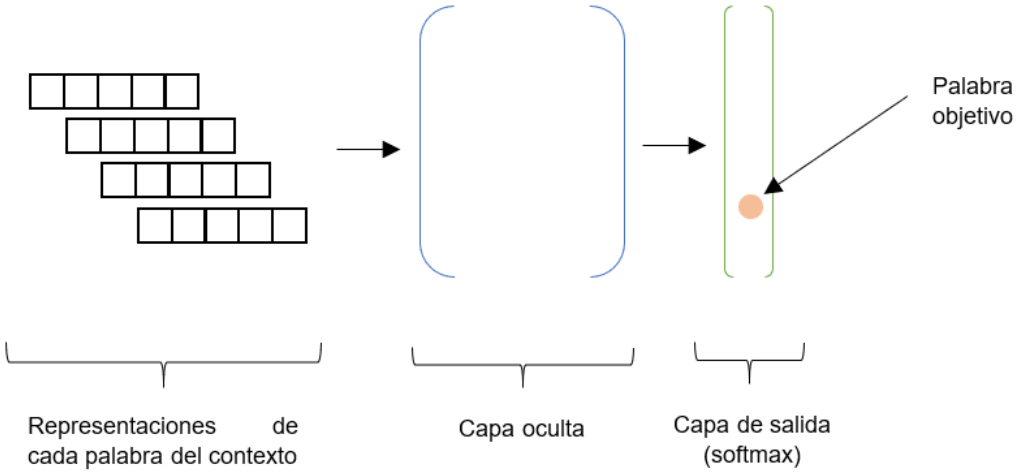


Figura 2.1: Multiplicación del vector de contexto y la matriz de pesos

Ahora para el caso de *Skip-gram* su propósito consiste en lo inverso a *CBoW*, en lugar de predecir la palabra objetivo a partir de las palabras de contexto, *Skip-gram* va a predecir las palabras de contexto a partir de la palabra objetivo. Es decir, considerando la misma sentencia:

El lobo alfa siempre vigila la manada desde atrás

Sucede la misma situación para el caso de *CBoW* hasta donde obtenemos la nueva representación, pero ahora solo es para la palabra objetivo, por lo tanto, primero convertimos la palabra objetivo en la codificación *one-hot*, después se hace la proyección y el resultado para la palabra objetivo quedaría algo así:

0	0	0	0	0
0	0	0	0	0
0.49	0.15	0.2	0.9	0.08
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

El enfoque ahora es buscar predecir las palabras de contexto, por lo tanto, la representación de la palabra objetivo estará en un proceso de constante cambio para lograr predecir a cada una de las palabras de contexto, esto es entrenar con la representación 4 veces dado que en

el ejemplo actual tenemos esa cantidad de palabras del contexto. El proceso restante de *Skip-gram* se puede ver en la Figura 2.2.

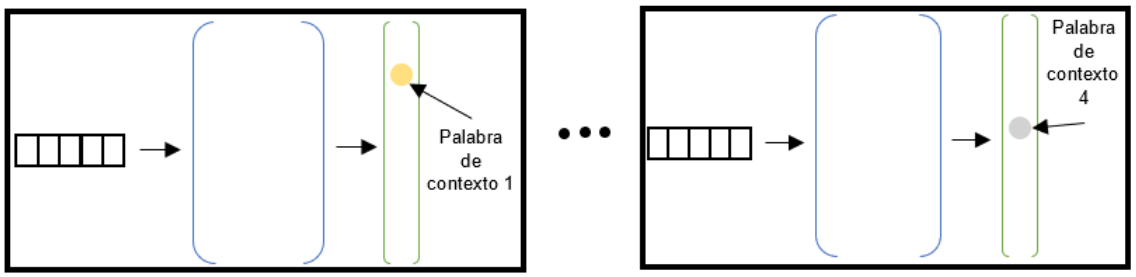


Figura 2.2: Fracción del proceso de *Skip-gram*

En general, el modelo de *Word2Vec* como se puede observar está capturando la información del texto de una manera distinta en comparación de las técnicas anteriores. Ahora las representaciones tienen información más compleja reflejada a través de números y no solamente a partir de ocurrencias, además esto sucede para cada palabra en lugar de todo un documento de texto. Si calculamos la similitud del coseno entre los vectores, se podrá observar que las palabras similares tienen una cercanía también dentro del espacio *n-dimensional*, y para el caso contrario la similitud será bastante baja, por ejemplo, para las palabras “dog”, “cat” y “fish”, si calculamos la similitud del coseno entre ellos, los resultados son los siguientes:

- (dog y cat) = 0.87
- (dog y fish) = 0.59
- (cat y fish) = 0.61

Estas representaciones han sido implementadas en diversas tareas del NLP y en su momento demostraron tener un mayor rendimiento a comparación de las técnicas tradicionales, y aun así, todavía no estaban capturando la suficiente información como para abordar todo tipo de problemas, por esa misma razón, unos años más tarde surgió otra técnica con una forma distinta de captura la información, una capaz de cambiar la representación de cada palabra según el contexto de toda la sentencia.

2.2.4. Mecanismo de atención

La última técnica descrita en este trabajo es la más popular actualmente por diversos factores, entre ellos, una mejor forma de

capturar la información del texto. Para entender el funcionamiento se plantea la siguiente sentencia "el coliseo romano realmente se llama anfiteatro flavio", la intención de este mecanismo sería para este caso, obtener información capaz de decirnos cuáles palabras tienen más relación entre sí. Para llevar a cabo un proceso habitual del mecanismo de atención, cada palabra de la sentencia tiene los siguientes vectores aleatorios entre 0 y 1:

el	0.23	0.54	0.93	0.74
coliseo	0.01	0.31	0.99	0.03
romano	0.37	0.20	0.76	0.64
realmente	0.82	0.35	0.52	0.61
se	0.44	0.83	0.31	0.65
llama	0.04	0.47	0.86	0.18
anfiteatro	0.25	0.73	0.90	0.69
flavio	0.87	0.10	0.42	0.38

Una vez que se tienen estos vectores, el siguiente paso es generar una matriz de atención, normalmente esa matriz puede tener distintos números y formas, pero para el caso de *auto-atención*, la matriz se genera a partir de los vectores anteriores. Nombremos M a la matriz generada por los vectores de arriba, y M^t a la *transpuesta* de la misma matriz.

La multiplicación matricial entre M y M^t internamente está realizando un producto punto entre los diferentes vectores, por ejemplo, para el vector *coliseo*, se hace su respectivo producto punto entre los vectores *el*, *coliseo*, *romano*, *realmente*, *se*, *llama*, *anfiteatro*, *flavio*, y esto mismo sucede con los demás vectores. El resultado se puede observar en la siguiente matriz:

1.757	1.1126	1.3735	1.3126	1.3187	1.196	1.7993	0.9259
1.1126	1.0772	0.8373	0.6498	0.5881	1.0029	1.1405	0.4669
1.3735	0.8373	1.1641	1.159	0.9804	0.8776	1.3641	0.9043
1.3126	0.6498	1.159	1.4374	1.209	0.7543	1.3494	1.1986
1.3187	0.5881	0.9804	1.209	1.4011	0.7913	1.4434	0.843
1.196	1.0029	0.8776	0.7543	0.7913	0.9945	1.2513	0.5114
1.7993	1.1405	1.3641	1.3494	1.4434	1.2513	1.8815	0.9307
0.9259	0.4669	0.9043	1.1986	0.843	0.5114	0.9307	1.0877

Hasta este punto los valores de la matriz no brindan ninguna información relevante, pero ya solo hace falta aplicar la función *softmax* para cada fila, de esta manera se convierten todos los valores

en probabilidades entre 0 y 1. El resultado se puede ver en la siguiente tabla:

	el	coliseo	romano	realmente	se	llama	anfiteatro	flavio
el	0.1804	0.0947	0.1230	0.1157	0.1164	0.1030	0.1882	0.0786
coliseo	0.1564	0.1509	0.1187	0.0984	0.0926	0.1401	0.1608	0.0820
romano	0.1639	0.0958	0.1329	0.1322	0.1106	0.0998	0.1623	0.1025
realmente	0.1447	0.0746	0.1241	0.1640	0.1305	0.0828	0.1502	0.1291
se	0.1534	0.0739	0.1093	0.1374	0.1665	0.0905	0.1737	0.0953
llama	0.1602	0.1321	0.1165	0.1030	0.1069	0.1310	0.1694	0.0808
afiteatro	0.1790	0.0926	0.1158	0.1142	0.1254	0.1035	0.1944	0.0751
flavio	0.1301	0.0822	0.1273	0.1709	0.1198	0.0860	0.1307	0.1530

La matriz actual es la conocida como *matriz de atención* y su propósito es reajustar los valores originales con base en la nueva información, de hecho se puede observar cómo cada palabra tiene asociado un valor de importancia hacia otra palabra, y así para todas las palabras que conforman la sentencia. Estos valores se deben de procesar una vez más con los vectores originales de cada palabra, es decir, con ello se llega al siguiente resultado:

el	0.5898	0.5814	0.6017	0.6046
coliseo	0.6476	0.6447	0.6691	0.6635
romano	0.6486	0.6458	0.6704	0.6645
realmente	0.6189	0.6136	0.6359	0.6345
se	0.6146	0.6087	0.6303	0.6303
llama	0.6293	0.6248	0.6478	0.6451
anfiteatro	0.6492	0.6465	0.6710	0.6652
flavio	0.6486	0.6457	0.6699	0.6647

La representación dada originalmente a cada palabra cambia debido a la *matriz de atención*, pero es importante entender que todos los valores tratados hasta ahora son valores aleatorios, por eso no hay todavía una relación lógica entre los valores de importancia de cada palabra. Dado lo anterior, es necesario encontrar nuevas representaciones con relaciones adecuadas cuando así lo dicte la sentencia, y a su vez generar *matrices de atención* enfocadas en el punto donde se encuentra dicha atención entre las palabras. Por lo anterior, los autores del artículo (Vaswani *et al.*, 2017), proponen una forma de buscar las nuevas representaciones del texto con un enfoque bastante innovador.

Entre los diferentes puntos importantes a destacar del artículo, está

el bloque *Multi-Head Attention*, el cual tiene un funcionamiento similar al explicado anteriormente. La fórmula para obtener la nueva representación con respecto a lo propuesto en el artículo, es la siguiente:

$$Attention(Q, K, V) = softmax(\frac{Q \times K^T}{\sqrt{d_k}}) \times V \tag{2.1}$$

Donde las variables Q, K y V provienen de la misma sentencia, pero en todos los casos la sentencia debe de pasar por diferentes capas lineales, por ejemplo, si tenemos una sentencia X, entonces X deberá pasar por 3 ANNs de 2 capas cada una, la de entrada y la de salida, ambas de las mismas dimensiones, esto para generar una nueva representación que en el entrenamiento se estará ajustando según sea el error. Por último, en el caso de $\sqrt{d_k}$ hace referencia a la longitud de palabras dentro de la sentencia, y para recibir el nombre de *Multi-Head attention*, todo lo anterior se hace en paralelo n veces, donde n se determina a criterio de cada autor.

Considerando la forma en la que funciona lo propuesto, las representaciones del texto siempre se verán ajustadas dependiendo del contexto de la sentencia, por lo tanto, ni su representación, ni las matrices de atención son estáticas. Además, dado el comportamiento del mecanismo de atención sobre todas las palabras de la sentencia, no importa la longitud de la misma, por ejemplo, si hay relación entre la palabra 1 y la palabra 17, esa relación eventualmente será captada por el mecanismo.

2.2.4.1. BERT y BETO

En esencia, de este último enfoque salen modelos como BERT y BETO, ambos generan nuevas representaciones de texto, los cuales fueron entrenados para diferentes idiomas, pero con el mismo propósito, obtener representaciones que puedan ser utilizadas para otras tareas en específico. Aparte de la nueva representación por palabra, también existe una representación basada en concatenar todas en una sola, para así generar un solo vector de las mismas

dimensiones como las de una palabra, pero en este caso estaría compactando toda la sentencia, dicho vector recibe el nombre de token [CLS]. En el caso de BETO sus datos de entrenamiento fueron todos los datos de Wikipedia, revistas gubernamentales y de las naciones unidas, charlas de TED, subtítulos, entre otros.

Ambos modelos fueron entrenados en tareas como predecir la palabra que falta dentro de una sentencia, así como generar una sentencia a partir de otra. También hacen la implementación de un tipo diferente de tokenización llamado *wordpiece*, la idea de este enfoque es generar un vocabulario a partir de la frecuencia existente entre los diferentes caracteres que conforman las distintas palabras. El propósito es generar un vocabulario donde van a existir fragmentos de palabras y caracteres, ya que todos forman parte del propio texto, pero solo algunas palabras se formarán en su totalidad dado que entre sus caracteres fueron los más frecuentes.

2.3. Selección de características

El objetivo de esta técnica es encontrar un subconjunto de características a partir de un conjunto original, el cual alcance un mejor desempeño dentro de alguna tarea dada, por ejemplo a la hora de alimentar a un clasificador. Para ello existen diversas formas de hacerlo, por esa razón cuando se busca hacer una selección de características, es habitual encontrar diferentes enfoques como lo son el de Filtro, Envoltura y Embebido.

En el campo del NLP la selección de características es aplicada en las diferentes representaciones que recibe el texto, sobre todo porque la cantidad de datos a procesar cada vez es mayor. Por eso se vuelve imprescindible aplicar diferentes tipos de filtros para recuperar la información valiosa y desechar la que no, esto con el fin no solo de reducir el costo computacional, sino también de aumentar el rendimiento de las características originales. Para entrar un poco más en detalle de los diferentes enfoques, se describirán brevemente en las siguientes secciones.

2.3.1. Filtro

El primero es el más utilizado debido a su simplicidad y bajo costo de implementación, en este caso la especialidad del enfoque se encuentra en conocer las similitudes entre las características, o la frecuencia, todo con base a las diferentes clases propias para cada conjunto de características. El paso principal para este enfoque es la representación del texto, la más famosa es la descrita anteriormente, BoW, pero existe una alternativa como la Frecuencia de Termino - Frecuencia Inversa de Documento, por sus siglas en inglés, TF-IDF. Recibe el nombre por la forma de llevar a cabo su funcionamiento, primero busca la frecuencia de las palabras dentro de un dominio y a su vez busca la frecuencia de los mismo términos en más de un dominio, con eso se puede hacer un filtrado inverso, donde las palabras más repetidas en varios dominios terminan siendo desechadas para la entrada del clasificador.

Un método de filtro bastante utilizado es el de Ganancia de Información, por sus siglas en inglés, IG, en esta técnica la relevancia se encuentra en un concepto llamado *Entropía*, el cual se puede conocer como la varianza existente entre un conjunto de datos y sus distintas clases, la fórmula de la entropía es la siguiente:

$$E = -\sum_{i}^{n} P(x_i) \log P(x_i) \tag{2.2}$$

Donde n es el número total de instancias, $P(x_i)$ la probabilidad de la clase x_i y el log está en base 2. Sí el resultado es alto, significa que la entropía entre los datos es significativa, es decir, que existe bastante incertidumbre entre las diferentes clases. Por otro lado, si el número es pequeño entonces no hay tanta dispersión entre ellas.

Dada la *Entropía*, primero se debe calcular aquella con base a todas las diferentes clases, de esa forma se obtiene la *entropía* principal, seguido de ello, el paso siguiente es reducir el valor de la entropía, también conocido como Ganancia de Información, *Information Gain* para el inglés (IG). Entre los diferentes atributos que determinan las

diferentes clases, se calcula con cuál se obtiene la IG más alta, la formula general se puede ver de la siguiente forma:

$$IG = E(C) - E(A, C) \tag{2.3}$$

Donde $E(C)$ hace referencia al cálculo de la entropía general, y $E(A, C)$ es la entropía general dado un atributo seleccionando en ese momento. En esta técnica es común hacer su implementación junto con un clasificador como lo es un árbol de decisión.

Estos solo fueron algunos métodos, pero existen diversas formas de obtener información relevante a partir de otros, varios son descritos en (Deng *et al.*, 2019). A grandes rasgos este es el funcionamiento de un enfoque de filtro, utilizar diferentes técnicas para seleccionar las características, y después aplicar un clasificador para medir el rendimiento del nuevo subconjunto de características.

2.3.2. Envoltura

La diferencia principal entre el enfoque anterior y el actual se encuentra en el proceso de búsqueda respecto al nuevo conjunto de características, en el caso anterior, para seleccionar las mejores características se aplica una formula u operación matemática y directamente las nuevas características entran a un clasificador. En este caso es un proceso donde la selección se hace de manera gradual con base a los resultados obtenidos por parte de un clasificador. Para observar de manera general el proceso iterativo de un enfoque de envoltura se encuentra la Figura 2.3.

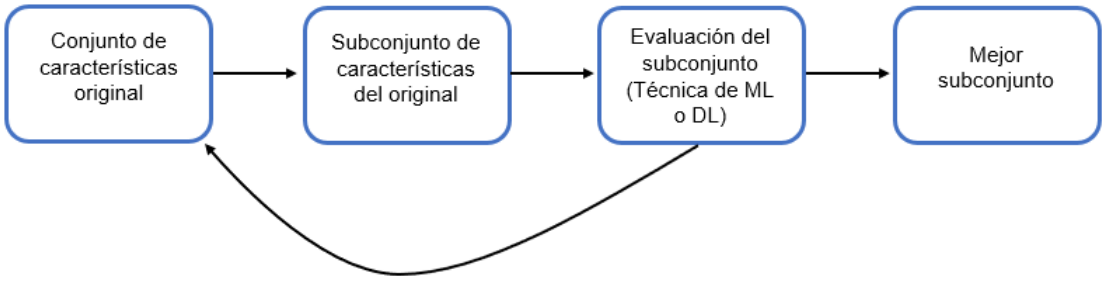


Figura 2.3: Estructura general de una FS desde un método de envoltura

Es importante resaltar la presencia de un *costo-beneficio*, se aumenta la

calidad de las características seleccionadas, pero existe un mayor costo computacional y de complejidad en la implementación. Un ejemplo de este enfoque se puede estudiar detalladamente en toda la Sección 4.

2.3.3. Embebido

Hasta este punto, la selección de características entre los diferentes enfoques se puede ver como cada uno va siendo más intrusivo con respecto al anterior, es decir, la manera en cómo se obtienen las nuevas características tiende a involucrar más al proceso de donde se evalúan. Incluso en la literatura se menciona al enfoque embebido como la integración de utilizar el de filtro en conjunto con el de envoltura, por ejemplo, en el artículo de (Imani *et al.*, 2013), hacen la respectiva colaboración de ambos enfoques para la tarea de clasificación de texto.

En general las tres técnicas tienen el mismo propósito, reducir o filtrar la dimensionalidad de la entrada aumentando el rendimiento de la misma, cada una con sus diferencias a la hora de la implementación, por lo tanto, también cada una tiene sus ventajas y desventajas. El uso de cada una de ellas es totalmente dependiente del problema que se está abordando, entre más complejo sea el problema, la técnica por ende tendrá que ser de la misma complejidad, incluso puede existir la combinación de varias como ya se había mencionado previamente.

2.4. Cómputo evolutivo

Esta metodología es ampliamente utilizada en diversas áreas, tiene su inspiración en algunos aspectos de la naturaleza, como lo son el comportamiento de las aves y su vuelo en conjunto, el de una colmena de abejas y la forma en la que polinizan las flores, también en como las hormigas realizan su búsqueda de alimentos, e incluso en el comportamiento de un cumulo de partículas. Todo lo anterior también se conoce como técnicas de optimización, y su fin es abordar estas analogías dentro de un problema para generar todo un espacio finito de soluciones y encontrar si es posible el óptimo global. Para

este trabajo de investigación la inspiración está en la *evolución*, así como generación tras generación sobreviven los más aptos de una especie, también se puede aplicar a nivel de cómputo en una técnica bastante conocida, esta es, el *Algoritmo Genético*. Se puede encontrar bastante información de *Cómputo Evolutivo* en la literatura, hay trabajos con suficiente nivel de detalle en las diferentes formas de implementación, desde autores de antaño como (Dumitrescu *et al*., 2000), (Tang & Wu, 2011), hasta autores con un enfoque más actual y por lo tanto innovador (Kulkarni *et al*., 2021). Para entender más sobre esto, la Sección 4.1 describe significativamente toda la implementación de un *Algoritmo Genético*, desde la generación de la población inicial, hasta la forma y criterio de reemplazo.

2.5. Aprendizaje profundo

Actualmente en el desarrollo de la IA, los modelos más competitivos tienen en su interior un gran número de capas de neuronas artificiales, por lo que se les da el nombre de *Aprendizaje Profundo*. Actualmente existe una variedad en la implementación de una técnica de DL, pero por razones de este trabajo, se van a describir los conceptos más utilizados en el desarrollo de la propuesta, abordando desde el componente básico (*Neurona*), hasta el cómo trabajan en conjunto para lograr el rendimiento que tienen en diversas tareas.

2.5.1. Red neuronal artifical

Inicialmente el concepto de una red neuronal viene dado desde el aspecto biológico, donde los autores (Mcculloch & Pitts, 1943) describen cuál es el funcionamiento natural de una red neuronal, es decir, cómo es que interactúan entre sí las neuronas, sus conexiones, su activación, entre otros conceptos. Además, asumen que una neurona por sí sola no tiene la capacidad de activarse y transmitir información, por ello es importante llegar a la conformación de una estructura fija de varias neuronas conectadas. La unidad básica de una red neuronal en términos computacionales era conocida como *Perceptrón*, en la actualidad está conformada por los siguientes componentes:

1. *Entrada*: Es un valor numérico propio de un conjunto de datos
2. *Peso*: Cada entrada tiene cierta información y el *Peso* se encarga de ir regulando la importancia de cada *Entrada*
3. *Sesgo*: Es un valor con la intención de alterar la operación principal de una *Neurona*
4. *Función de activación*: El propósito de la función es evitar la linealidad intrínseca de una *Neurona*
5. *Salida*: Es el valor resultante después del proceso.

La Figura 2.4 refleja cada uno de los componentes descritos anteriormente. La función principal de una *Neurona* es hacer una multiplicación con respecto al *Peso* y la *Entrada*, sumar el *Sesgo* y por último, aplicar la *Función de activación* para obtener una *Salida*, la operación se puede observar en la Formula 2.4.

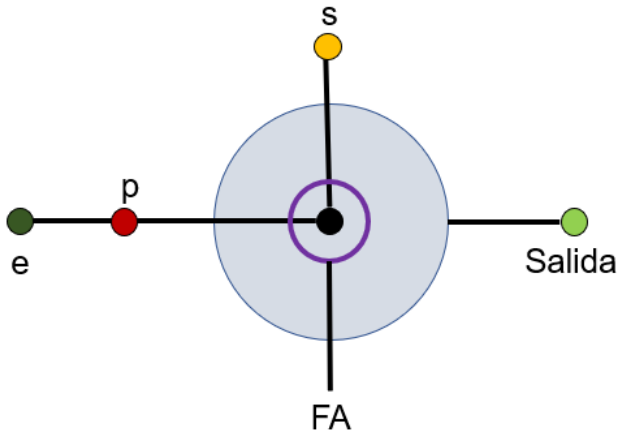


Figura 2.4: Una neurona artificial

$$Salida = FA((ep) + s) \tag{2.4}$$

Donde FA es la *Función de activación*, e es la *Entrada*, p el *Peso* y s el *Sesgo*. Considerando un problema trivial solo para ejemplificar el uso de una neurona y su crecimiento, sería algo como elegir la compra de una computadora, si el valor de salida es negativo no se compra y si es positivo se compra, entonces, la *Entrada* en el primer caso es si la computadora tiene o no un buen precio, el *Peso* tendría un valor aleatorio entre 0 y 1, solo para alterar un poco el valor de entrada, el *Sesgo* en este caso solo tendría la intención de sumar más valor para

aumentar la probabilidad de comprar la computadora, por último, la función de activación es una de las más sencillas, nombrada Unidad Lineal Rectificada, por sus siglas en inglés, *ReLU*, su mecanismo de operación solo consta de volver todo valor negativo a 0 y todo valor positiva lo pasa tal y como llegó. Dado el caso anterior, en la Figura 2.5 se pueden observar asignados todos los parámetros y la salida del proceso.

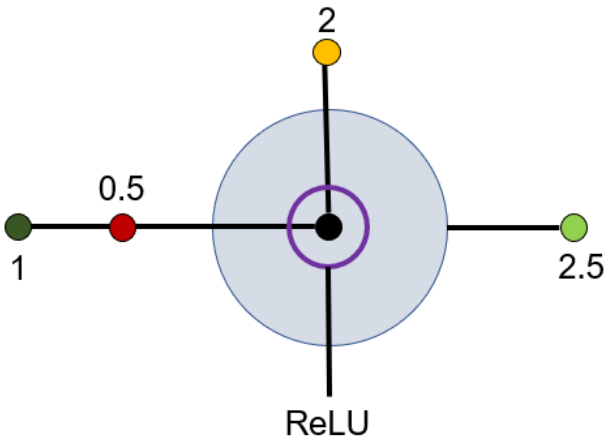


Figura 2.5: Ejemplo de una neurona en función

Debido al contexto del ejemplo, todo el tiempo la salida de la neurona será positiva, por lo tanto, la computadora se va a comprar. Bajo otro panorama donde se agregan más valores de *Entrada*, por ejemplo, si se tiene el presupuesto o no, si hay en inventario o no, y si tiene buenas referencias o no, todas estas entradas tienen un enfoque binario, pero van agregando más valor a la entrada, esto en consecuencia tendrá un resultado diferente en la salida. En la Figura 2.6 se puede observar cómo se agregan más *Entradas* y *Pesos* a la neurona, así como también cambian dichos valores incluso el del *sesgo*. El aumento de valores afecta la formula anterior y esto se puede ver ajustado en la Formula 2.5.

$$Salida = FA((\sum_{i=1}^{n} e_i p_i) + s) \tag{2.5}$$

Las variables tienen el mismo propósito como en la Formula 2.4, solo para el caso de n su valor depende del número de valores de entrada. Dado que tener un buen precio y buenas referencias no fue suficiente, el resultado dio como salida un valor negativo, y dada la

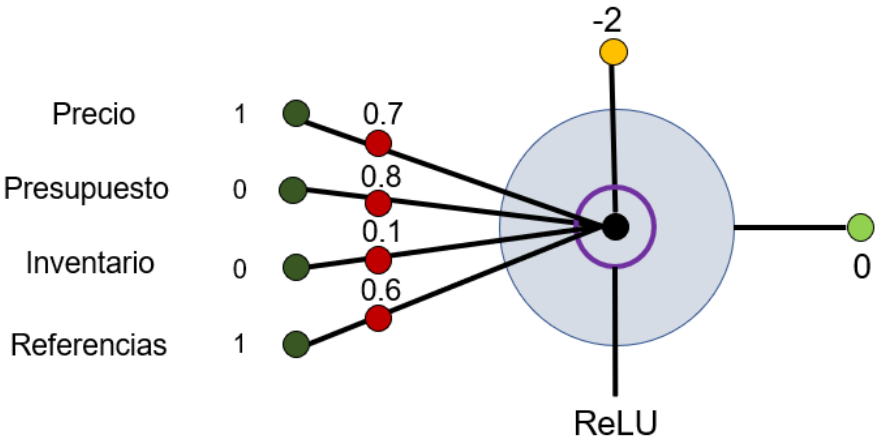


Figura 2.6: Ejemplo de una neurona en función con más entradas

función de activación, el valor se vuelve directamente a 0. Es importante notar como el comportamiento de la *Neurona* ahora está dependiendo de más variables, los pesos se pueden manipular, al igual el sesgo e incluso se puede cambiar la función de activación y hacer que la salida cambie totalmente.

Si el problema fuera diferente, donde las entradas ya no son variables binarias como *Precio*, *Presupuesto*, *Inventario* y *Referencias*, ahora están representando un texto, y la salida ahora proyecta diferentes polaridades, el *Perceptrón* no tendría la capacidad de ajustar sus parámetros para abordar el problema. En estos casos es donde la unión de varias *Neuronas* entre varias *Capas*, permiten expandir la capacidad de reconocer patrones, a esto también se le conoce como un *Perceptrón Multicapa*, aunque en la actualidad ya es más conocido como *Red Neuronal Artificial* o *ANN*. Un ejemplo de una ANN se puede observar en la Figura 2.7.

Los pesos representados con el círculo rojo ahora solo son las líneas conectadas entre las diferentes *Neuronas*, lo demás sigue representando lo mismo, tal y como se explicó en el primer caso cuando solo era una *Neurona*. Las *Capas* se pueden dividir en tres aspectos generales, la *Capa de entrada* (los círculos verdes oscuro), la *Capa oculta* (todas las neuronas antes de la salida) y la *Capa de salida* (los círculos verdes claro), es importante resaltar la ausencia de una operación en la capa de entrada, solo se están reflejando los datos, por otro lado, para la capa de salida sí hay el mismo proceso como el

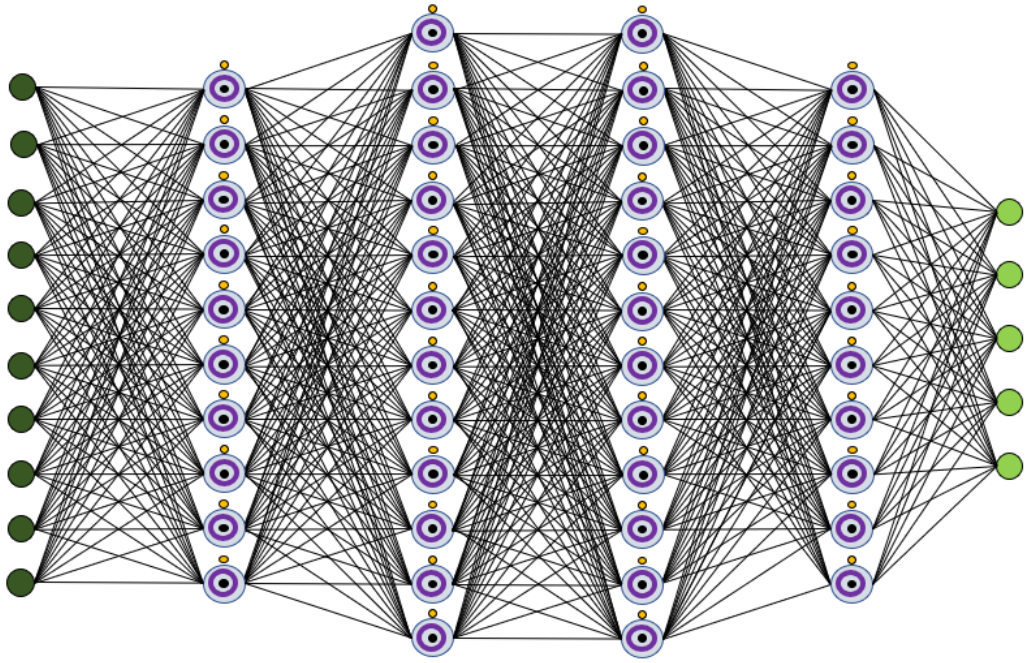


Figura 2.7: Una red neuronal artificial

de una *Neurona*. Es una buena práctica agregar en la capa de salida una *Función softmax* en lugar de una *Función de activación*, con esto cada salida se convierte en una probabilidad entre 0 y 1.

En el problema sobre comprar la computadora, los parámetros de *Pesos* y *Sesgos* ya estaban asignados, pero siguiendo con el problema de clasificar un texto, el número de parámetros aumentó significativamente, por eso aquí se encuentra el punto medular de cualquier ANN, es decir, ajustar todos los parámetros de manera **automática**. Para ello existe la famosa *Retropropagación* (mejor conocida en el inglés como *Backpropagation*), primero se calcula el *error* entre la salida de la red y el valor esperado, dicho *error* junto con un hiperparámetro nombrado *Taza de aprendizaje* (más buscado en el inglés como learning rate), se *retropropagan* por todas las neuronas de la red y en su paso van ajustando los parámetros con la finalidad de ir reduciendo el *error*, este proceso se repite una cantidad finita de Épocas.

En el estado del arte hay muchas tareas con un alto rendimiento usando las ANNs, pero las más célebres actualmente son aquellas con un número de *Neuronas* que se disparan a niveles vertiginosos,

dicho de otra forma, las arquitecturas tienen miles de millones de *Parámetros* ajustables, por esa razón reciben el nombre de DL. También el éxito se debe a todo el estudio y las diferentes formas de calcular el error, optimizar los parámetros, de aplicar técnicas distinguidas en el inglés como el *Dropout*, *Early stopping*, y otras más complejas como las CNN, las Redes Neuronales Recurrentes, entre otras. En la siguiente sección se entra en materia con respecto a la CNN.

2.5.2. Red neuronal convolucional

Una CNN en esencia sigue siendo una ANN, la diferencia se encuentra en una operación extra agregada entre la capa de entrada y la capa oculta, la cual consiste en aplicar una convolución en el proceso de entrenamiento, esto para extraer características que puedan ser más significativas y en consecuencia generar un mejor desempeño. El componente principal de la convolución es el *Kernel*, una matriz con valores ajustables, donde su tamaño puede variar a partir de la evidencia empírica. Considerando la naturaleza del *Kernel*, su aplicación se debe de realizar sobre otra matriz, por eso era habitual ver esta implementación en trabajos relacionados con imágenes, ahora también se aborda la convolución en la representación del texto. Para entender su funcionamiento, inicialmente se va a generar de manera aleatoria un *Kernel* de $3x3$, sobre una matriz de $8x10$, donde cada fila corresponde a una palabra y cada columna a las características de la misma, la sentencia sería algo como “el coliseo romano realmente se llama anfiteatro flavio”. En la Figura 2.8 se puede observar lo descrito anteriormente.

El *Kernel* se va a sobreponer sobre la matriz de texto, empezando desde la parte superior izquierda y haciendo un recorrido de uno en uno hacía la derecha hasta llegar al borde, una vez ahí, el *kernel* baja al siguiente renglón y sigue de nuevo hasta el extremo de la derecha, así sucesivamente hasta haber recorrido toda la matriz del texto. En cada paso del *kernel*, tanto sus propios valores como los que están abajo en la matriz del texto, realizan la operación de un producto punto, en consecuencia, el valor resultante de cada operación pasa a formar parte de una nueva matriz. Dado el comportamiento de una

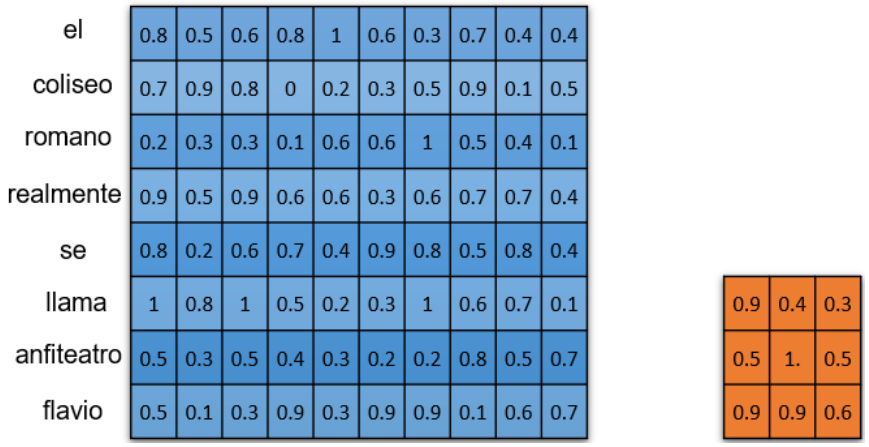


Figura 2.8: Componentes de una convolución

convolución, la nueva matriz tendrá una forma diferente a la matriz original, algo determinado a partir del tamaño del *kernel*, es decir, en el ejemplo actual el *kernel* es de $3x3$ y la matriz de texto de $8x10$, por lo tanto, la nueva matriz tendrá una forma de $6x8$. Esto debido a la resta $3 - 1$ del número de filas y columnas de la matriz de texto, por otro lado, si el *kernel* tuviera la forma $5x5$, se resta $5 - 1$ y la nueva matriz de texto sería de $4x6$, esta lógica se cumple cuando el tamaño del *kernel* es simétrico, de lo contrario se debe hacer el mismo proceso considerando de manera individual la resta entre las filas y columnas. En la Figura 2.9 se puede observar una operación de convolución.

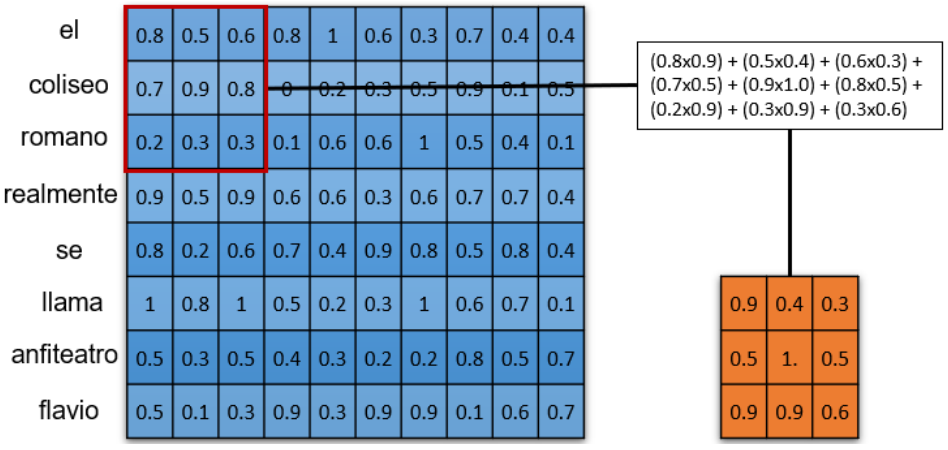


Figura 2.9: Ejemplo de una convolución

En la CNN después de hacer la convolución sigue un proceso de agrupación, mejor conocido en el inglés como *pooling*, el cual consiste

en determinar un número de filas y columnas para calcular su promedio, o bien tomar el valor máximo de ese grupo. El resultado es nuevamente una matriz, la cual también reduce su tamaño dependiendo de la propia ventana del *pooling*, cabe destacar una diferencia entre el desplazamiento de la convolución y el *pooling*, en este caso el movimiento de la ventana no es de 1 en 1, en la literatura el salto por lo regular es del mismo tamaño como las dimensiones de la ventana. Considerando una configuración del *pooling*, donde las ventanas son de $2x2$ y se desea extraer el valor máximo de cada una, en la Figura 2.10 se puede observar el resultado de la convolución anterior y el resultado del *pooling*.

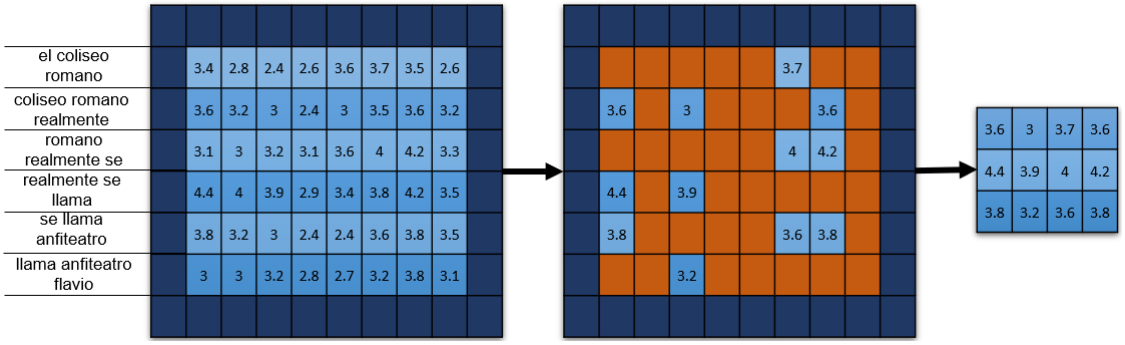


Figura 2.10: Resultado de una convolución y *pooling*

Existe mucho trabajo acerca de las CNNs en el ámbito de las imágenes, aunque también hay un poco en el aspecto del texto, en ambos casos la convolución por lo regular se hace de manera secuencial, es decir, se conecta todo el proceso detallado anteriormente a otra convolución, esto va generando nuevas matrices entre cada convolución, cada vez con información más condensada. Lo importante aquí es mencionar un proceso alterno al secuencial, en otras palabras, un proceso paralelo, donde las convoluciones no están conectadas una después de otra, al contrario, cada convolución es independiente a la otra y para cada una entra la misma matriz inicial, pero el resultado son diferentes matrices según el número de convoluciones en paralelo. Por último, el paso final antes de llegar a la ANN, es aplanar las matrices, conectar si es el caso, cada vector con los demás, y generar una sola entrada para la ANN. Lo anterior se menciona porque los vectores finales son conocidos en la literatura como *mapas de características*, y en bastantes

casos se generan varios *kernels* a la vez para tener muchos *mapas de características*.

Ambos enfoques tienen su propósito, por un lado con la secuencial se condensa más la información, y por el otro se tiene una mayor variación en los datos de entrada. En la sección 5.2 se presenta más a detalle las diferencias entre los dos enfoques, incluso se describen algunos experimentos para conocer de manera más clara su comportamiento.

3. Marco Referencial

Actualmente se genera demasiada información en formato de texto a través de los medios sociales, tanto es el caso, que se vuelve indispensable utilizar estrategias para procesar y obtener la información más relevante, algo donde la *FS* puede desenvolver un buen papel. Para implementar una *FS* existen diferentes formas (filtro, envoltura y embebido), conocidas en el inglés como *Filter*, *Wrapper* y *Embedded*, todo esto descrito con más a detalle en (Deng *et al*., 2019). Los mismos autores mencionan como al usar un enfoqué de envoltura tiene un mayor costo a nivel computacional y de complejidad, pero también presentan un mejor rendimiento a diferencia del enfoque de filtro. En el trabajo de (Thirumoorthy & Muneeswaran, 2022) se exponen diferentes métodos de *FS* y usan técnicas de Aprendizaje Automático, ML, por sus siglas en inglés, para conocer su rendimiento, además tiene un enfoque de filtro bastante remarcado, y todo lo comparan contra varias técnicas de selección de características. Incluso mencionan trabajos donde utilizan el método de envoltura por sí solo, y combinado con el de filtro (Cai *et al*., 2018).

Existe un amplio panorama en lo referente a la selección de características, pero solo algunos trabajos mencionan dichas técnicas utilizando una representación del texto a través vectores numéricos por cada palabra, entre ellos, el trabajo de (Uysal & Murphey, 2017) hace una relevante comparación entre las técnicas tradicionales de *FS* con Aprendizaje Automático que utilizan un representación de tipo *BoW* contra algunas técnicas de Aprendizaje Profundo, DL, por sus siglas en inglés, con una representación de tipo *vector-palabra*. Al final de su trabajo se puede destacar como los métodos de DL tienen un mejor rendimiento en comparación de los métodos tradicionales, en específico resalta con los resultados utilizando una CNN.

También hay un trabajo donde mencionan un problema presente en *BERT* a la hora de trabajar con textos largos, y para tratar de

proponer algo distinto a solo incrementar el número de GPUs, utilizan seis métodos de selección de características junto con *BERT*, es decir, primero aplican los métodos de selección de características para reducir la longitud de los textos, y después, los textos reducidos se convierten en la respectiva entrada de *BERT*, así logran pasar de texto largos a unos cortos donde solo tienen los tokens más relevantes para la tarea dada, (Wang *et al*., 2021), en los resultados se puede observar una cierta mejora en algunos casos aunque no destaca mucho.

Entre otros trabajos con la representación vectorial por tokens, pero utilizando un modelo cómo *Word2Vec*, se encuentra el de (Tian *et al*., 2018), donde como en el caso anterior combinan métodos de selección de características con la representación de *Word2Vec*. La diferencia recae principalmente en la manipulación de las características de cada una de las palabras, en este caso se trabajan una vez ya tienen su representación por parte del modelo. Es importante destacar como en el trabajo de (Wang *et al*., 2021) aunque ya estamos tomando la representación de *BERT* realmente no está habiendo una selección de características directa con esa representación, es más depuración antes de convertir los tokens a su respectiva representación numérica.

La *FS* hasta ahora tiene un enfoque bastante inclinado al de Filtro, y si se trata de guiar la búsqueda para un enfoque de Envoltura, está el trabajo de (Uğuz, 2011), donde la representación vuelve a ser con base en BoW, no en *BERT*, pero si tenemos una aproximación de cómo se puede hacer una *FS* a través de un GA. En los resultados se puede notar como para un clasificador las características seleccionadas entre menor sean, mayor es el *accuracy*, y para el otro caso el rendimiento se mantiene casi similar. Otra aportación con un enfoque de envoltura similar, orientado a mejorar el *accuracy* en textos de diagnósticos médicos, se encuentra el trabajo de (Antony Gnana Singh *et al*., 2016). En este panorama la dimensionalidad trabajada no es tan amplia como en los casos anteriores, pero sí muestran un interesante comportamiento que se encuentra entre

algunos operadores de cruza utilizados. Todos los trabajos expuestos hasta ahora están reflejados en la Tabla 3.1, en la cual se puntualiza la diferencia entre ellos.

Autores	tipo de FS	Representación	Técnica de clasificación	Aporte
Thirumoorthy y Muneeswaran, 2022	Filtro	Tipo BoW	NB y SVM	Una comparación
Uysal y Murphey, 2017	Filtro	Tipo BoW y Word Embeddings	SVM, CNN, LSTM y CNN+LSTM	Una comparación
Wang *et al.*, 2021	Filtro	BERT	fine-tuning	Nueva técnica
Tian *et al.*, 2018	Filtro	Word2Vec	SVM	Word2vec-SM
Uğuz, 2011	Envoltura	Tipo BoW	K-NN y C4.5	Una comparación
Antony Gnana Singh *et al.*, 2016	Envoltura	Tipo BoW	NB, J48 y K-NN	Una comparación

Tabla 3.1: Trabajos relacionados con respecto a FS

Hasta ahora ningún trabajo está orientado a una selección de características bajo el modelo de *BETO*, todos están enfocados en trabajar con otra representación o con modelos entrenados para el idioma inglés. Si el propósito solo está en trabajos utilizando el modelo *BETO* para otras tareas, encontramos trabajos como el de (Angel *et al.*, 2023), donde utilizan el modelo *BETO* para la clasificación de polaridad de una opinión, tipos de lugares turísticos y la ciudad en donde se presenta ese ambiente turístico, además de aplicar ciertos métodos para tomar solo instancias del conjunto de datos total, lo cual demuestra en los resultados tener un buen rendimiento. Un trabajo que utiliza el mismo conjunto de datos anterior es el de (Castorena-Salas *et al.*, 2023), aquí demuestra como la *SVM* tiene un buen rendimiento en comparación de *BETO*, pero es un *BETO* sin *fine-tuning*, algo con gran importancia porque los pesos de cada token ya no se ajustan para el enfoque de la tarea. También está el trabajo de (Pan *et al.*, 2023) utilizando *BETO* pero en un contexto más a las finanzas, y sobre ese mismo dominio está el trabajo de (Garcia-Diaz *et al.*, 2023).

Hay más trabajos, sobre aspectos similares, centrados en probar el modelo con diferentes conjuntos de datos y diferentes tareas, pero no bajo la perspectiva de este trabajo de investigación, lo cual agrega mayor relevancia a la implementación de la FS con un GA para la búsqueda de una mejor representación, esto puede complementar a trabajos como los anteriores, donde incluso podrían tener un enfoque similar pero con diferente representación.

4. Desarrollo de la propuesta

4.1. Metodología

En primera instancia, el proceso de este trabajo de investigación comienza con la implementación del método de envoltura a través del GA, a fin de buscar si existe un subconjunto de las características originales que puedan garantizar no solo eficiencia, sino también un mejor rendimiento. Además de comprobar las dos hipótesis propuestas con respecto al número de características necesarias según sea la cantidad de datos, y si las características seleccionadas tienen la capacidad de mantener un buen rendimiento en diferentes dominios. Todo esto se mantendrá a través de una constante comparación entre las características originales y las características seleccionadas, además del análisis necesario para trabajar con los diferentes conjuntos de datos utilizados.

Es importante mencionar que todo el proceso descrito en la siguiente sección fue trabajado en el lenguaje de programación Python. Además, la técnica del GA se utiliza debido a su gran capacidad de búsqueda, así como por la representación binaria de los individuos, la cual se adapta perfectamente a las necesidades de la propuesta. El primer proceso es la implementación del GA, así que inicialmente se va a describir todo lo concerniente a su implementación. El proceso general de un GA se puede ver gráficamente en la Figura 4.1.

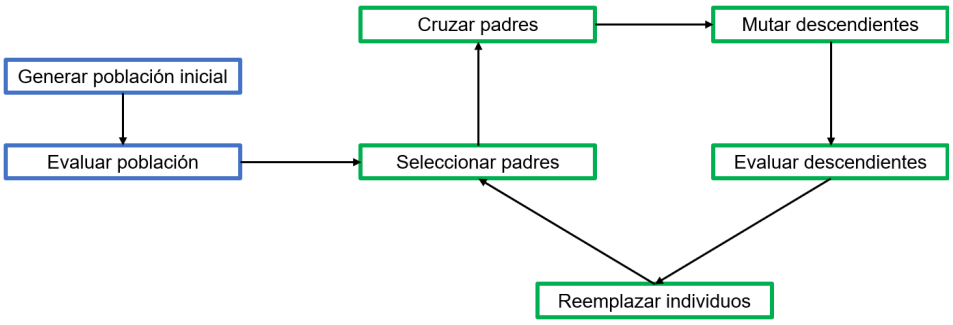


Figura 4.1: Estructura general de un algoritmo genético

4.1.1. Implementación del algoritmo genético

El primer paso para cualquier algoritmo genético es la generación de su población, en este caso cada individuo es generado aleatoriamente como un vector binario, es decir, dado un número aleatorio n, se produce un conjunto $[i_1, i_2, i_3, ...i_n]$ de índices de manera aleatoria (el rango de los índices aleatorios debe de estar entre 0 y 767. Dado los índices se determina en que posiciones de un vector de 768 se reemplazarán los valores en 0 por un 1, de esta manera se tendrán las características a seleccionar y las que se van a omitir. Para expresar lo anterior desde otro enfoque y de forma más detallada, se puede observar el Algoritmo 1, también se puede ver el ejemplo de un vector binario en la Figura 4.2. Es importante mencionar que en la variable na, donde se asigna la función que genera un número aleatorio entre 1 y 767, el 2do parámetro es 768 porque la función $random.randint()$ de la biblioteca Numpy de Python excluye ese valor y solo genera los valores aleatorios dentro del rango de 1 y 767.

Algoritmo 1: Generar la población

Datos: $numeroIndividuos$
Resultado: $poblacion$
$poblacion \leftarrow matrizCeros(numeroIndividuosx768)$;
$i \leftarrow 0$;
mientras $i < numeroIndividuos$ **hacer**
  $na \leftarrow numeroAalteorio(1, 768)$;
  $vi \leftarrow vectorIindices(0, 768, na)$;
  $individuo \leftarrow vectorbinario(vi)$;
  $poblacion[i] \leftarrow individuo$;
  $i \leftarrow i + 1$;
fin

Figura 4.2: Representación de un vector binario

Una vez que la población ha sido generada, ahora ya tenemos un número n de diferentes representaciones, por consecuente, empieza el proceso más importante del GA, **la evaluación** de cada uno de los individuos para conocer sus distintos rendimientos. El resultado de los individuos es conocido como **Aptitud**, y con ello podemos mantener ordenados a los individuos del mejor al peor. El proceso de forma general para evaluar un individuo lo podemos describir de la

siguiente forma:

1. Obtener la nueva representación del texto
2. Generar la ANN a partir del número de características del individuo actual y el número de tokens
3. Entrenar y evaluar la red neuronal con el conjunto de datos

Considerando el primer paso, dado el texto de ejemplo, *"Compartir el conocimiento es algo bueno"*, obtenemos su correspondiente representación matricial por parte de BETO, donde cada fila representa un token y cada columna representa las características que pertenecen a ese token. Podemos ver el texto representado con su forma matricial en la Figura 4.3. Después de obtener la matriz de texto, se hace el proceso de seleccionar a partir de los valores binarios del individuo, esto es, las posiciones donde el individuo tiene un valor igual a 1 se considera como una característica candidata para ser seleccionada, por otro lado, las posiciones donde el valor es 0 la característica será eliminada. Se puede observar el proceso gráficamente en la Figura 4.4, donde el color gris de cada celda representa un 0 y para el caso de las celdas color verde representan un 1.

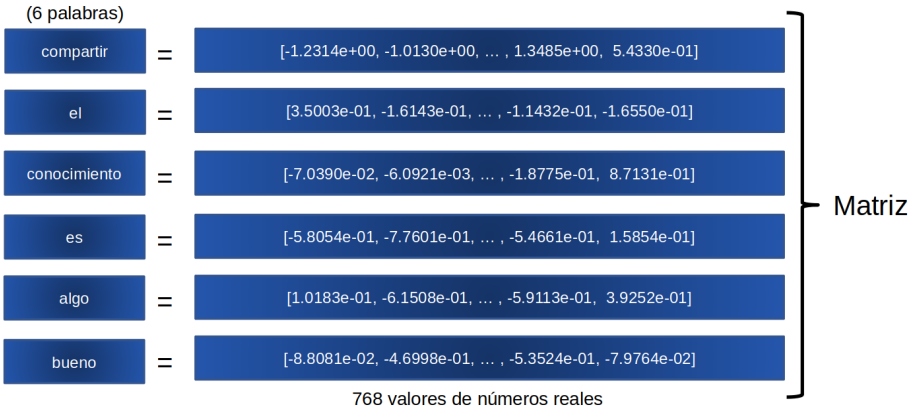


Figura 4.3: Representación obtenida a partir de un texto dado como entrada a BETO

A partir de que la matriz original pasa a tener su nueva representación, ya provocamos el efecto que nuestro individuo actual puede dar con esa distribución binaria, ahora resta conocer el rendimiento de la nueva representación del texto. Para ello vamos a pasar al segundo paso fundamental de la función de aptitud, es decir, generar la ANN a partir del número de características del

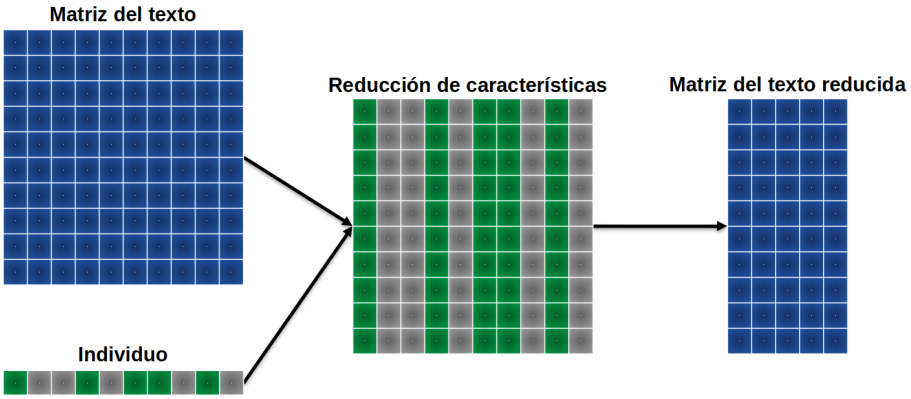


Figura 4.4: Proceso de seleccionar las características a partir de un individuo dado.

individuo y el número de tokens. En este punto hay que destacar dos aspectos, a fin de conocer el comportamiento de la propuesta desde un proceso incremental, primero se realizó una prueba de concepto, por eso la ANN inicial está basada en el trabajo de (Hern & Rodr, 2021), donde demuestran que una ANN de tan solo 2 capas puede tener un buen rendimiento en la clasificación de texto. Por otro lado, la ANN utilizada después, tiene sus bases en los trabajos de (Collobert *et al.*, 2011) y (Kim, 2014), ellos dan la pauta para incorporar el uso de una convolución antes de la ANN en la misma tarea de clasificación de texto.

Ahora solo resta el último paso para obtener la aptitud de los individuos, el entrenamiento y evaluación de la red, con ello se obtiene el valor de la presión de los datos de evaluación. Junto con ese valor y el número de características de cada individuo, se calcula el valor de aptitud, la Fórmula 4.1 muestra a detalle cómo se hace la operación.

$$aptitud = W_1 * C(x) + (W_2 * \frac{|X| - R(x)}{|X|}) \tag{4.1}$$

Donde W_1 y W_2 son valores entre 0 y 1, además la suma de ellos debe de dar 1, para el caso de W_1 es el peso de importancia al valor de la exactitud y para W_2 es el peso de importancia con respecto al número de características seleccionadas, $C(x)$ representa la exactitud obtenida con las características de $R(x)$, por último, $|X|$ es el número total de características (768). Todo el proceso de la función de aptitud

se puede ver en el Algoritmo 2. Aquí también es importante destacar que en la prueba de concepto se buscaba reducir el número características, por eso la función de aptitud tenía esos pesos de importancia, pero en las demás pruebas del GA la función de aptitud solo radicaba en la exactitud obtenida por cada individuo, es decir, la búsqueda de la nueva representación se volvió completamente libre de elegir el número de características dependiendo solo de su rendimiento. Por último, también se debe mencionar el uso de la técnica de parada temprana, mejor conocida en el inglés como early stopping, la cual nos permite evaluar el rendimiento del clasificador en cada época con un conjunto de datos diferente al de entrenamiento y prueba, esto tiene el propósito de guardar la mejor configuración del clasificador sin importar el número total de épocas.

Algoritmo 2: Función de aptitud

Datos: $epocas, individuo, datos_e, datos_v, datos_p, lr, BETO$
Resultado: $precision_p, numeroCaracteristicas$
$clasificador, fError, fOptimizador \leftarrow ClaseRN(individuo, lr, BETO)$;
para $e \in epocas$ **hacer**
  para $sentencia, clase \in datos_e$ **hacer**
    $matrizTexto \leftarrow BETO(texto)$;
    $matrizTextoReducida \leftarrow featureSelection(matrizTexto)$;
    $salida \leftarrow clasificador(matrizTextoReducida)$;
    $error \leftarrow fError(salida, clase)$;
    $\emptyset \leftarrow fActualizar(error)$;
  fin
  $precision_v, mejor_clasificador \leftarrow evaluar(clasificador, datos_v, precision_v)$;
  $numeroCaracteristicas \leftarrow individuo$;
fin
$precision_p \leftarrow evaluar(mejor_clasificador, datos_p)$;

El proceso de evaluación lleva su tiempo, pero una vez finalizado se puede analizar la calidad de la población, y con ello también se puede continuar con la selección de los padres, en este caso, se está trabajando con una selección mediante un torneo determinista dado que su funcionamiento permite mantener la diversidad en la población, evitando una convergencia prematura del GA. Dicho torneo consiste en seleccionar aleatoriamente un número t de toda la población (en muchos casos de la literatura $t = 2$) de individuos, una vez que se tienen seleccionados a los 2 individuos se elige al que tenga la aptitud más alta, para entrar en un proceso iterativo, es decir, se vuelven a seleccionar 2 individuos de toda la población y

otra vez se selecciona al mejor de los 2, todo esto hasta llegar al número de padres que se haya especificado.

Los pasos siguientes del GA constan de tomar a los padres y generar descendientes, así mismo hacer variaciones en ellos, esto con el fin de obtener nuevas distribuciones binarias que puedan mejorar la representación del texto, a través de la aptitud de los individuos. A estos pasos se les conoce como **operador de cruza** y **operador de mutación**; para el primer caso tenemos un tipo de cruza AND, utilizada en el trabajo de (Antony Gnana Singh *et al.*, 2016), donde los resultados muestran que este operador tiende a reducir el número de $1s$ en los descendientes. Esto sucede por la condición propia del operador AND, la cual básicamente consiste en que ambos padres tengan en la misma posición un 1, y solo así la característica de ambos padres es heredada al descendiente, en el caso contario, el descendiente mantiene su número de características en 0. De hecho, por la naturaleza del operador, es habitual generar descendientes con 0 características seleccionadas, en esos casos se genera un descendiente binario de manera aleatoria. El proceso del operador de cruza se puede observar en la Figura 4.5.

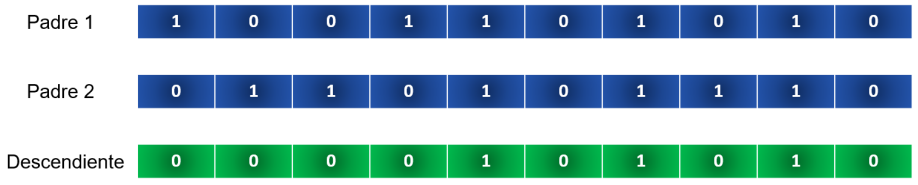


Figura 4.5: Proceso del operador de cruza AND

El operador restante es el de mutación, este puede jugar un papel más pequeño a la hora de generar variaciones en los descendientes, y para el caso de este trabajo es importante destacar que la mutación utilizada es una *mutación simple modificada*. A diferencia de la mutación simple tradicional, donde en una posición aleatoria se cambia el valor de la celda por un 1 si había un 0, o un 0 si había un 1. En el caso de la *mutación simple modificada* se garantiza cambiar aleatoriamente la posición de un 1 a una celda igualmente aleatoria donde exista un 0, con esto se logra tener una mayor exploración dentro del espacio del problema. Es decir, podemos ir cambiando la

posición de una característica de manera aleatoria y conocer si esas pequeñas variaciones al cambiar una sola característica de posición pueden alterar lo suficiente el rendimiento de un individuo. Así como en el caso del operador de cruza AND, se puede ver gráficamente cual es el proceso de la *mutación simple modificada* en la Figura 4.6.

Figura 4.6: Proceso del operador de mutación simple modificada

Ahora que se han aplicado diversos métodos para generar nuevos individuos a partir de la población inicial, se debe de someter a los nuevos individuos al mismo proceso de evaluación para conocer si el rendimiento ha mejorado, o en el peor caso, ha decaído. Después de obtener las aptitudes de los descendientes se puede hacer una comparación entre las aptitudes de la población inicial y los descendientes, esto para determinar cuáles individuos van a prevalecer en la siguiente generación. Lo anterior se conoce como reemplazo y es el paso restante para concluir con un solo ciclo del GA, dado que se busca entrar en un proceso de búsqueda, las diferentes técnicas descritas previamente se deben de repetir una g cantidad de veces, algo nombrado como generaciones.

Para llevar a cabo el reemplazo, por un lado, se puede realizar el reemplazo generacional aplicando elitismo, así se conserva al mejor individuo de toda la población generación tras generación, pero se reemplaza a casi toda la población anterior por los descendientes. La otra medida consiste en un reemplazo *proporcional*, es decir, se sustituye solo una fracción de los peores individuos de toda la población por los mejores descendientes. Esto mantiene la diversidad en el espacio del problema y también garantiza preservar a los mejores individuos de la población y los descendientes. El proceso de un reemplazo *proporcional* se puede ver en la Figura 4.7, el cual es un ejemplo donde hay 10 individuos en la población inicial y 4 en la población de descendientes, ambas poblaciones se unen en

una sola para ser ordenadas del mejor al peor con respecto a su aptitud, seguido de ello se elimina una cantidad n de los peores individuos hasta mantener el mismo número de individuos que la población inicial.

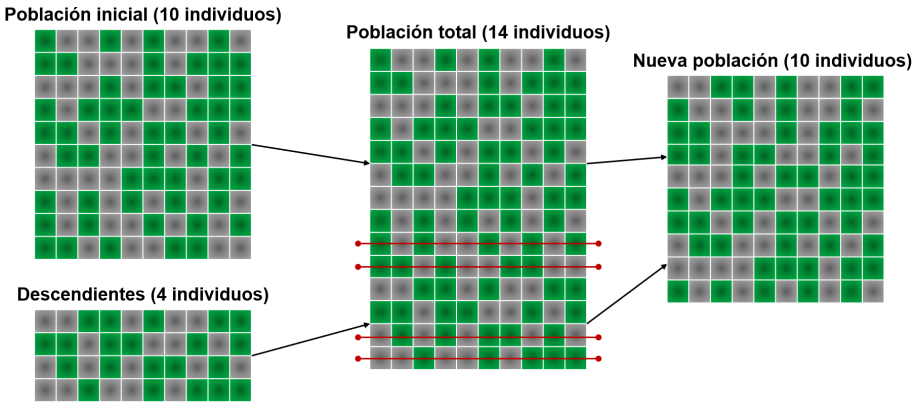


Figura 4.7: Proceso del reemplazo *proporcional*

Todo el proceso anteriormente descrito, es en esencia el proceso central para la búsqueda de una nueva representación del texto, ahora solo resta iterar cada uno de los pasos (menos el de generar la población inicial) una cantidad g de generaciones, cuyo número se puede determinar a partir de la experimentación, dado que existe un momento en donde el número de ciclos cumplidos por parte del GA sigue en aumento, pero ya no se ha encontrado un mejor individuo que genere una mejor aptitud. El Algoritmo 3 describe todo el proceso de los párrafos anteriores. Es importante mencionar que la $Formula_1$ dentro del Algoritmo 3 solo existe en la prueba de concepto, y para las demás pruebas, la aptitud que se considera es solo la variable $precision_p$.

La metodología propuesta hasta ahora solo describe el punto medular para realizar la búsqueda de la mejor representación, pero el trabajo también tiene como objetivo hacer un aprendizaje de pocas oportunidades, mejor conocido en el inglés como *few-shot learning*, y así comprobar su eficacia a la hora de realizar la búsqueda y obtener las nuevas representaciones. Para esto es importante empezar a conocer los conjuntos de datos utilizados en este trabajo, por esa razón, la siguiente sección se encarga de abordar los diferentes conjuntos, cada uno con su dominio en específico, y sus diferentes números de clases, aunque todos comparten la misma tarea de

Algoritmo 3: Algoritmo genético

Datos: $nIndividuos, nTokens, Pc, Pm, epo, gen, t, nPadres, lr, datos_e, datos_v, datos_p, BETO$
Resultado: $MejorIndividuo$
$poblacion \leftarrow generarPoblacion(nIndividuos)$;
$aptitudes \leftarrow []$;
para $individuo \in poblacion$ **hacer**
  $precision_p, nCaracteristicas \leftarrow$
   $fAptitud(epo, individuo, datos_e, datos_v, datos_p, lr, BETO)$;
  $aptitud \leftarrow Formula_1(precision_p, nCaracteristicas)$;
  $aptitudes.append(aptitud)$
fin
$i \leftarrow 1$;
mientras $i \leq g$ **hacer**
  $padresSeleccionados \leftarrow torneoDeterminista(aptitudes, t, nPadres)$;
  $descendientes \leftarrow cruza(poblacion, padresSeleccionados, Pc)$;
  $descendientesMutados \leftarrow mutacion(descendientes, Pm)$;
  $aptitudesDescendientes \leftarrow []$;
  para $descendiente \in descendientesMutados$ **hacer**
    $precision_p, nCaracteristicas \leftarrow$
     $fAptitud(epo, descendiente, datos_e, datos_v, datos_p, lr, BETO)$;
    $aptitud \leftarrow Formula_1(precisin_p, nCaracteristicas)$;
    $aptitudesDescendientes.append(aptitud)$;
  fin
  $poblacion, aptitudes \leftarrow$
   $reemplazoPro(poblacion, aptitudes, descendientesMutados, aptitudesDescendientes)$
   $i \leftarrow i + 1$;
fin
$MejorIndividuo \leftarrow mejor(poblacion, aptitudes)$;

clasificar la polaridad de un texto.

4.2. Datos

Se encontraron un total de 5 conjuntos de datos en español, de los cuales, 4 de ellos son nativos y 1 es traducido del inglés al español a través de una biblioteca de Python llamada *googletrans*. Para la prueba de concepto el conjunto utilizado corresponde a la contribución de (Fernandez, 2021), del portal de *Kaggle*, donde se trabaja en temas relacionados con la Inteligencia Artificial, AI, por sus siglas en inglés, donde se pueden generar competencias entre sus integrantes, proyectos, trabajar en la misma plataforma, entre otras cosas, también se permite subir y compartir conjuntos de datos. Los datos consisten en $50,000$ reseñas traducidas al español acerca de

películas, el género de la película que se está criticando es variado, y la cantidad de palabras por texto también, es destacable mencionar que los datos están bien equilibrados, existen $25,000$ reseñas por cada clase. Las reseñas en su versión original están en inglés, y la contribución en este caso es de los autores (Maas *et al.*, 2011).

Es indispensable definir un número máximo de palabras, dado que es un parámetro necesario a la hora de trabajar con BETO, esto porque se deben restringir todos los textos al mismo número de palabras. Aquellos textos que contengan un menor número de palabras con respecto al parámetro establecido, serán rellenados con una etiqueta especial ("[PAD]"). Por esa razón es importante encontrar el rango óptimo, y para ello se realizó una revisión de los datos, primero se dividieron todas las reseñas entre las dos polaridades, esto con el fin de encontrar el rango donde la cantidad de palabras entre ambas polaridades era más frecuente, en la Figura 4.8(a) y 4.8(b) podemos observar los resultados obtenidos.

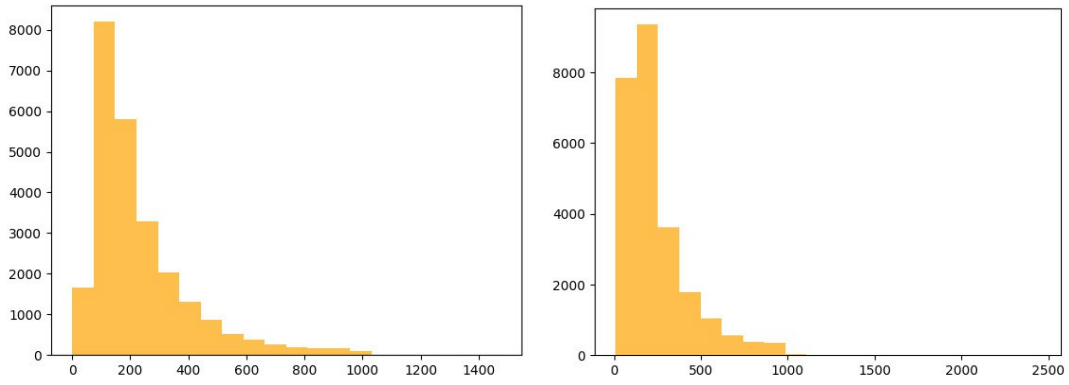


Figura 4.8: Frecuencia de las palabras en las reseñas negativas y postivas

Ambos histogramas tienen en su eje x el número de palabras que hay por todos los textos, y para el eje y corresponde a la frecuencia con la que existe ese número de palabras en cada sentencia. El histograma del inciso (a) pertenece a las palabras de los textos negativos, por otro lado, para el caso de la derecha son los textos de las palabras positivas. Considerando un aproximado, el mayor número de reseñas tienen entre 200 y 400 palabras, algo que nos permite hacer un filtro del conjunto de datos original. El riesgo inevitable al no

definir un máximo de tokens reside en generar entradas con demasiada información inútil, en este caso para el conjunto de datos actual hay reseñas que incluso llegan a tener 2500 palabras, por lo tanto, si se define por ejemplo, un máximo de palabras tal como $1,400$ para todas las reseñas, eso indica que sin importar cual sea el número de palabras de las demás reseñas todas deberán tener la misma longitud $1,400$, y como se puede observar en los histogramas hay reseñas con menos de 50 palabras, esto quiere decir que habrá un número suficientemente alto de reseñas con pura información de relleno. Retomando el rango de 200 a 400 palabras, el conjunto de datos se redujo a $14,000$ reseñas, manteniendo el equilibrio entre ambas clases, es decir, son $7,000$ reseñas de cada clase. A su vez para la experimentación de la prueba de concepto solo se utilizaron 100 reseñas, 70 para entrenamiento y 30 para prueba, solo para este caso no se hizo la implementación del early stopping.

Encontrar datos nativos del español es todo un reto, algo que los autores (Salgueiro *et al.*, 2022), mencionan como una importante ausencia en datos que sean nativos del español, específicamente para la tarea de Análisis de Sentimientos Dirigido. Por ello mismo brindan un conjunto de datos, lo cuales provienen de las elecciones presidenciales de Argentina en el 2019, los datos son encabezados de noticias de política, recolectados en ese año entre el 21 de septiembre y 27 de octubre. La tarea recibe el término de "Dirigido" porque están agregando más información a la entrada común en los modelos del lenguaje, es decir, no solo están proporcionando un texto de entrada, si no también un objetivo, dicho valor agregado es en esencia un par de palabras clave. Para etiquetar y agregar el objetivo a cada encabezado, se utilizaron 3 anotadores, además de un esquema de mayoría de votos para validar las etiquetas. La descripción detallada del número de datos por cada conjunto (entrenamiento, validación y prueba) se puede ver en la Tabla 4.1.

También se encontró un conjunto de datos para la clasificación de la polaridad con respecto a artículos de Amazon. Este conjunto es proporcionado por (Keung *et al.*, 2020) y como se puede observar en la Tabla 4.2, hay una considerable cantidad de datos para

Datos	Clase 1	Clase 2	Clase 3
Entrenamiento $(1,371)$	389	434	547
Pruebas (609)	202	166	241
Validación (459)	119	166	174

Tabla 4.1: Datos de política

entrenamiento, validación y prueba, además los datos están equilibrados en cada una de sus respectivas clases. Las reseñas fueron recopiladas desde el 2015 hasta el 2019, y no solo cuentan con una recopilación para el idioma español, también hicieron su respectiva recolección para el inglés, japones, alemán y francés. Existen varios puntos a destacar con respecto a la obtención y procesamiento de estos datos, algunos de ellos se enlistan a continuación:

1. Se utilizó un algoritmo para la detección del lenguaje en cada sentencia, así que solo el lenguaje objetivo era el seleccionado respectivamente para cada conjunto de datos
2. Se aplicó un filtro basado en el vocabulario, esto es, si había una sentencia con un token que no aparecía en al menos otras 20 reseñas, entonces la reseña no era seleccionada
3. Todas las reseñas se cortaron en un máximo de 2,000 caracteres, además los caracteres especiales como los saltos de línea y los tabuladores también se removieron de las reseñas

Datos	Clase 1	Clase 2	Clase 3	Clase 4	Clase 5
Entrenamiento $(200,000)$	$40,000$	$40,000$	$40,000$	$40,000$	$40,000$
Pruebas $(5,000)$	$1,000$	$1,000$	$1,000$	$1,000$	$1,000$
Validación $(5,000)$	$1,000$	$1,000$	$1,000$	$1,000$	$1,000$

Tabla 4.2: Datos de reseñas de los artículos de Amazon

El siguiente conjunto de datos es sobre el análisis de sentimientos en sentencias de Twitter, los autores de su uso son (Barbieri *et al.*, 2022), además del conjunto de datos, proporcionan un modelo nombrado, Modelo de Lenguaje Multilingüe en Twitter, o por las siglas establecidas, XLM-T. Los datos en este caso no son tan significativos como con los datos de (Keung *et al.*, 2020), pero siguen teniendo

bastante relevancia por el simple hecho de ser nativos del español, en la Tabla 4.3 se pueden ver los detalles correspondientes al número de datos para entrenamiento, validación y prueba. Los datos que proporcionan son de 8 diferentes lenguas, árabe, inglés, francés, alemán, hindi, italiano, portugués y español, aunque es importante mencionar que el origen de todos estos datos realmente son desde otras fuentes, por ejemplo, para el caso del español esos datos fueron proporcionados por primera vez en el artículo de (Martínez-Cámara *et al.*, 2018). La razón por la que se hizo una recolección de los datos de diferentes lenguas, fue para poner a prueba su modelo XLM-T, y para ello hicieron ajustes en todos los conjuntos con la finalidad de que todos tuvieran la misma cantidad de datos (entrenamiento, validación y prueba).

Datos	Clase 1	Clase 2	Clase 3
Entrenamiento $(1,839)$	613	613	613
Pruebas (870)	290	290	290
Validación (324)	108	108	108

Tabla 4.3: Datos de tweets

El último conjunto de datos recolectado ya es un conjunto sumamente conocido para la clasificación de la polaridad de un texto en el dominio de reseñas de películas, este conjunto actualmente se puede encontrar en línea a través de la plataforma de HugginFace, una plataforma cuyo enfoque se encuentra en la construcción del futuro de la IA, existen diversas funciones dentro de la plataforma, como cargar y utilizar modelos entrenados por la comunidad, documentación de las diferentes funciones, aplicaciones de ML, foros, cursos y también un apartado de diferentes conjuntos de datos, entre ellos se encuentra el de *muchocine.net*, el cual fue proporcionado por (Mata, 2017), donde hizo una recopilación de algunas reseñas y las subio a la plataforma. El número de reseñas para cada clase es bastante variado, no mantiene un equilibrio, al contrario, se puede observar bastante discrepancia entre un número para una clase y otra, pero sí es un conjunto de datos nativo del español, los detalles de lo anterior se pueden observar en la Tabla 4.4

Datos	Clase 1	Clase 2	Clase 3	Clase 4	Clase 5
Total $(3,872)$	351	922	1249	889	461

Tabla 4.4: Datos de reseñas de películas de la plataforma muchocine

4.2.1. Limpieza de los datos

La limpieza del texto es indispensable para cualquier proceso del NLP, pero no existe un estándar. Para algunas técnicas puede ser necesario retirar algunos componentes ruidoso del texto, pero para otras ese ruido realmente puede ser información valiosa, por ello lo primero es analizar cuál es la tarea y la técnica con la que se pretende abordar el problema. Dado el enfoque de este trabajo, a continuación se enlistan los diferentes procesos de limpieza aplicados a los conjuntos de datos que se describieron en la sección anterior:

1. Eliminar los espacios repetidos que puedan existir antes, entre y después de un texto
2. Convertir todo el texto a minúsculas
3. Eliminar los emojis
4. Si es el caso, convertir los "@algún nombre de usuario" por "@usuario"
5. Reemplazar cualquier URL por el término "url"
6. Separar entre espacios los carácteres especiales

El vocabulario de BETO además de considerar palabras y fragmentos de palabras, también tiene contemplado ciertos signos de puntuación, es decir, algunos signos de puntuación tienen su propia representación vectorial, por eso no se están eliminando y solo se está aplicando un método para dejar espacio entre las palabras que los acompañan. Por otro lado, los emojis se eliminan porque no se encuentran dentro del vocabulario de BETO, por lo tanto, solo reciben la etiqueta especial de "[UNK]". Siguiendo con la justificación de la limpieza, los acentos de las palabras no se eliminan porque sí se reconocen y están dentro del vocabulario, además se está aplicando una normalización a minúsculas por la versión de BETO utilizada en este trabajo.

Por último, las palabras vacías (stopwords) para técnicas como BoW o n-gramas, pueden significar solo información sin relevancia, pero a la hora de trabajar con BETO sucede todo lo contrario, incluso esas palabras pueden brindar cierta información que haga entrenar mejor al modelo para su tarea. Esto sucede por la naturaleza propia de BETO, que es un modelo entrenado para dar representación al texto según sea el contexto de la oración completa, por ello hasta las palabras vacías reciben información importante que influyen por completo en el ajuste apropiado del modelo.

En la próxima sección se describirán los experimentos realizados, en donde se podrá observar y analizar a detalle como fue el comportamiento de cada uno de los resultados, desde el tamaño de la población, los diferentes conjuntos de datos y sus variaciones, la aplicación y rendimiento del *few-shot learning*. Incluso hay una comparación entre el rendimiento de las dos representaciones propias de BETO y algunas técnicas de DL, pero sobre todo, los resultados para valorar el rendimiento de las nuevas representaciones en diferentes conjuntos de datos.

5. Experimentos y resultados

Durante el transcurso de esta sección se mostrará el escalamiento de un experimento con la configuración mínima, hasta llegar a la máxima con respecto a los recursos técnicos de este trabajo de investigación. El experimento de la sección 5.1) consiste de una prueba de concepto, esto con el fin de conocer el comportamiento del GA en la búsqueda de una nueva representación. El experimento de la sección 5.2) tiene el fin de mostrar cuál representación a nivel vectorial o matricial tiene un mejor rendimiento, y con ello continuar con el escalamiento del GA. Para el caso del experimento en la sección 5.3) se hace la búsqueda más amplia para la obtención de las nuevas representaciones, esto utilizando la representación más adecuada del caso anterior, junto con el enfoque de *few-shot learning*. Por último, la sección 5.4) tiene la finalidad de mostrar el rendimiento de las nuevas representaciones obtenidas con respecto a la representación original entre diferentes conjunto de datos.

5.1. Prueba de concepto

El primer experimento busca ser en esencia el punto de partida para la trayectoria de los demás, por lo tanto, se están utilizando los datos traducidos al español descritos en la sección 4.2. Además la ANN dentro de la función de aptitud es la propuesta por (Hern & Rodr, 2021), la cual tiene una arquitectura simple. La siguiente lista describe la experimentación inicial:

1. Obtener los mejores porcentajes de cruza y mutación
2. Obtener la nueva representación
3. Comparar el rendimiento de la nueva representación contra la representación original

El equipo de cómputo utilizado para la experimentación inicial tiene por procesador un Intel Xeon(R) CPU E542 de 8 núcleos 2.80GHZ, una memoria RAM de 6GB, y como sistema operativo, Ubuntu versión 22.04.1 LTS. Además, el lenguaje de programación utilizado para todo el proceso fue Python versión 3.10. Es importante

mencionar que para este caso la limpieza de los datos solo se utilizó el punto 2 descrito en la sección 4.2.1. Todos los parámetros utilizados en las experimentación inicial del GA se pueden ver en la Tabla 5.1. Los resultados reflejaron el comportamiento deseado con respecto a la reducción de características, pero se debe de resaltar que el GA fue sensible a los valores de w_1 y w_2, es decir, el GA fue conducido a quedarse con los individuos con un menor número de características, aun así, la nueva representación fue mejor en comparación con el número de características original.

Parámetro	**Exp. 1**	**Exp. 2**	**Exp. 3**
Tamaño de la población	50	50	-
Número de sentencias	40	100	100
Número de generaciones	10	10	-
Épocas de entrenamiento	5	5	30
Palabras por reseña	30	50	50
w_1	0.2	0.2	-
w_2	0.8	0.8	-
Tamaño del torneo (t)	2	2	-
Número de padres	20	20	-
No. Ejecuciones	10	10	30

Tabla 5.1: Parámetros usados en cada experimento

Los resultados principales son con respecto a los mejores porcentajes de cruza y mutación. Esto se puede observar en la Tabla 5.2, donde cada fila hace referencia a una combinación que fue ejecutada 10 veces, y las 2 columnas principales hacen referencia a la combinación de los porcentajes de cruza y mutación que se utilizaron en el experimento, la columna 3 está relacionada a las características seleccionadas y se puede observar que contiene 2 subcolumnas, el promedio y la mediana, para la siguiente columna sucede la misma situación solo que con respecto a la exactitud de los datos de prueba, y en la última columna, con respecto a la aptitud.

La tabla muestra un interesante comportamiento relacionado con la reducción del número de características original en todos los diferentes porcentajes, también se obtiene una exactitud considerablemente cercana entre sí, pero si analizamos la mediana, existen 3 combinaciones que predominan ante las demás, y en el promedio de esas 3 la mejor combinación de porcentajes para los operadores de cruza y mutación, es cuando ambos son 0.4.

Porcentaje de cruza	Porcentaje de mutación	Características		Exactitud		Aptitud	
		Avg	Med	Avg	Med	Avg	Med
0.2	0.2	5.	6	0.78	0.75	0.951	0.945
0.2	0.4	4.3	6	0.80	0.79	0.955	0.954
0.2	0.6	8	6	0.82	0.83	0.956	0.960
0.2	0.8	8.8	6	0.79	0.75	0.949	0.944
0.4	0.2	4.4	5	0.80	0.79	0.955	0.954
0.4	**0.4**	5.8	6	0.82	0.83	**0.958**	0.960
0.4	0.6	4.5	4	0.75	0.75	0.945	0.944
0.4	0.8	7.5	6	0.80	0.79	0.952	0.948
0.6	0.2	3.9	4	0.76	0.75	0.949	0.946
0.6	0.4	4.5	6	0.80	0.83	0.955	0.960
0.6	0.6	4.8	4	0.77	0.75	0.950	0.946
0.6	0.8	6.2	5	0.79	0.79	0.951	0.950
0.8	0.2	10.3	7.5	0.84	0.83	0.957	0.958
0.8	0.4	7.1	3.5	0.79	0.75	0.950	0.948
0.8	0.6	4.5	3.5	0.74	0.75	0.943	0.945
0.8	0.8	3	2.5	0.73	0.75	0.943	0.946

Tabla 5.2: Resultado de 10 ejecuciones del GA para cada combinación de las probabilidades de cruza y mutación (%).

Una vez obtenidos los porcentajes de cruza y mutación, son utilizados para obtener la nueva representación del texto. El resultado del experimento se puede observar en la Tabla 5.3, en la cual también tenemos 10 ejecuciones, la primera columna hace referencia al mejor número de características encontrado, la segunda columna a la exactitud que se obtuvo con ese número de características, y por último, se encuentra la columna correspondiente a la aptitud formada a partir de los datos de las primeras 2 columnas.

Características (El mejor)	Exactitud (El mejor)	Aptitud (El mejor)
6	0.70	0.933
6	0.70	0.933
6	0.70	0.933
6	0.70	0.933
6	**0.76**	**0.947**
6	0.70	0.933
6	0.70	0.933
6	0.70	0.933
6	0.70	0.933
6	0.70	0.933

Tabla 5.3: Resultado de 10 ejecuciones del GA para reducir el número de características.

En todas las ejecuciones, las nuevas representaciones obtenidas coinciden en el número de características seleccionadas 6. Tomando en cuenta el experimento restante, la ejecución 5, la que tiene una aptitud relativamente mejor a las demás, será aquella que se utilizará para realizar la comparación con la representación original. Lo

resultados se pueden observar en la Tabla 5.4. La comparación consiste en utilizar la nueva representación, contra las 768 características originales de BETO, ambas competirán en la misma tarea de clasificación.

Características	Exactitud			Longitud de los datos	Operaciones de la red	Tiempo aprox. Ejecución en minutos			Wilcoxon p-value
	Min	Avg	Max			Min	Avg	Max	
768	0.43	0.52	0.66	3,840,000	1,474,713,604	244	429	755	1.2953e-06
6	0.70	**0.70**	0.70	30,000	91,204	73	75	76	

Tabla 5.4: Resultados de la comparación entre la nueva representación y la original.

En general, los resultados muestran una diferencia que se destaca en todos los casos de comparación, es decir, la exactitud con respecto a la mínima, media y máxima de ambas características deja notar un mejor desempeño con 6 características. Lo mismo sucede tomando en cuenta el número de datos (5.1) y el número de parámetros de la red (5.2), la cantidad necesaria en ambos casos se reduce drásticamente, por lo tanto, también hay una reducción considerable en el tiempo aproximado de ejecución. Para la última comparación, dado que no se puede asumir una distribución normal entre las diferentes exactitudes de cada representación, se utilizó la prueba de *Wilcoxon rank sum*, donde el $p - value$ refleja un valor suficientemente bajo como para destacar diferencias significativas entre las dos muestras.

$$ND = f \times p \times s \tag{5.1}$$

Donde ND es el Número de datos, f es el número de características, p el número de palabras y s el número de sentencias

$$PR = ((f \times p) \times w) + (b + a) \tag{5.2}$$

Donde PR es Parámetros de la red, f y p son las mismas variables que en la ecuación anterior, w es el número total de pesos, b el número total de sesgos y a el número total de funciones de activación. Todos los resultados obtenidos en esta sección están reportados en un artículo que lleva como título, "Selección de características de representaciones de texto de BETO usando un algoritmo genético" publicado en la revista *Research in Computing Science. Volumen 152. Número 7*, (Guzmán-Landa *et al.*, 2023).

Estos resultados de la prueba de concepto nos permiten observar varios puntos importantes, la nueva representación no causó un disminución en el rendimiento de la exactitud para la tarea de clasificación, al contrario, el beneficio se disparó en dos direcciones, aumentó el rendimiento y se redujo el número de parámetros necesarios. Por lo anterior, se puede empezar a intuir que las características originales no son necesariamente indispensables en su totalidad, pero para poder afirmarlo se debe explorar más a fondo la forma en la que se puede obtener la representación, es decir, una cantidad mayor de datos.

5.2. Comparación entre la representación Vectorial y Matricial de BETO

El modelo de lenguaje BETO debe de recibir una entrada a partir de alguna sentencia, esto devuelve dos cosas, una matriz y un vector (El vector es conocido como el token [CLS]). La matriz es toda la información de la sentencia, ya que cada fila es correspondiente a un token de la sentencia y cada columna corresponde a las características propias de cada token. Por otro lado, el token [CSL] está encapsulando toda la información de la sentencia en un solo vector, su longitud es de la misma a cualquier otro token. En ambos casos se puede hacer una selección de características, es decir, tanto en el vector cómo en la matriz se puede conocer si existen características más relevantes o no. Por la razón anterior, la siguiente serie de experimentos van a comprobar el rendimiento de una representación contra otra, además se hará el uso de tres técnicas diferentes de DL para comprobar en conjunto cuál combinación entre técnica y representación demuestran tener un mejor rendimiento.

Los datos utilizados para realizar los experimentos son los de Amazon, descritos en la Sección de Datos 4.2, esto porque son aquellos con mayor cantidad de sentencias, por lo tanto, son los datos con mayor relevancia a diferencia de los otros. Es importante

mencionar que se hicieron tres filtros de los datos de Amazon, los cuales están descritos en la siguiente lista:

1. $(66,495)$: Todas aquellas sentencias con mas de 30 tokens
2. $(104,090)$: Todas aquellas sentencias con menos o igual a 30 tokens
3. $(75,850)$ y $(76,150)$: Todas aquellas sentencias con la mayor similitud del coseno entre clases

Los tres nuevos conjuntos después de aplicar el filtro fueron equilibrados para mantener el mismo número de sentencias entre clases. Se definieron 30 tokens dado que ahí se concentra una gran cantidad de sentencias con ese número de tokens, por eso se generaron dos conjuntos, uno con 30 tokens o menos, y su contraparte con más de 30 tokens. Para el otro filtro se utilizó la medida de la similitud del coseno y la representación vectorial del token [CLS], con ello se fue trabajando con todas las sentencias por clase $(40,000)$. El proceso consiste en un ciclo dentro de otro ciclo, esto con el fin de calcular la similitud del coseno de cada sentencia contra las demás sentencias de su clase, y de esta manera ver cuales sentencias tienen una mayor similitud con respecto a las demás. De esta manera se obtiene una medida entre -1 y 1 la cual nos permite filtrar a las sentencias con la similitud más baja y solo guardar a las de mayor similitud. Lo anterior se hizo en bloques de $10,000$ y $5,000$ sentencias, por lo tanto, son 4 y 8 veces por clase.

Para poner a prueba el rendimiento de los nuevos conjuntos, se realizo una experimentación, para conocer cuál conjunto de datos había mantenido mayor información relevante. Los resultados se pueden ver en la Tabla 5.5.

Los resultados muestran un mayor rendimiento con el conjunto de datos *similitud coseno v2*, por lo tanto, esos serán los utilizados para la experimentación siguiente. El propósito de dicha experimentación como ya se había mencionado previamente, es para conocer cuál combinación entre representación y técnica tienen un mejor rendimiento. De esa forma a la mejor representación se le aplicará la selección de características para obtener la nueva representación. A

No. Ejecución	Exactitud ($> 30tokens$)	Exactitud ($<= 30tokens$)	Exactitud (similitud coseno)	Exactitud (similitud coseno v2)
1	0.41759998	0.50940001	0.53759998	0.54219997
2	0.47139999	0.52279997	0.53780001	0.53380001
3	0.46559998	0.5244	0.5266	0.54159999
4	0.42379999	0.52340001	0.52700001	0.52639997
5	0.45339999	0.52160001	0.54159999	0.53240001
6	0.44419998	0.50800002	0.53219998	0.53920001
7	0.47139999	0.5108	0.53839999	0.54280001
8	0.46199998	0.50919998	0.53560001	0.53600001
9	0.461	0.51099998	0.53780001	0.54140002
10	0.46399999	0.50879997	0.53600001	0.53319997
Promedio	0.45343999	0.51494	0.53506	**0.5369**

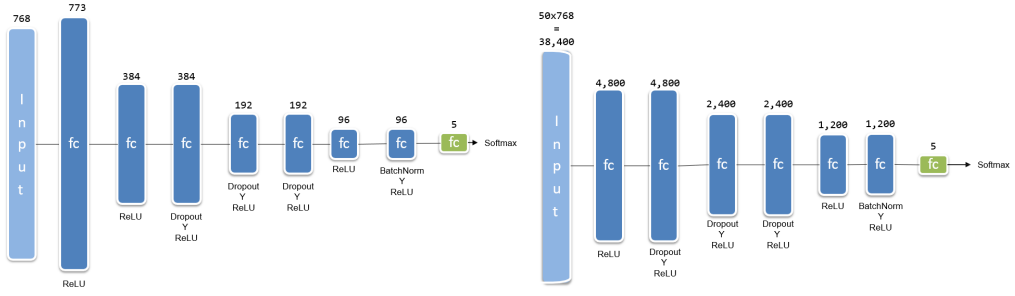
Tabla 5.5: Resultados utilizando los diferentes conjuntos filtrados de Amazon.

fin de mostrar las diferentes combinaciones de la experimentación, se puede observar el diseño experimental en la Tabla 5.6.

Representación	Técnica
Vectorial	SH-NN
Matricial	SH-NN
Vectorial	S-CNN
Matricial	S-CNN
Vectorial	P-CNN
Matricial	P-CNN

Tabla 5.6: Representaciones y técnicas utilizadas en la experimentación

Cada representación y técnica tienen una figura asociada para conocer a detalle cómo fue la arquitectura de cada modelo y con ello poder entender las ventajas y desventajas. Para el caso de la combinación *(SH-NN con Vectorial)* y *(SH-NN con Matricial)* está la Figura 5.1(a) y 5.1(b). En el primer caso la combinación viene con su representación vectorial, por lo tanto, la entrada solo es el token [CLS] y para la representación matricial se puede apreciar una entrada mucho mayor, por ejemplo, con 50 tokens la entrada se vuelve de la dimensión 768×50.

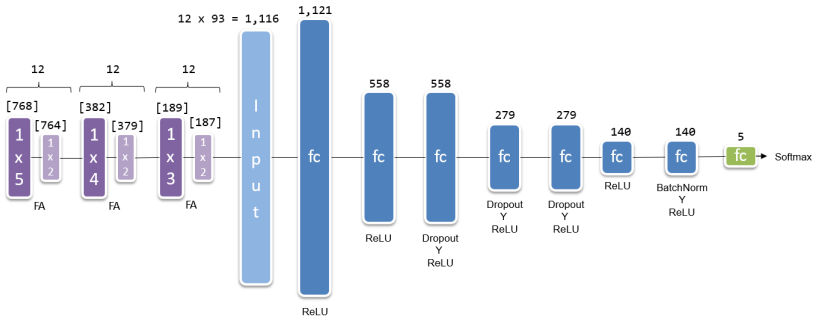


(a) (b)

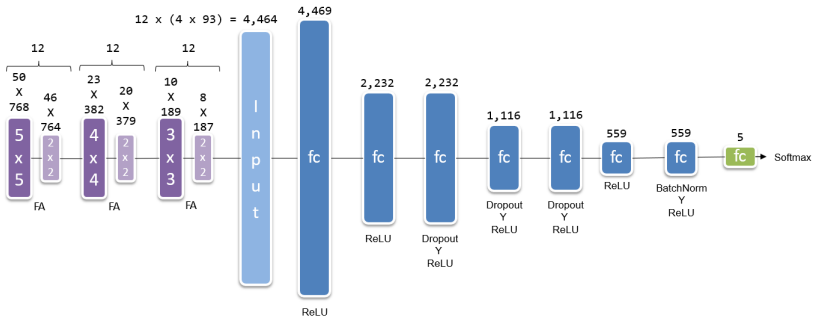
Figura 5.1: Combinación de representación y técnica de SH-NN

Considerando la siguiente combinación, *(S-CNN con Vectorial)* y

(S-CNN con Matricial) está la Figura 5.2(a) y 5.2(b), donde se puede observar la forma habitual en la que se hace una convolución antes de la capa fully-connected. Al igual que en el caso anterior se ve la diferencia entre usar una y otra representación, por un lado, con un solo vector la extracción de características en la convolución se vuelve bastante compacta, sin embargo, en los resultados expresados más adelante, se puede observar un rendimiento favorable. Para el otro caso, la extracción de características beneficia a generar una entrada más compacta dado que se puede aplicar una convolución mayor.



(a)

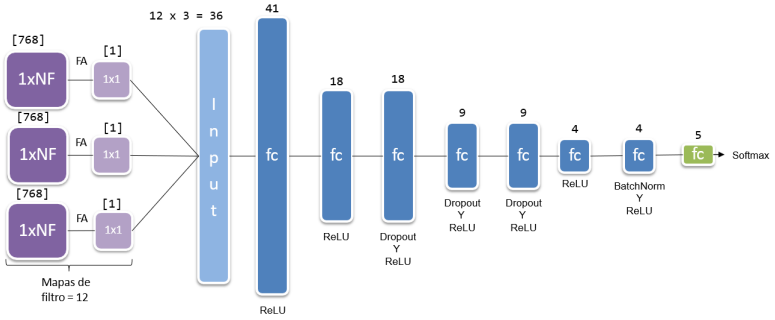


(b)

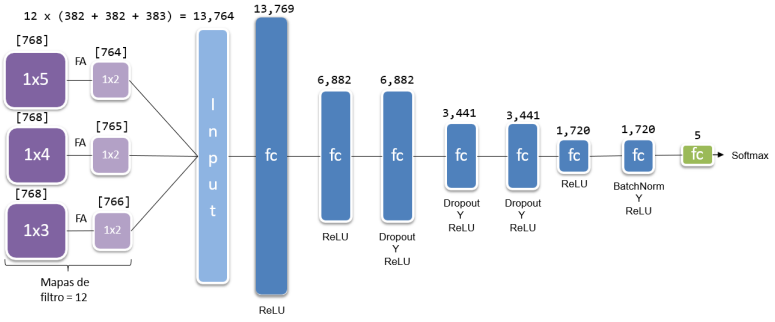
Figura 5.2: Combinación de representación y técnica de S-CNN

La última combinación, *(P-CNN con Vectorial)* y *(P-CNN con Matricial)*, tiene una variante en la representación vectorial, dado que la convolución se puede aplicar en dos sentidos, como lo propone (Kim, 2014) haciendo el kernel del mismo tamaño que el total de características, por otro lado, utilizando los kernels como en los ejemplos anteriores. Se puede observar a detalle la variante de la representación vectorial, y la combinación normal con la matricial en la Figura 5.3(a), 5.3(b) y 5.3(c). A diferencia de la convolución secuencial, en la paralela la convolución se aplica por separado a la misma entrada, después se concatenan los diferentes mapas de

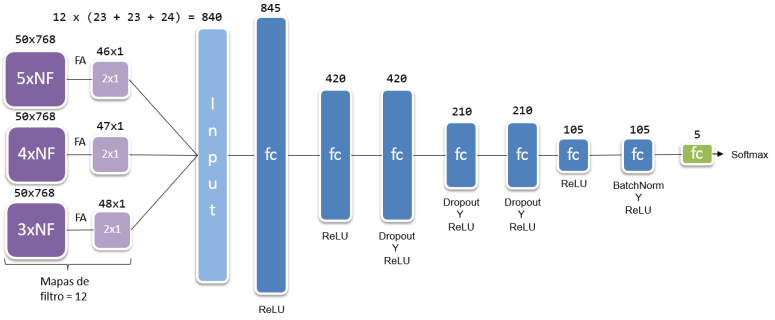
características y se obtiene la entrada del modelo.



(a)



(b)



(c)

Figura 5.3: Combinación de representación y técnica de P-CNN

Ahora que se han expuesto todas las arquitecturas utilizadas en este diseño experimental, Los resultados se pueden observar en la Tabla 5.7.

Los resultados muestran que la representación *vectorial* con la técnica de *S-CNN* tienen un mejor rendimiento a comparación de las demás, esto indica que debería de ser la vía para la selección de características y obtener la nueva representación. Pero en las primeras ejecuciones del experimento se encontraron ciertos inconvenientes, por ejemplo, no se pueden crear individuos con un número de características menor a 22 porque las capas de convolución terminan generando una entrada no valida. Además de

No. Ejecución	SH-NN (Vectorial)	SH-NN (Matricial)	S-CNN (Vectorial)	S-CNN (Matricial)	P-CNN (Vectorial v1)	P-CNN (Vectorial v2)	P-CNN (Matricial)
1	0.5530	0.5546	0.5596	0.5526	0.2000	0.5488	0.5520
2	0.5592	0.5506	0.5592	0.5488	0.4370	0.5538	0.5576
3	0.5530	0.5574	0.5592	0.5470	0.2000	0.5502	0.5544
4	0.5482	0.5542	0.5536	0.5522	0.2000	0.5562	0.5636
5	0.5544	0.5646	0.5618	0.5534	0.2620	0.5530	0.5486
6	0.5588	0.5468	0.5510	0.5566	0.3282	0.5488	0.5558
7	0.5508	0.5412	0.5540	0.5574	0.2000	0.5600	0.5554
8	0.5602	0.5472	0.5570	0.5524	0.3796	0.5524	0.5590
9	0.5530	0.5492	0.5624	0.5560	0.3210	0.5398	0.5554
10	0.5508	0.5542	0.5624	0.5512	0.3278	0.5504	0.5612
Promedio	0.5541	0.5520	**0.5580**	0.5528	0.2856	5513	**0.5563**
Tiempo Aprox. 1 ejecución	38 minutos	65 minutos	38 minutos	44 minutos	15 minutos	85 minutos	38 minutos

Tabla 5.7: Resultados de las diferentes combinaciones entre representación y ténica de DL

eso la representación vectorial como ya se había mencionado previamente, es un vector el cual está representando a toda una sentencia, algo en principio bastante útil a la hora de hacer un *fine-tuning* para una tarea en específico, pero para el caso que nos compete en este trabajo lo ideal es trabajar con todos los tokens propios de una sentencia. Todo lo anterior nos permite sustentar porque se está considerando la $2da$ mejor combinación de los resultados, por lo tanto, la búsqueda de las nuevas representaciones se harán bajo esa configuración, *(P-CNN con Matricial)*.

5.3. Nuevas representaciones de texto

La búsqueda de la mejor representación se realizó desde un enfoque *few-shot learning*, por eso primero se filtraron los datos de Amazon, pero ahora considerando aquellas sentencias con mayor cantidad de palabras únicas. En este caso sí se eliminaron las palabras vacías, aunque solo fue para hacer el filtrado de las sentencias y después los datos se siguen limpiando desde la misma forma como se describe en la Sección 4.2.1. Es importante mencionar que además de la condición de filtro, también se hizo otro proceso para juntar las 5 clases de Amazon en 3 clases. Al final se generaron 4 conjuntos de datos nuevos (Todos están equilibrados), los cuales se pueden ver detalladamente en la Tabla 5.8. Es importante resaltar que los nombres de las columnas hacen referencia a cada uno de los nuevos conjuntos de datos que se generaron, de esa misma forma se expresan a lo largo del trabajo de investigación.

Datos	Entrenamiento	Validación	Prueba	N. Clases
Amazon 1	500	150	150	3
Amazon 2	500	150	150	5
Amazon 3	5000	1500	1500	3
Amazon 4	5000	1500	1500	5

Tabla 5.8: Datos generados para aplicar el *few-shot learning* en la búsqueda de las nuevas representaciones

Ahora con los diferentes conjuntos de datos establecidos, solo resta mostrar los parámetros especificados para cada una de las corridas del GA, todos estos están descritos en la Tabla 5.9.

Parámetro	**Amazon 1**	**Amazon 2**	**Amazon 3**	**Amazon 4**
Tamaño de la población	100			
Número de generaciones	10			
Tamaño del torneo (t)	2			
Número de padres	20			
Porcentaje de cruza	0.4			
Porcentaje de mutación	0.4			
Número de tokens	245	245	137	137
Tamaño del lote	36			
Número de clases	3	5	3	5
Épocas	24			
Learning rate	$1e-5$			

Tabla 5.9: Parámetros usados en cada experimento

A diferencia de la prueba de concepto, la intención en este punto fue de establecer una población inicial de 100 individuos con el propósito de generar una mayor diversidad de vectores binarios para abarcar más zonas del espacio de búsqueda. También se aumentó el número de tokens por cada sentencia, en un caso tenemos 245 y en el otro de 137, ese valor se ajusta según donde se encuentre la mediana. Por último, el número de épocas se estableció haciendo una prueba para conocer hasta qué número de épocas la red ya no logra mejorar la exactitud, esto se repite 10 ejecuciones y se puede observar en la Figura 5.4

Tomando el orden de las Tablas 5.8 y 5.9, la primera nueva representación es con respecto al conjunto de datos de *Amazon 1*, es decir, con los datos de entrenamiento de (500), validación (150) y prueba (150), además de que son los datos donde se unieron las clases para solo ser 3. Los resultados se pueden ver expresados en la Tabla 5.10, donde están los mejores individuos de todas las generaciones, y por otro lado la gráfica de convergencia en la Figura

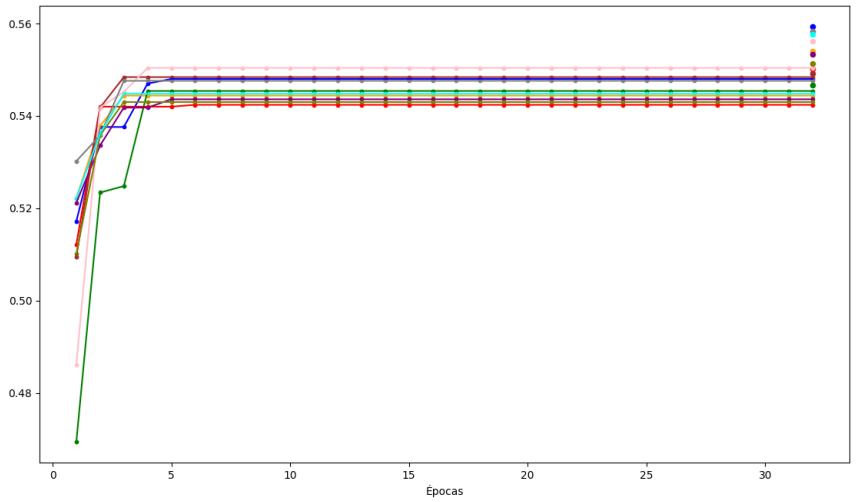


Figura 5.4: Comportamiento de la exactitud con los datos originales de Amazon

5.5. Es importante mencionar que la gráfica de convergencia muestra una estabilización desde la población inicial, es decir, el mejor individuo se encuentra entre los primeros 100 individuos generados, después en las demás generaciones solo se expande la búsqueda, pero no se encuentra a un mejor individuo.

Mejor representación	Aptitud (Exactitud)
212	0.80
212	0.80
212	0.80
212	0.80
212	0.80
212	0.80
212	0.80
212	0.80
212	0.80
212	0.80
212	0.80

Tabla 5.10: Resultados del GA con los Datos de *Amazon 1*

La nueva representación cuenta con 212 características y una exactitud del 80 %. Para poder analizar de una manera más detallada cómo fue el comportamiento del GA, se puede observar la Figura 5.6, donde el eje x hace referencia al diferente número de características de las representaciones, y el eje y a la exactitud obtenida con dichas representaciones. Se debe de resaltar que el espacio de búsqueda en la figura anterior, refleja una diversidad considerable entre los individuos generados, sin embargo, el mejor fue el individuo con 212 características.

En la gráfica se pueden observar diferentes puntos, entre ellos el punto de color verde que hace referencia al mejor individuo

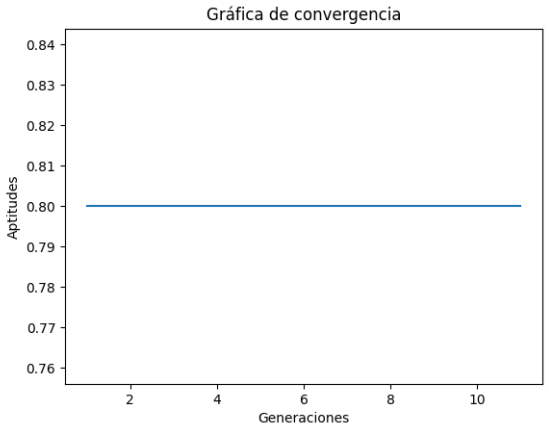


Figura 5.5: Gráfica de convergencia del GA con los datos *Amazon 1*

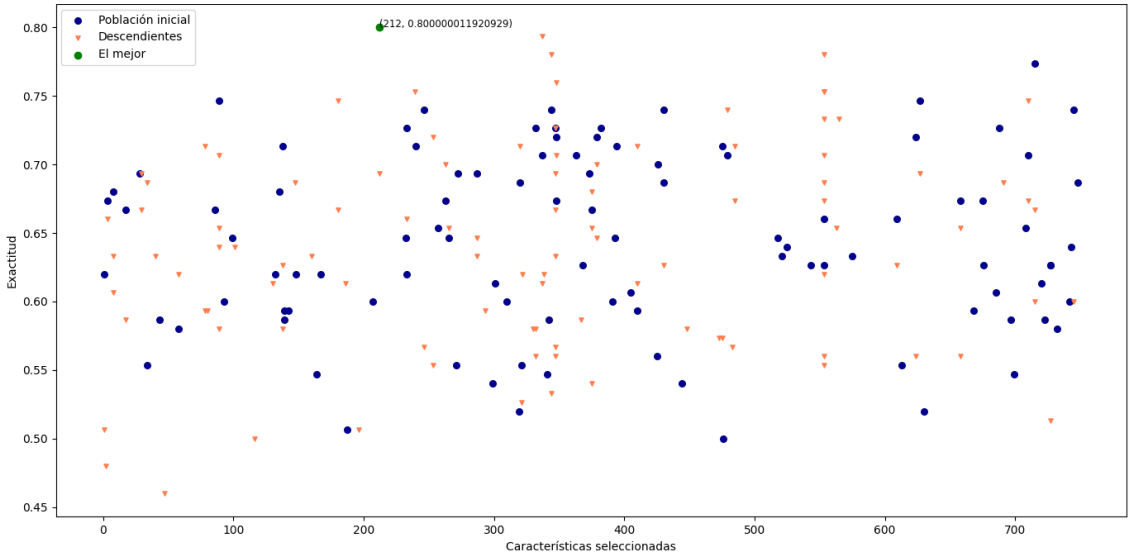


Figura 5.6: Gráfica de puntos de la población de *Amazon 1*

encontrado, incluso se puede observar el número exacto de características junto con su exactitud. Para el color azul son todos aquellos individuos generados en la población inicial, y para el color coral son todos los descendientes de todas las generaciones. Además, podemos observar claramente la ausencia de alguna relación entre la exactitud con respecto al número de características, de hecho, los mejores rendimientos no están necesariamente en el número de características más cercano a la representación original. Al contrario, en donde el número de características es menor, se puede percibir ligeramente una línea de varios puntos apilados, y para corroborar esta percepción se puede consultar la Figura 5.7. En esta otra gráfica se está reflejando una tendencia del número de características entre

335 y 360, justo en donde la gráfica de puntos 5.6 también tiene en ese rango individuos con una aptitud alta, acercándose a la del mejor. En la Figura 5.8 se observan los puntos de manera más detallada dentro del rango mencionado anteriormente, por otro lado, el punto más alto con respecto a la exactitud es el del individuo con 337 características, a su vez en el otro extremo de la gráfica, hay varios puntos que comparten el mismo número de características (347 y 348), sin embargo algunos tienen un mejor rendimiento. Aquí se tienen dos hallazgos:

1. Tienen el mismo orden, el mismo número de características, y lo único diferente es la exactitud porque son diferentes ejecuciones
2. Tienen el mismo número de características, sin embargo, las características no tienen el mismo orden de distribución y por eso se obtiene un mejor rendimiento.

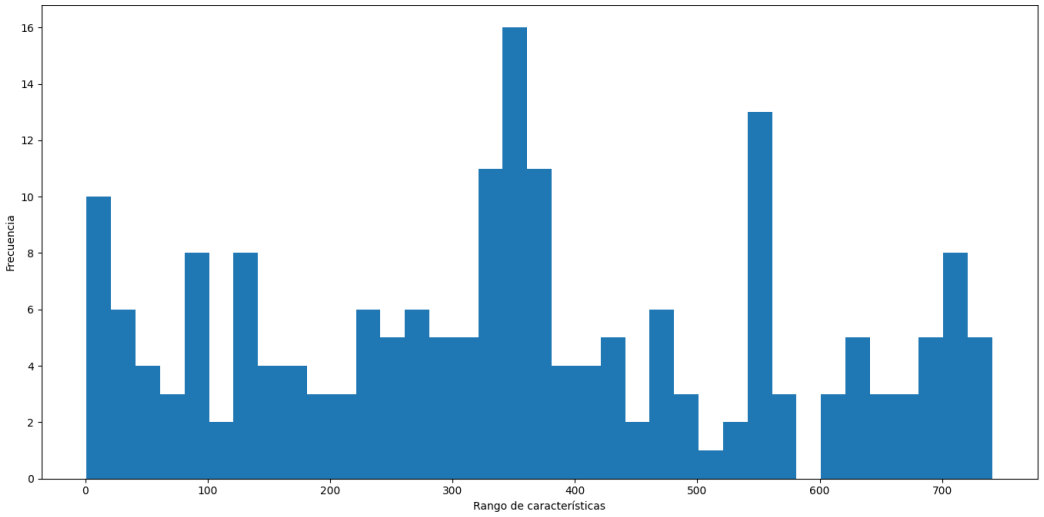


Figura 5.7: Gráfica de barras de la población de *Amazon 1*

Para verificar lo anterior, se generó una imagen con las representaciones binarias de cada uno de los puntos dentro del rango de 337 hasta 348 y el mejor individuo, en total son 21 individuos, cada vector está colocado uno debajo del otro. Solo hay dos colores, el azul refleja las características **NO** seleccionadas, y para el caso del color amarillo son las características que **SI** están siendo seleccionadas. La descripción anterior se puede observar en la Figura 5.9, debido a la naturaleza de la longitud de cada vector, una representación visual clara tiene sus complicaciones, pero sí se pueden distinguir que en algunas partes de los vectores prevalece el

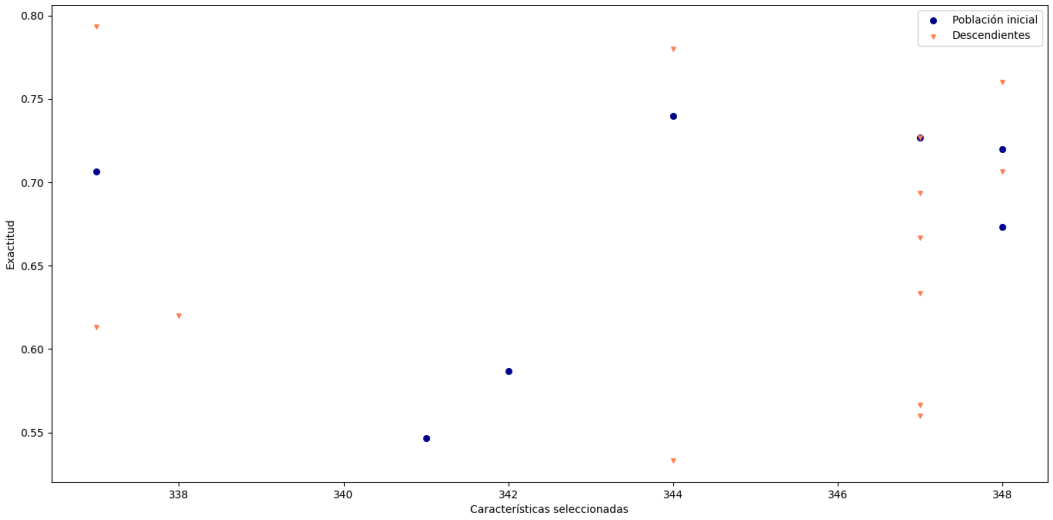


Figura 5.8: Gráfica de puntos dentro del rango 340 y 360

color amarillo entre los 21 individuos, es decir, comparten características.

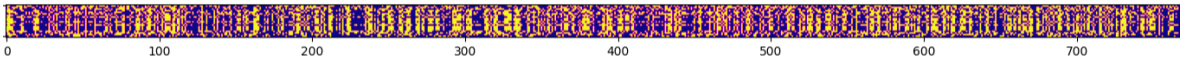


Figura 5.9: Representación de los individuos dentro del rango 337 - 348

Con el fin de tener un mayor acercamiento de la figura anterior, la Figura 5.10(a) y 5.10(b) muestra en 2 partes la representación de todos los individuos. A grandes rasgos se puede percibir un patrón de las características seleccionadas, pero también se distinguen diferencias entre los vectores, es decir, hay características que no están compartiendo la misma distribución, por lo tanto, sí hay representaciones que están teniendo una exactitud más alta por la diferencia entre las características seleccionadas. El vector de la última fila hace referencia a la mejor representación obtenida en la ejecución del GA, y se puede observar cómo ciertas características seleccionadas están tanto en el mejor individuo como en los demás, pero hay otras muchas que no. Sí es importante notar la incertidumbre existente en cuales son exactamente esas características que prevalecen entre los diferentes individuos, y para poder ir teniendo un mayor acercamiento, el siguiente experimento corresponde a *Amazon 2*.

El experimento de *Amazon 2* no aumenta el número de sentencias, pero sí el número de clases, ahora son 5, y al igual que en los

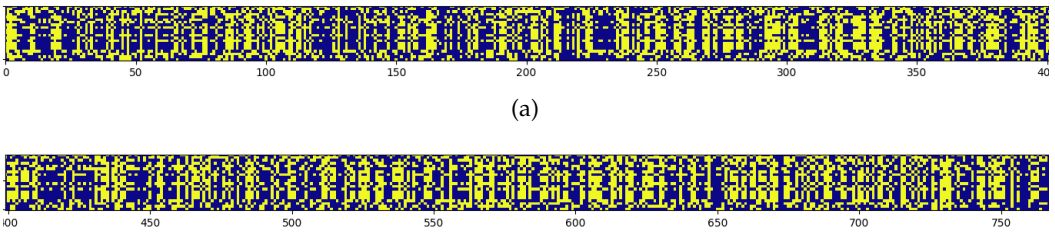


Figura 5.10: Representación más detallada de los individuos dentro del rango 337 - 348

resultados anteriores, la primer gráfica 5.11 hace correspondencia a toda la población generada en el GA. Lo primero a resaltar es que el rango de exactitud es menor, esto porque es más complicado clasificar 5 clases en lugar de solo 3, además, también se puede observar una dispersión entre los puntos, es decir, sigue sin existir una correlación entre la exactitud y el número de características seleccionadas. Aunque la dispersión no es tan notoria como en el caso anterior, incluso los puntos tienden a alejarse de las exactitudes más bajas.

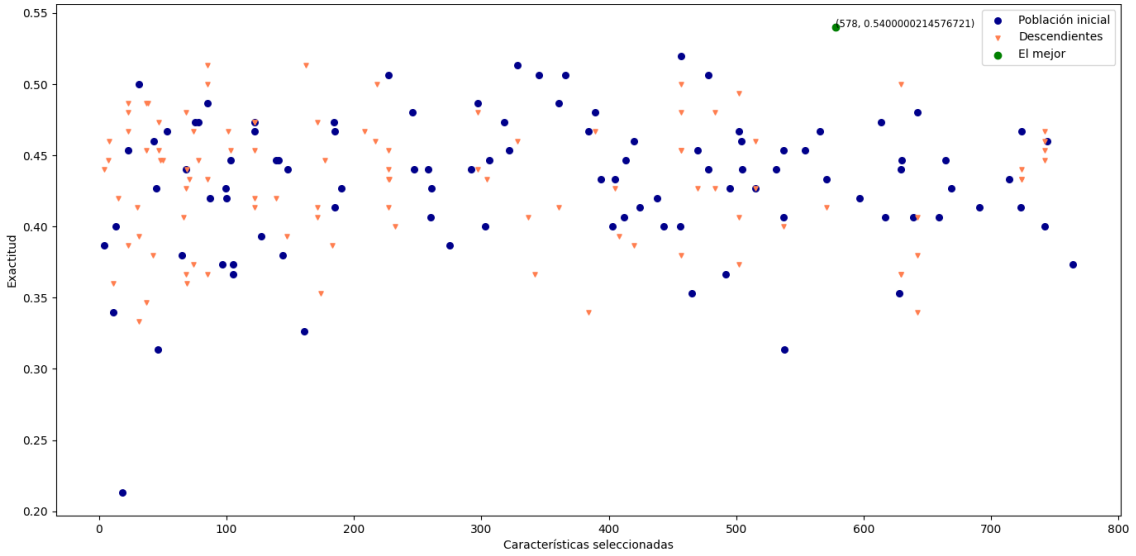


Figura 5.11: Gráfica de puntos de la población de *Amazon 2*

Para llevar el mismo proceso de análisis como en la experimentación anterior, la gráfica de barras de la Figura 5.12. En este caso se puede observar un alto número de individuos que al final tendieron por una representación más pequeña, es decir, menor número características.

Aunque existen 2 picos en la gráfica de barras, el rango considerado para proyectar su representación binaria en una imagen, fue dentro

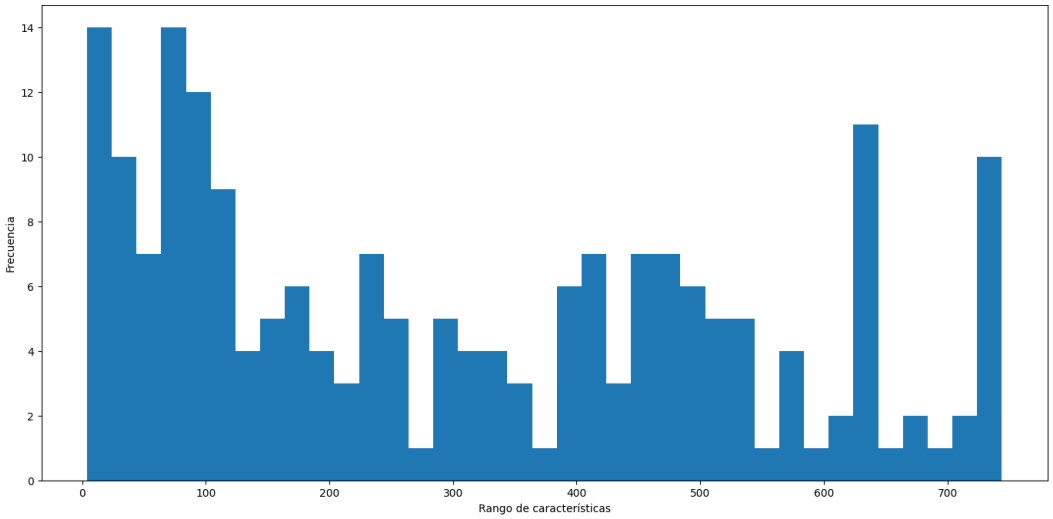


Figura 5.12: Gráfica de barras de la población de *Amazon 2*

de 2 rangos, primero donde el número de características es de 60 y 100 y el otro entre 450 y 480, esto porque también dentro de ese rango se encontró a los individuos con una exactitud cercana a la del mejor individuo. La Figura 5.13 está mostrando 23 individuos junto con el mejor (el último vector), así se puede conocer la diferencia entre las características seleccionadas.

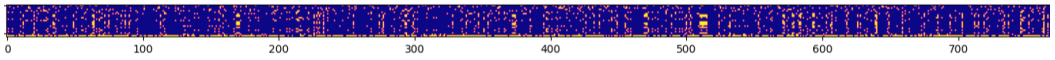


Figura 5.13: Representación de los individuos dentro del rango 60 - 100

Dado que no se puede ver claramente un patrón como con los individuos del GA anterior, la Figura 5.14 está tomando solo las primeras 400 características de cada vector para observar más detalladamente las diferencias.

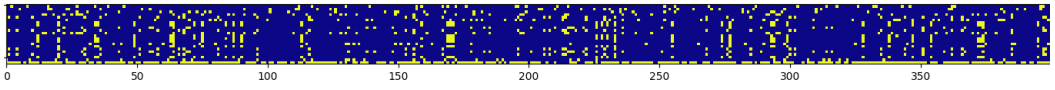
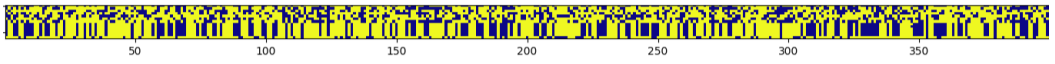


Figura 5.14: Representación más detallada de los individuos dentro del rango 60 - 100

Se puede observar como predomina el azul en gran medida, es decir que hay una ausencia de características seleccionadas en la mayor parte de individuos que se encuentran dentro del rango, por otro lado, en contraste con el mejor individuo, hay bastantes características que se comparten dado que el mejor individuo tiene 578 características seleccionadas. Ahora, considerando el otro rango,

el de 450 a 480, el número de características seleccionadas se acerca más al del mejor individuo. En la Figura 5.15(a) y 5.15(b) se ven dos fragmentos que dividen a los 13 vectores (el último es el mejor) dentro del rango, dado que estos individuos tienen una representación con un mayor número de características, se ve todavía una más alta coincidencia entre las características seleccionadas por parte del mejor individuo contra los demás. Es importante destacar que al ser más características seleccionadas, es más difícil conocer cuáles son aquellas características que son más relevantes con respecto al rendimiento, por lo tanto, también es más difícil notar un patrón.



(a)

(b)

Figura 5.15: Representación más detallada de los individuos dentro del rango 450 - 480

En este punto de la experimentación, pareciera que al aumentar el número de clases y hacer más compleja la tarea del modelo, la representación tiene que ser mayor, pero también se pudo notar que hay individuos con un menor número de características y que preservan un buen rendimiento. En otras palabras, todavía existe la incertidumbre de conocer cuáles son las características exactas que generan un mejor rendimiento, pero también ya aumentó la certeza con respecto a no requerir el mayor número de características para representar el texto y alcanzar un buen rendimiento. En la experimentación con *Amazon 3*, se puede seguir corroborando lo anterior, sobre todo en el aspecto del número de características necesario.

Ahora con *Amazon 3* aumenta el número de sentencias y disminuye el número de clases, su espacio de búsqueda se puede ver en la Figura 5.16. Aquí se puede observar cómo a comparación de utilizar solo 500 sentencias, ahora que son $5,000$ la exactitud máxima alcanzada disminuyó de 80 % con 3 clases, a 60 % con 5 clases, claro

que la razón principal con respecto a la reducción del rendimiento, está en gran parte ligada a los nuevos datos, es decir, deben de existir sentencias más ruidosas, sentencias que comparten cierta similitud entre otras sentencias de diferentes clases, todo eso hace entrenar un modelo con menos poder para reconocer patrones. Lo interesante de la gráfica de puntos es observar la constante tendencia de dispersión en los puntos, sigue sin existir una relación con respecto al número de características y la exactitud. Aunque sí hay cierta inclinación en los descendientes, la mayoría se fueron generando con una menor cantidad de características, de hecho, el mejor individuo ya no aumentó su número de características seleccionadas a pesar de hacer la búsqueda con una mayor cantidad de sentencias.

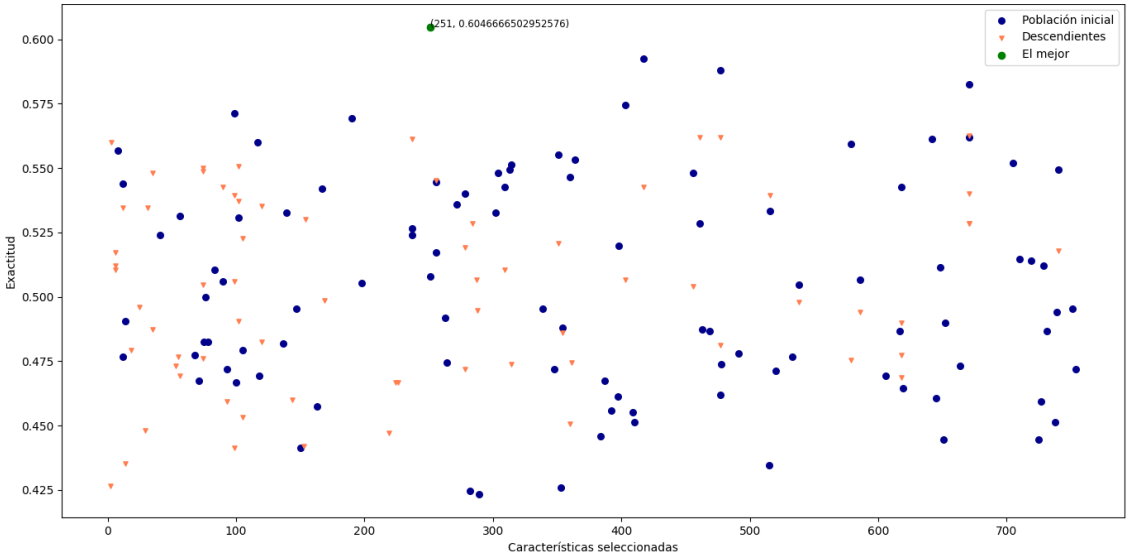


Figura 5.16: Gráfica de puntos de la población de *Amazon 3*

Para visualizar mejor la tendencia de los individuos generados se puede observar la Figura 5.17, donde se muestra como en los casos anteriores, una gráfica de barras.

En la gráfica se puede observar que hay un pico en los individuos que tienen de 20 a menos características seleccionadas, también hay una gran cantidad de individuos dentro del rango de 60 y 120, pero al igual que en el caso anterior, se tomaron en cuenta los rangos a partir de donde hay una mayor cantidad de características y mayor exactitud (cercana a la del mejor), por eso los rangos para generar las proyecciones de los vectores de cada individuo son entre 60 a 120 y 390 a 430. La Figura 5.18(a) y 5.18(b) muestran el primer rango de

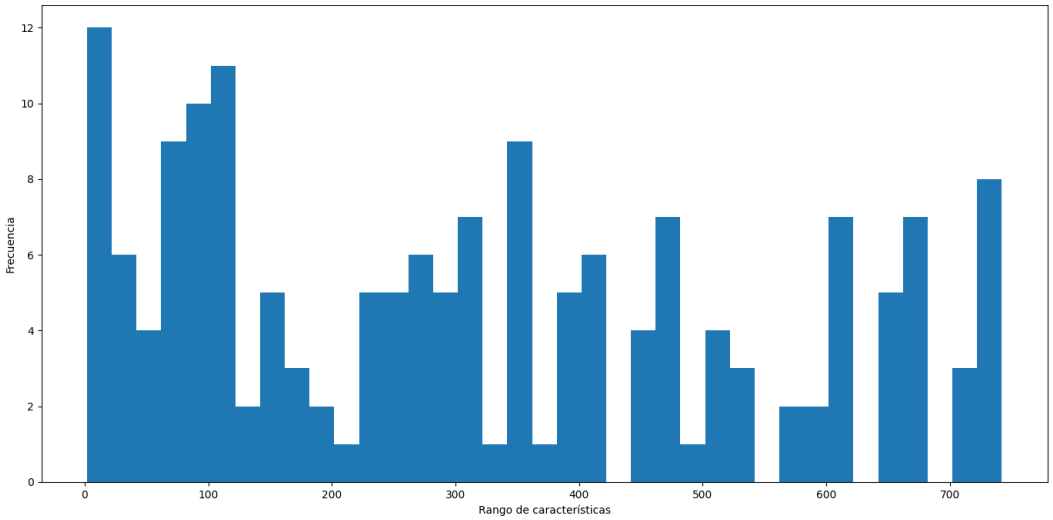
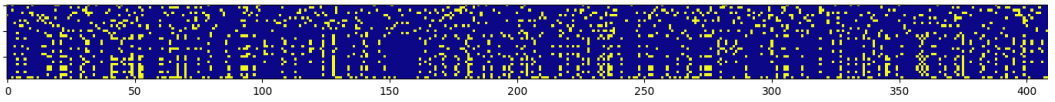
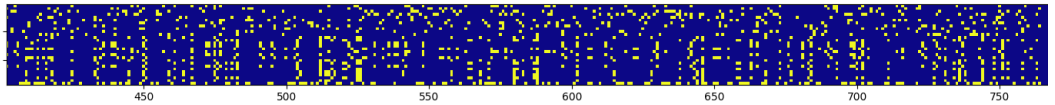


Figura 5.17: Gráfica de barras de la población de *Amazon 3*

los vectores, esto en 2 fragmentos para que sea más distinguible la diferencia entre las características seleccionadas.



(a)



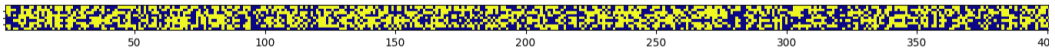
(b)

Figura 5.18: Representación más detallada de los individuos dentro del rango 60 - 120

Resalta suficiente el color azul dado que son individuos con un número de características menor a 120, pero en comparación del caso anterior donde el mejor individuo tenía 578 características seleccionadas, ahora el mejor individuo tiene tan solo 251 características, incluso si se compara con la mejor representación de *Amazon 1*, el número de características era de 212, un número menor, por lo tanto, puede existir una relación en el aumento de características actual porque la búsqueda se realizó con un mayor número de sentencias. La ventaja actual con el mejor individuo de *Amazon 3*, recae en la facilidad con la que ahora se pueden percibir las características compartidas entre los otros individuos y el mejor, incluso los patrones son más fáciles de detectar, por ejemplo, la característica justo antes de 50 del mejor individuo solo se está

compartiendo una vez con otro individuo, la primer característica seleccionada del mejor tiene un caso similar pero sí se está compartiendo un par de veces más. En otras características interesantes, también se encuentran 2 entre 300 y 350, una característica solo se comparte con varios individuos y la otra característica solo con algunos otros, pero no se comparte la misma entre sí. También hay características que se comparten casi en todos los individuos, por ejemplo, varias características antes de llegar a la 600 hay una que se comparte con varios, igual pasa para la primera característica seleccionada después del 700, esa se comparte entre varios. Por último, también hay características que solo están en el mejor individuo, por ejemplo, unos espacios después del 750 se encuentra una seleccionada.

El siguiente rango de vectores proyectados es el de 390 a 430 características, el cual se puede ver en la Figura 5.19(a) y 5.19(b). En este caso al ser un rango donde los vectores tienen un mayor número de características seleccionadas, siempre hay una característica del mejor que también está en algún otro individuo, o al menos es una probabilidad bastante alta, porque antes de llegar al número 550 hay una característica que solo la tiene el mejor, y lo mismo sucede después del número 600, pero después de esas dos características, las demás necesariamente aparecen en algún otro de los demás individuos. Algo más que se puede mencionar es en el rango entre el número 200 y 250 hay una característica no seleccionada en ningún individuo, ni en el mejor.



(a)

(b)

Figura 5.19: Representación más detallada de los individuos dentro del rango 390 - 430

Los resultados hasta ahora, siguen fundamentando la misma conclusión con respecto al número de características necesario, no hay un número en específico, pero tampoco es el mayor número

posible, es decir, no existe una tendencia, algo que se ha podido observar en las gráficas de puntos con respecto a todos los individuos generados. Además, se pudo observar de manera más detallada ciertos comportamientos de las características del mejor en comparación con los individuos de mayor frecuencia. La última representación generada es con *Amazon 4*, ahí se termina el proceso de generación y comienza el de comprobación de rendimiento con las nuevas representaciones.

Considerando los resultados hasta ahora, la nueva representación debería ser mayor a la que se obtuvo en *Amazon 2*, esto por el aumento de la complejidad en la tarea, es decir, el número de sentencias ahora es de $5,000$ y el número de clases es de 5, por lo tanto, la representación debería de ser la más grande de las otras tres representaciones obtenidas hasta ahora. Sin embargo, el mejor individuo tiene 300 características seleccionadas, además la no-relación entre el número de características y la exactitud sigue prevaleciendo. Esto se puede observar en la Figura 5.20, en donde también se puede visualizar nuevamente la tendencia del GA a generar descendientes con un menor número de características porque ahí es donde se encuentran los más cercanos en exactitud al mejor.

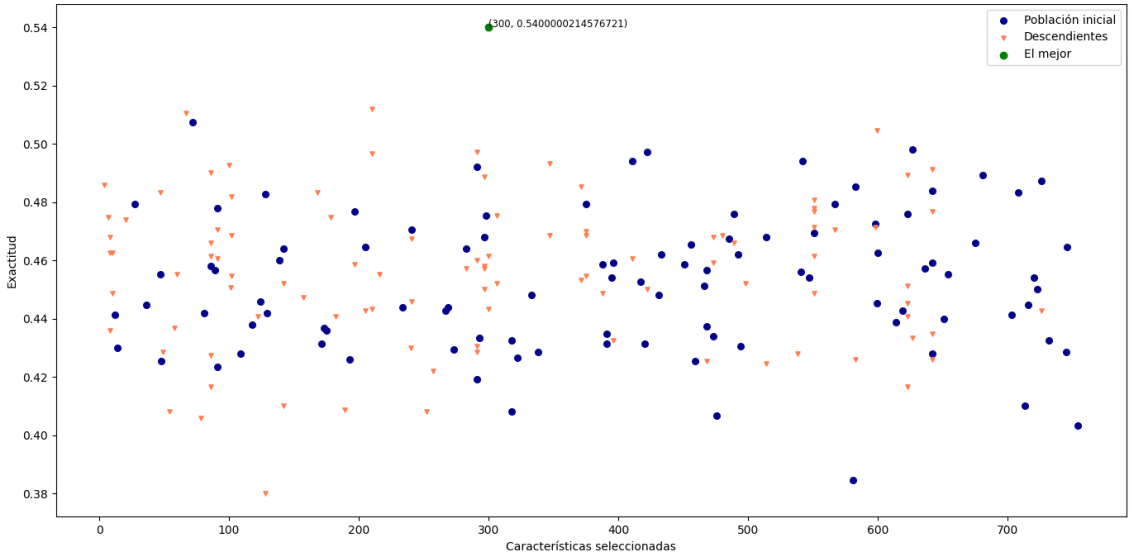


Figura 5.20: Gráfica de puntos de la población de *Amazon 4*

Para rectificar lo anterior, nuevamente existe el refuerzo por parte de

la Figura 5.21, en donde podemos ver la frecuencia de los individuos generados con respecto a su número de características.

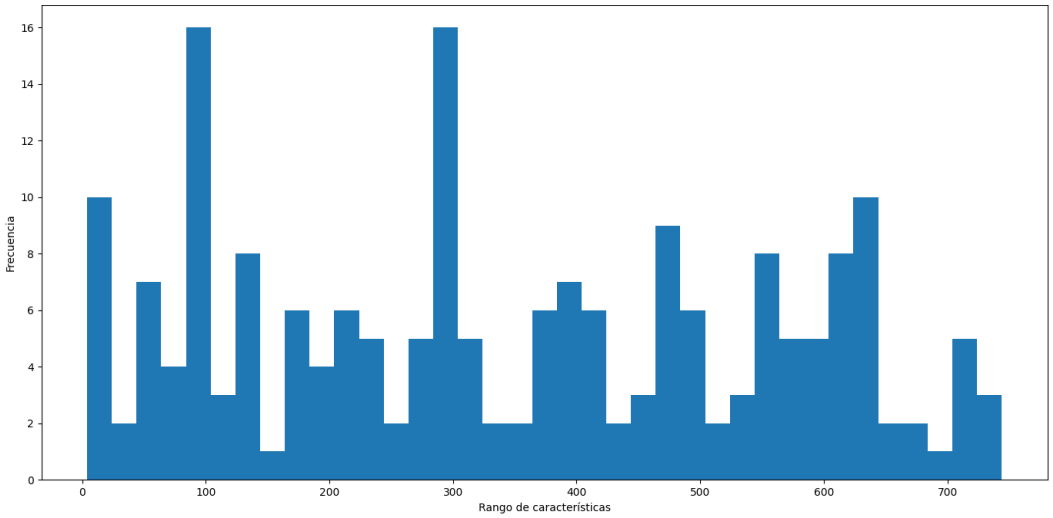
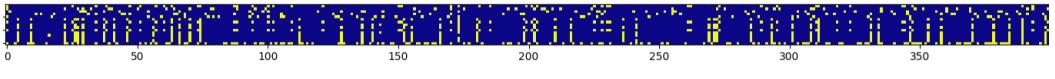


Figura 5.21: Gráfica de barras de la población de *Amazon 4*

El rango en esta ocasión se encuentra entre 60 y 100, dado que ahí está uno de los picos y a su vez algunas representaciones con una exactitud cercana a la del mejor. Los vectores proyectados se pueden observar en la Figura 5.22(a) y 5.22(b).



(a)

(b)

Figura 5.22: Representación más detallada de los individuos dentro del rango 60 - 100

En este caso hay individuos con pocas características así como en el caso anterior, pero se pueden distinguir más patrones entre el mejor individuo y los individuos dentro del rango. También hay características seleccionadas del mejor que solo tiene él, así como el caso contrario, algunos individuos tienen características seleccionadas que no tiene el mejor. En este punto de la experimentación, hay dos puntos importantes por destacar, el primero es la representación obtenida en la tarea más compleja con respecto a *Amazon 4*, el número de características es menor a lo esperado. El segundo punto es con respecto a todas las

representaciones generadas, en todos los casos el GA tuvo la libertad de generar individuos de más de 700 características, sin embargo, en ningún caso esos individuos fueron seleccionados como la mejor representación. Antes de pasar a la última sección de experimentación, se generó otra imagen de vectores, pero ahora solo con la mejor representación de cada GA, es decir, es una imagen que contiene a las cuatro mejores representaciones, esto se puede observar en la Figura 5.23(a), 5.23(b), 5.23(c) y 5.23(d). Para tener un acercamiento detallado de las cuatro diferentes representaciones en un solo espacio, la Figura se divide en cuatro imágenes.

(a)

(b)

(c)

(d)

Figura 5.23: Representación detallada de los 4 mejores individuos de cada GA

A primera vista puede parecer que no hay una relación en las características seleccionadas entre las diferentes representaciones, pero sí hay características que aparecen en las cuatro mejores representaciones, esto se puede observar entre los puntos amarillos, como se forman líneas verticales, la cuales reflejan la coincidencia entre las características seleccionadas, de hecho sucede en varios casos, y aunque en algunos no tienen las mismas cuatro características, sí hay al menos tres. Esto puede dar respuesta a la incertidumbre sembrada desde la primera experimentación, dado que si esas características prevalecieron a lo largo de los diferentes experimentos, entonces esas características deben de tener la suficiente influencia en el entrenamiento de los modelos para ser siempre parte de la mejor representación. También se puede observar

el caso contrario, hay varias características que no fueron seleccionadas en ninguna de las cuatro nuevas representaciones, por lo tanto, el razonamiento anterior también puede aplicar en este caso, pero desde un enfoque inverso, donde esas características no afectan realmente al rendimiento del modelo y por eso en ninguna representación fueron seleccionadas. Por último, tres de las cuatro mejores representaciones seleccionaron un número menor o igual a 300 características, así como también en las diferentes poblaciones generadas, la tendencia en la mayoría de los casos estaba en individuos con un número de características menor a 300.

Así concluye la experimentación dirigida a obtener nuevas representaciones de texto, el siguiente ciclo de experimentos consiste en conocer el rendimiento de las nuevas representaciones con diferentes conjuntos de datos, además de contemplar que los datos de *Amazon* todavía no se prueban en su totalidad, hasta ahora se hizo un tipo de *few-shot learning* y también hace falta comprobar su rendimiento.

5.4. Rendimiento de las nuevas representaciones de texto

El primer conjunto de datos utilizado para conocer el rendimiento de las nuevas representaciones es el de *Amazon*. Las 4 nuevas representaciones junto con la representación original van a pasar por el mismo proceso de entrenamiento, es decir, todas las representaciones estarán compitiendo bajo las mismas condiciones a nivel de parámetros. Lo anterior se puede ver en la Tabla 5.11, ahí se describen cada uno de los parámetros de las 5 representaciones, además de la configuración para los demás conjuntos de datos.

El parámetro de *Número de tokens* se obtiene a partir de cada conjunto, es decir, se calcula la mediana del número de tokens de todas las sentencias para un conjunto en específico. El número de *épocas* se define trabajando con la representación original, en otras palabras, se entrena un modelo con la representación original y se guarda el comportamiento con respecto a la exactitud, entonces a

Parámetro	Amazon	Tweets	Política
Sentencias entrenamiento	$200,000$	$1,839$	$1,371$
Sentencias prueba	$5,000$	870	609
Sentencias validación	$5,000$	324	459
Número de tokens	27	33	29
Tamaño del lote	36	64	64
No. Clases	5	3	3
No. Épocas	12	128	128
Learning rate	$1e-5$	$1e-5$	$1e-5$
No. Ejecuciones	10	10	10

Tabla 5.11: Parámetros usados para cada conjunto de datos

partir de las gráficas generadas se asigna el valor correspondiente. La Figura 5.4 muestra como en el entrenamiento del modelo utilizando los datos de *Amazon*, después de la época 6 ya no se alcanzó otra configuración del modelo que tuviera un mejor rendimiento, a pesar de que el número máximo de épocas era de 32, por ello el número de épocas establecido en este caso fue de 12.

Los resultados de la comparación entre las diferentes representaciones y la original, se pueden observar en la Tabla 5.12, cada valor exceptuando el de la última fila, es la **mediana** y la **mejor** de las 10 ejecuciones.

Representación	Accuracy		Recall		F1		Tamaño entrada
	Mediana	Mejor	Mediana	Mejor	Mediana	Mejor	
Original (768)	0.5625	0.5668	0.5625	0.5668	0.5589	0.5623	$20,736$
Amazon 1 (212)	0.5618	0.5698	0.5618	0.5698	0.5611	0.5658	$5,724$
Amazon 2 (578)	0.5596	0.5672	0.5596	0.5672	0.5542	0.5615	$15,606$
Amazon 3 (251)	0.5593	0.5669	0.5593	0.5670	0.5580	0.5639	$6,777$
Amazon 4 (300)	**0.5653**	**0.5748**	**0.5653**	**0.5748**	**0.5595**	**0.5706**	**8,100**
mBERT	-	0.5690	-	-	-	-	-

Tabla 5.12: Resultados de la comparación con los datos de *Amazon*

Los resultados muestran que la representación de *Amazon 4* fue donde se obtuvo el valor de *Accuracy* más alto, tanto contra la representación *Original* como contra la comparación indirecta con la técnica de *mBERT*, es decir, el valor proviene de la fuente de donde se obtuvieron los datos (Keung *et al*., 2020), ahí entrenan una variante de BERT nombrado mBERT (BERT multilingüe), con el cual obtuvieron el valor mostrado en la tabla. También se realizó una

prueba estadística para conocer si realmente existe una diferencia entre el rendimiento de las dos representaciones. Primero se está proyectando la distribución de la exactitud con respecto a la representación de *Amazon 4* y la representación *Original*, esto se puede observar en la Figura 5.24.

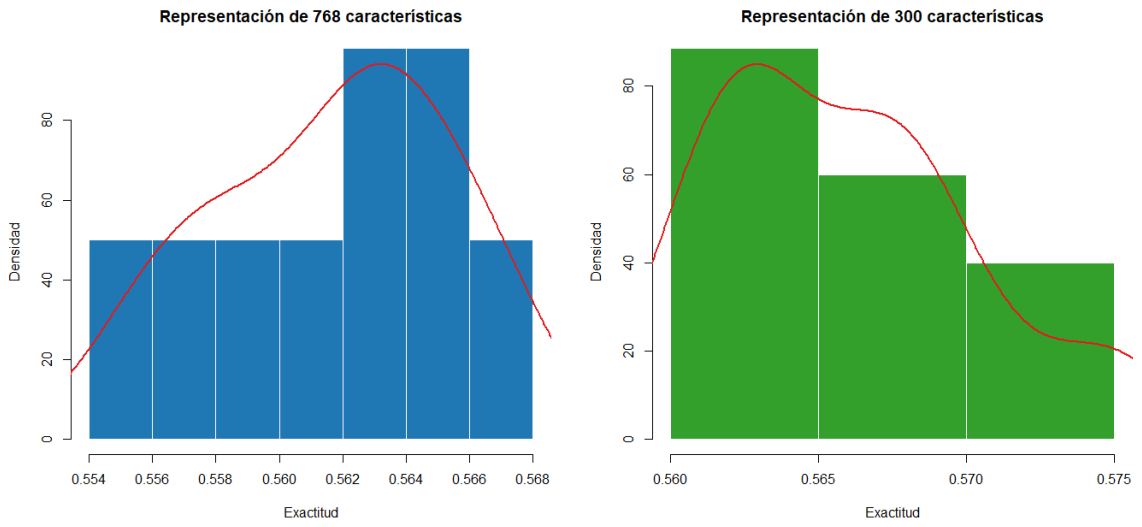


Figura 5.24: Distribución de la representación de *Amazon 4* y la representación *Original*

Dada la distribución de los datos anteriores, para verificar si es una distribución normal o no, se está utilizando la técnica de *Shapiro-Wilk*, la cual arrojó como resultado un *p-value* de **0.02087**, por lo tanto su distribución no es normal, así que la prueba estadística utilizada es la de *Kruskal Wallis*, de esta manera se puede conocer si hay diferencias significativas. El valor del *p-value* para las dos muestras de *Accuracy* fue de **0.05363**, por lo tanto la hipótesis nula no se puede rechazar. Por último, se puede ver una matriz de confusión con respecto al mejor resultado en la Figura 5.25. Algo más a destacar es la última columna, no solo se está logrando un mejor rendimiento, también se reduce el tamaño de la entrada para cualquier modelo.

La siguiente comparación es con los datos de *Tweets*, pero al igual que en el caso anterior, primero se hace referencia a la gráfica con la cual se determina el número de épocas 5.26. Ahí se puede observar como el número de épocas para la mayoría de casos se estabiliza antes de las 60 épocas, pero también existen algunos casos donde aún se encuentra una mejor configuración antes de la época 120, por esa situación el número de épocas para esta comparación es de 128.

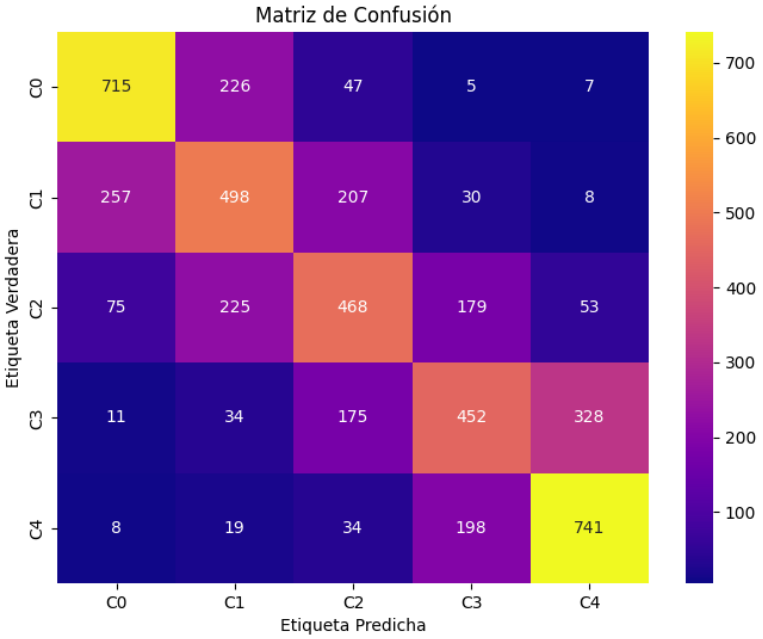


Figura 5.25: Matriz de confusión del mejor resultado en el primer experimento de comparación

Se debe de mencionar que los puntos al final de la gráfica están haciendo referencia a la exactitud obtenida con los datos de prueba, mientras las líneas de puntos son con respecto a los datos de validación. También hay un aumento en el número de tokens, a comparación de los datos de *Amazon*, dado que ahora se está trabajando con una menor cantidad de sentencias, el número de tokens a considerar ya no es el de la media, ahora es el número de tokens donde se encuentra el 95 % de los datos, esto con el fin de tener más información en la entrada del modelo.

Los resultados de la comparación con las diferentes representaciones se puede observar en la Tabla 5.13. Es importante resaltar el hecho de que ahora las nuevas representaciones están trabajando en dominios diferentes de donde fueron obtenidas.

En los resultados de ahora sigue destacando la representación de *Amazon 4*, esto porque se volvió a obtener el valor más alto en las medidas de *Accuracy*, *Recall* y *F1* con respecto a las otras representaciones generadas y la representación original. Para el caso de la comparación indirecta contra la técnica de *XLM-T* (XLM-Twitter) hace referencia al modelo generado por los mismos

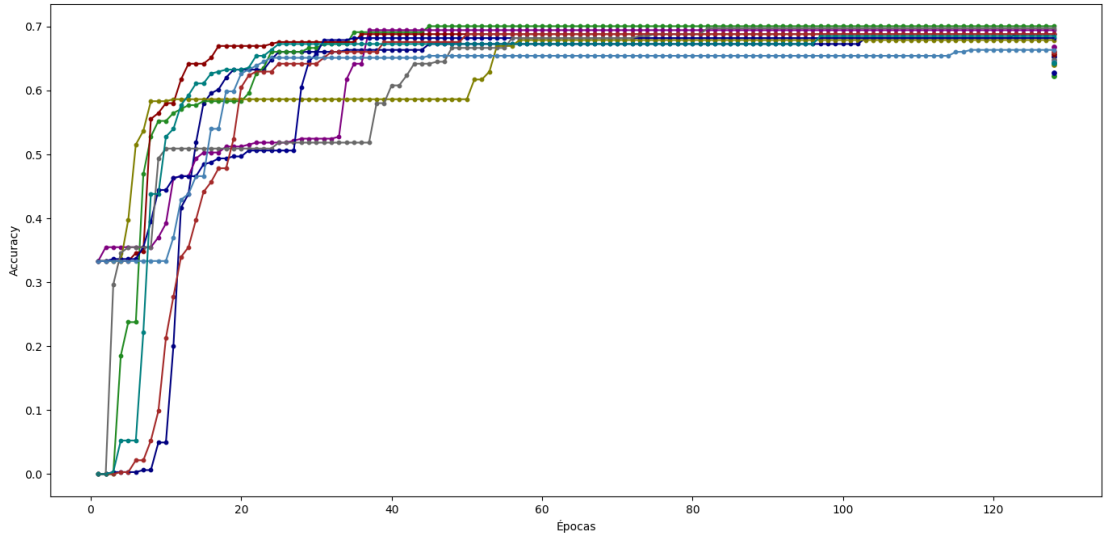


Figura 5.26: Comportamiento de la exactitud con los datos de *Tweets*

Representación	Accuracy		Recall		F1		Tamaño entrada
	Mediana	Mejor	Mediana	Mejor	Mediana	Mejor	
Original (768)	0.6460	0.6678	0.6460	0.6678	0.6435	0.6677	25,344
Amazon 1 (212)	0.6466	0.6678	0.6466	0.6678	0.6440	0.6664	6,996
Amazon 2 (578)	0.6563	0.6678	0.6563	0.6678	0.6529	0.6688	19,074
Amazon 3 (251)	0.6517	0.6621	0.6517	0.6621	0.6535	0.6598	8,283
Amazon 4 (300)	**0.6534**	**0.6724**	**0.6534**	**0.6724**	**0.6506**	**0.6732**	**9,900**
XLM-R	-	-	-	-	-	0.6587	-
XLM-T	-	-	-	-	-	**0.6852**	-

Tabla 5.13: Resultados de la comparación con los datos de *Tweets*

autores que proporcionan el conjunto de datos actual, dicho modelo está entrenado por una gran cantidad de datos de diversos idiomas, y después valoran su rendimiento utilizando otros datos de donde tienen una versión para obtener su rendimiento en el idioma español. Un caso similar sucede con el modelo XLM-R (XLM-RoBERTa) propuesto por (Conneau *et al.*, 2019), de hecho este modelo lo utilizan como su medio de comparación porque es su referencia base. Para el caso de XLM-R se logra tener un mejor rendimiento con la representación de *Amazon 4*, y para XLM-T no se alcanza a superar, sin embargo la representación sí fue la más alta en los otros casos.

Considerando que los datos en este caso se mantienen equilibrados, el proceso de la prueba estadística para comparar la representación *Original* contra la de *Amazon 4* será similar al caso anterior, primero se puede observar la gráfica de distribución para cada caso en la Figura 5.27. En esta ocasión el *p-value* de la prueba de *Shapiro-Wilk*

fue de **0.5802**, por lo tanto, las muestran pertenecen a una distribución normal, y la prueba estadística en este caso es la de *t-test*, con la cual se obtuvo un *p-value* de **0.4743**, de esta manera la hipótesis nula sigue sin poder rechazarse. Por último, la matriz de confusión para este conjunto de datos se puede observar en la Figura5.28.

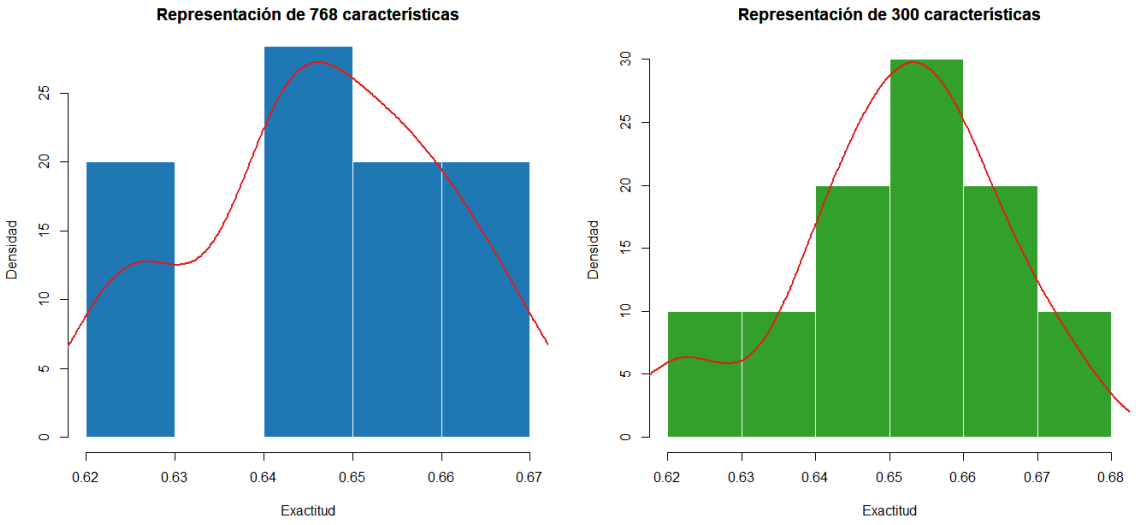


Figura 5.27: Distribución de la representación de *Amazon 4* y la representación *Original*

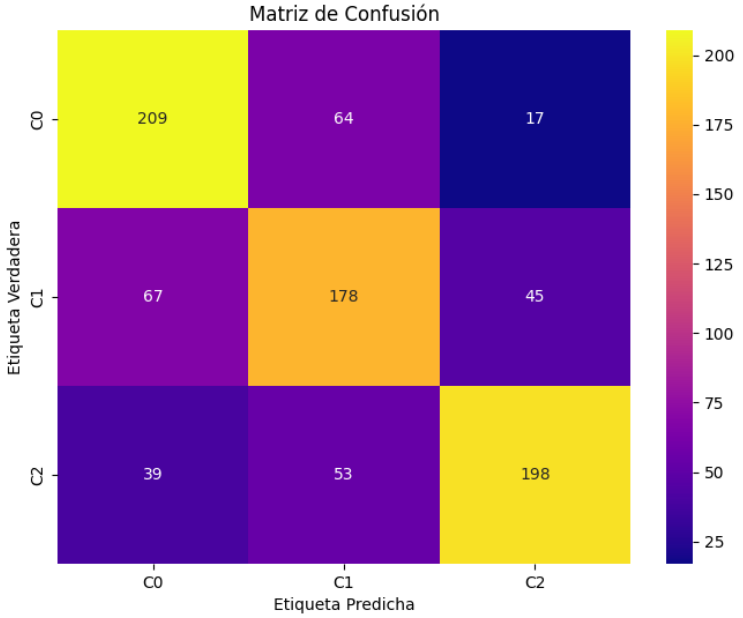


Figura 5.28: Matriz de confusión del mejor resultado en el segundo experimento de comparación

Hasta este punto el rendimiento en general de las nuevas

representaciones está manteniendo un rendimiento competitivo, no se alejan mucho de los mejores valores e incluso en algunos casos están superando tanto a la representación original, como a las otras técnicas. El siguiente conjunto a comparar son los datos de *Política*, pero antes de la tabla viene la respectiva gráfica para validar el número de épocas utilizado, esto en la Figura 5.29.

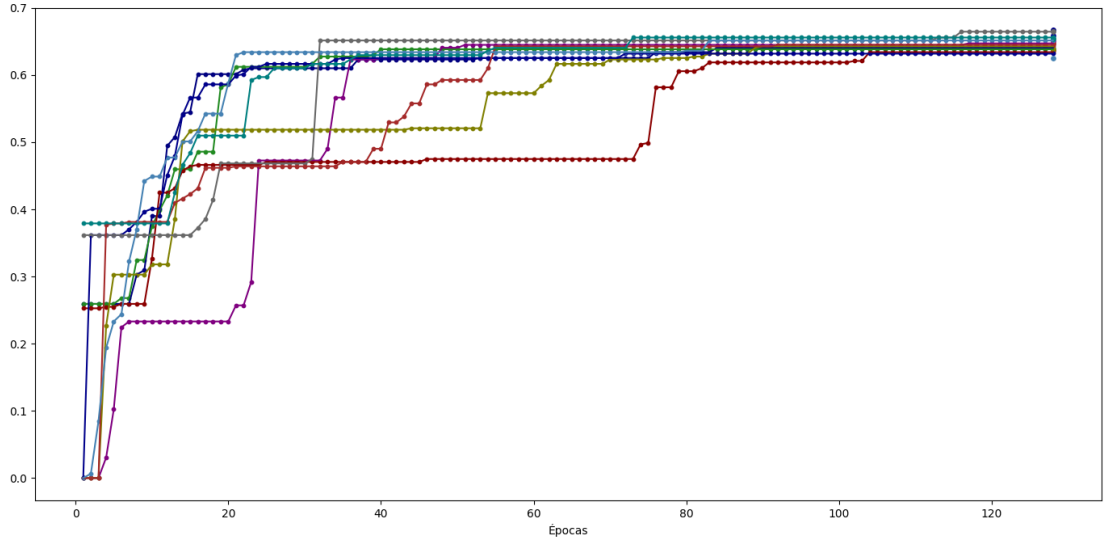


Figura 5.29: Comportamiento de la exactitud con los datos de *Política*

El comportamiento de las diferentes ejecuciones refleja una mayor variación a diferencia del caso anterior, sin embargo, antes de llegar a la época 120 ya se han estabilizado todas, lo mismo sucedía con los datos de *Tweets*, por esa misma razón el número de épocas en esta ocasión sigue siendo de 128. Por último antes de pasar a la tabla de resultados, el número de tokens seleccionado corresponde nuevamente al número que se encuentra en el 95 % de las sentencias. Los resultados de la comparación se pueden ver en la Tabla 5.14.

Representación	**Accuracy**		**Recall**		**F1**		Tamaño
	Mediana	Mejor	Mediana	Mejor	Mediana	Mejor	entrada
Original (768)	0.6519	0.6667	0.6519	0.6667	0.6514	0.6667	22,272
Amazon 1 (212)	**0.6535**	**0.6765**	**0.6535**	**0.6765**	**0.6555**	**0.6764**	**6,148**
Amazon 2 (578)	0.6527	0.6650	0.6527	0.6650	0.6510	0.6663	16,762
Amazon 3 (251)	0.6527	0.6749	0.6527	0.6749	0.6530	0.6739	7,279
Amazon 4 (300)	**0.6470**	**0.6782**	**0.6470**	**0.6782**	**0.6468**	**0.6780**	**8,700**
BETO fine-tuning	-	0.6258	-	0.6220	-	0.6211	-

Tabla 5.14: Resultados de la comparación con los datos de *Política*

Nuevamente los valores más altos fueron obtenidos por parte de la

representación de *Amazon 4*, y con base en la mediana la mejor representación fue la de *Amazon 1*, otra vez la representación original no fue la mejor. La última fila hace referencia a la última comparación indirecta de la técnica utilizada por los autores (Salgueiro *et al.*, 2022) que proporcionan el conjunto de datos utilizado para estos resultados, en esta ocasión los valores de *Amazon 4* superan considerablemente los valores del *BETO fine-tuning*. Así como en el caso anterior, se realizó una prueba estadística entre la representación *Original* y la representación de *Amazon 4*, la gráfica de distribución se puede observar en la Figura 5.30, en este caso se está proyectando el valor de **F1** dado que los datos están desequilibrados.

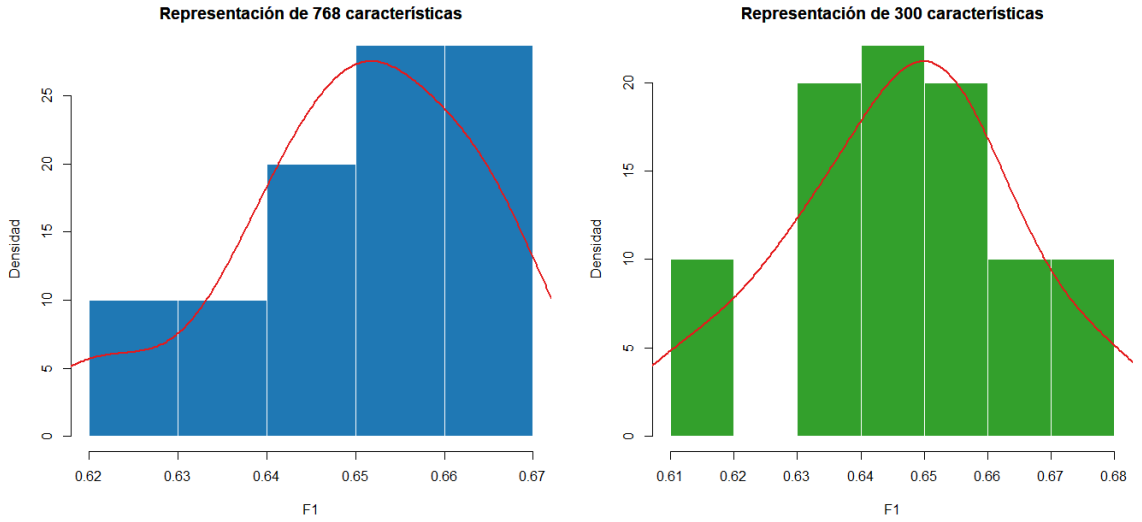


Figura 5.30: Distribución de la representación de *Amazon 4* y la representación *Original*

Para la prueba de *Shapiro-Wilk* el *p-value* en esta ocasión fue de **0.6881**, de tal forma que las muestras pertenecen a una distribución normal, esto permite utilizar nuevamente la prueba estadística de *t-test*, con la cual se obtuvo un *p-value* de **0.6424**, por lo tanto, en esta ocasión tampoco se puede rechazar la hipótesis nula. La matriz de confusión de los mejores valores se pueden observar en la Figura 5.31.

En la Sección 4.2 se describió un conjunto de datos más, específicamente el de *muchocine.net*, pero dada la naturaleza de los datos obtenidos, es decir, el desbalanceo tan significativo que existe entre sus clases, al final ese conjunto se dejó fuera las comparaciones de rendimiento. Además, los resultados hasta ahora ya han demostrado la relevancia que existe en las nuevas representaciones,

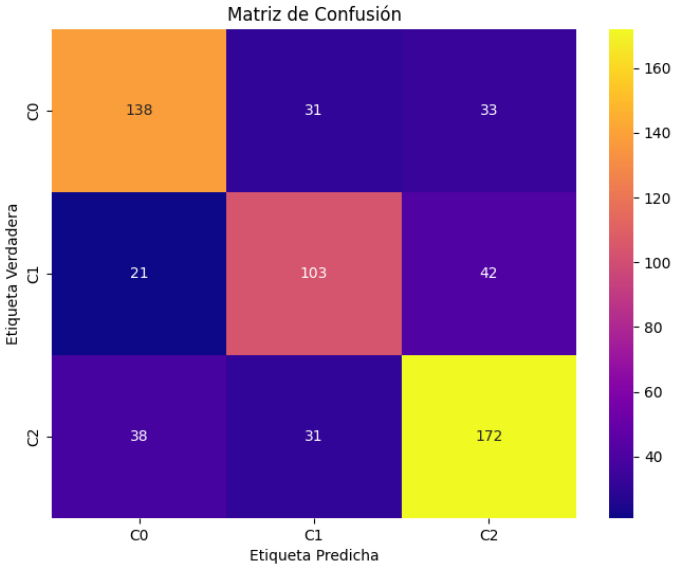


Figura 5.31: Matriz de confusión del mejor resultado en el tercer experimento de comparación

en todos los casos se alcanzó y superó el rendimiento en comparación con la representación original, también en la mayoría de los casos las nuevas representaciones superaron a las referencias de donde se obtuvieron los datos.

6. Conclusiones y Trabajo futuro

Los resultados inicialmente mostraron que el GA tiene un buen desempeño a la hora de realizar la búsqueda de nuevas representaciones, entre sus diferentes operadores favorecen la diversidad y permiten mantener una búsqueda amplia en el espacio del problema. En el caso de las búsquedas más grandes los resultados permitieron observar cómo no existe una relación entre el número de características seleccionadas y el número de sentencias, por otro lado, las nuevas representaciones mantienen un buen rendimiento a pesar de variar su dominio, algo que se había planteado en la hipótesis de este trabajo de investigación. En los últimos experimentos, aquellos donde se pusieron a prueba las nuevas representaciones, se obtuvo un rendimiento competitivo, así fue el caso para la representación de *Amazon 4*, con tan solo el 39 % del número total de características, logró obtener los valores más altos en comparación de la representación original.

El trabajo de investigación también ha proporcionado principalmente un acercamiento en lo que respecta a la representación más famosa actualmente del texto, esto para saber si la representación actual con todas sus características es realmente indispensable o puede variar según el dominio de una tarea o el número de datos. Por ello todos los resultados obtenidos han dado pauta a un amplio panorama, donde el número de características y sus diferentes combinaciones sí hacen variar significativamente el rendimiento y la eficiencia del modelo, algo bastante interesante porque existen diversas tareas, con diversos enfoques y diversos números de datos, por lo tanto, las representaciones también podrían ser más versátiles, pensando en modelos con la capacidad de asignar una representación base al inicio, pero que se pueda ir ajustando tanto en dimensiones como en el rango de sus valores conforme avanza su entrenamiento.

En general lo más importante a destacar con los resultados obtenidos es que nunca hubo una relación clara entre el número de características y el rendimiento obtenido, de hecho, el espacio del

problema en cada ejecución tenía individuos de una diversidad considerable. Además, el hecho de excluir varias de sus características no llevó al rendimiento a caer en picada, al contrario, existieron varios individuos con pocas características que obtenían un rendimiento cercano al mejor individuo. En gran medida se debe de recordar que uno de los propósitos de este trabajo de investigación era demostrar si el total de características tenían la misma relevancia, y con los resultados podemos observar todo lo contrario, por eso es de suma importancia seguir estudiando cuáles son aquellas características con mayor relevancia, dado que las 768 no necesariamente lo son.

Bajo el panorama de las nuevas representaciones obtenidas, se pueden crear otras, por ejemplo, seleccionando toda aquella característica que coincida con las mejores 4, incluso si solo coincide en al menos 3. De otra forma se pueden hacer otras ejecuciones del GA y generar nuevas representaciones para ir encontrando más patrones de las características más relevantes.

Todas las nuevas representaciones internamente pertenecen a diferentes rangos de valores, es decir, cada una de sus características tienen un valor y en cualquier caso existe un mínimo y un máximo, como más trabajo futuro se puede realizar una búsqueda entre los diferentes rangos que se generan a partir de las diferentes representaciones. Otra propuesta es con respecto a los datos de *Amazon*, en el trabajo de investigación actual solo se trabajaron las sentencias, pero también se puede usar el título y la categoría para agregar más información, esto puede generar nuevos resultados que se pueden comparar con los del artículo. Algo similar pasa con los datos de *Política*, también hay otros valores para agregar más información a las entradas del modelo. Un último trabajo futuro recae en la importancia de mejorar la calidad de los textos que entran a los modelos del lenguaje, por eso la propuesta se basa en realizar un nuevo enfoque de limpieza o filtrado, no para eliminar cualquier ruido, sino incluso para agregar más información relevante al texto, y con ello lograr mejores rendimientos.

Bibliografía

Angel, M., Carmona, A., Jurado-Buch, J. D., Minayo-Díaz, S., Tello, J. A., Chaucanes, K. E., Salazar, L. V., Oquendo-Coral, M. D., & Álvarez-Carmona, M. Á. (2023). A single model based on beto to classify spanish tourist opinions through the random instances selection. http://ceur-ws.org

Anthony, L. F. W., Kanding, B., & Selvan, R. (2020). Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. http://arxiv.org/abs/2007.03051

Antony Gnana Singh, D. A., Leavline, E. J., Priyanka, R., & Priya, P. P. (2016). Dimensionality reduction using genetic algorithm for improving accuracy in medical diagnosis. *International Journal of Intelligent Systems and Applications*, *8*(1), 67-73. https://doi.org/10.5815/ijisa.2016.01.08

Barbieri, F., Anke, L. E., & Camacho-Collados, J. (2022). Xlm-t: Multilingual language models in twitter for sentiment analysis and beyond. https://huggingface.co/

Cai, J., Luo, J., Wang, S., & Yang, S. (2018). Feature selection in machine learning: A new perspective. *Neurocomputing*, *300*, 70-79. https://doi.org/10.1016/j.neucom.2017.11.077

Cañete, J., Chaperon, G., Fuentes, R., Ho, J.-H., Kang, H., & Pérez, J. (2020). Spanish pre-trained bert model and evaluation data. *PML4DC at ICLR 2020*.

Castorena-Salas, L. G., Sánchez-Vega, F., & López-Monroy, A. P. (2023). Buaa's team in rest-mex 2023-sentiment analysis: A basic and efficient stylistic and thematic features approach. http://ceur-ws.org

Collobert, R., Weston, J., Com, J., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., & Stoyanov, V. (2019). Unsupervised cross-lingual representation learning at scale. http://arxiv.org/abs/1911.02116

Dalal, M. K., & Zaveri, M. A. (2011). Automatic text classification: A technical review.

Deng, X., Li, Y., Weng, J., & Zhang, J. (2019). Feature selection for text classification: A review. *Multimedia Tools and Applications*, *78*(3), 3797-3816. https://doi.org/10.1007/s11042-018-6083-5

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, *1*(1950), 4171-4186.

Dumitrescu, D., Lazzerini, B., Jain, L., & Dumitrescu, A. (2000). *Evolutionary computation*. Taylor & Francis. https://books.google.com.mx/books?id=MSU9ep79JvUC

Fernandez, L. (2021). *Imdb dataset of 50k movie reviews (spanish)*. https://www.kaggle.com/datasets/luisdiegofv97/imdb-dataset-of-50k-movie-reviews-spanish

Garcia-Diaz, J. A., Garcia-Sanchez, F., & Valencia-Garcia, R. (2023). Smart analysis of economics sentiment in spanish based on linguistic features and transformers. *IEEE Access*, *11*, 14211-14224. https://doi.org/10.1109/ACCESS.2023.3244065

Garcia-Silva, A., & Gomez-Perez, J. M. (2021). Classifying Scientific Publications with BERT - Is Self-attention a Feature Selection Method? *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12656 LNCS*, 161-175. https://doi.org/10.1007/978-3-030-72113-8_11

Guzmán-Landa, J. J., Hernández-Hernández, J. C., Hoyos-Rivera, G. D. J., & Mezura-Montes, E. (2023). Feature selection of beto token embeddings using a genetic algorithm. https://www.rcs.cic.ipn.mx/2023_152_7/Seleccion%20de%20caracteristicas%20de%20representaciones%20de%20texto%20de%20BETO%20usando%20un%20algoritmo%20genetico.pdf

Hern, C., & Rodr, O. (2021). Neuroevolution for Sentiment Analysis. (550), 101-110.

Imani, M. B., Keyvanpour, M. R., & Azmi, R. (2013). A novel embedded feature selection method: A comparative study in the application of text categorization. *Applied Artificial Intelligence*, *27*, 408-427. https://doi.org/10.1080/08839514.2013.774211

Keung, P., Lu, Y., Szarvas, G., Smith, N. A., & Amazon, ‡. •. (2020). The multilingual amazon reviews corpus. https://github.com/fxsjy/jieba

Kim, Y. (2014). Convolutional neural networks for sentence classification. http://arxiv.org/abs/1408.5882

Kowsari, K., Meimandi, K. J., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Text classification algorithms: A survey. https://doi.org/10.3390/info10040150

Kulkarni, A. J., Mezura-Montes, E., Wang, Y., Amir, ·., Gandomi, H., & Krishnasamy, G. (2021). Constraint handling in metaheuristics and applications.

Liu, Z., Lin, Y., & Sun, M. (2020). *Representation learning for natural language processing*. Springer Nature Singapore. https://doi.org/10.1007/978-981-15-5573-2

Liu, Z., Lin, Y., & Sun, M. (2021). Representation learning for natural language processing.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 142-150. http://www.aclweb.org/anthology/P11-1015

Martínez-Cámara, E., Almeida-Cruz, Y., Díaz-Galiano, M. C., Estévez-Velarde, S., García-Cumbreras, M. A., García-Vega, M., Gutiérrez, Y., Montejo-Ráez, A., Montoyo, A., Muñoz, R., Piad-Morffis, A., & Villena-Román, J. (2018). Overview of tass 2018: Opinions, health and emotions resumen de tass 2018: Opiniones, salud y emociones. http://www.sepln.org/workshops/tass

Mata, F. L. C. (2017). http://www.lsi.us.es/~fermin/index.php/Datasets

Mcculloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. http://arxiv.org/abs/1301.3781

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Meysam, C., & Jianfeng, G. (2021). Deep learning based text classification: A comprehensive review. *2022-March*. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Pan, R., García-Díaz, J. A., Garcia-Sanchez, F., & Valencia-García, R. (2023). Evaluation of transformer models for financial targeted sentiment analysis in spanish. *PeerJ Computer Science*, *9*. https://doi.org/10.7717/peerj-cs.1377

Salgueiro, T. A., Zapata, E. R., Furman, D., Pérez, J. M., & Larrosa, P. N. F. (2022). A spanish dataset for targeted sentiment analysis of political headlines. http://arxiv.org/abs/2208.13947

Tang, W. H., & Wu, Q. H. (2011). Condition monitoring and assessment of power transformers using computational intelligence. *Power Systems*, *58*. https://doi.org/10.1007/978-0-85729-052-6

Thirumoorthy, K., & Muneeswaran, K. (2022). Feature Selection for Text Classification Using Machine Learning Approaches. *National Academy Science Letters*, *45*(1), 51-56. https://doi.org/10.1007/s40009-021-01043-0

Tian, W., Li, J., & Li, H. (2018). A Method of Feature Selection Based on Word2Vec in Text Categorization. *Chinese Control Conference, CCC*, *2018-July*, 9452-9455. https://doi.org/10.23919/ChiCC.2018.8483345

Uğuz, H. (2011). A two-stage feature selection method for text categorization by using information gain, principal component analysis and genetic algorithm. *Knowledge-Based Systems*, *24*(7), 1024-1032. https://doi.org/10.1016/j.knosys.2011.04.014

Uysal, A. K., & Murphey, Y. L. (2017). Sentiment Classification: Feature Selection Based Approaches Versus Deep Learning. *IEEE CIT 2017 - 17th IEEE International Conference on Computer and Information Technology*, 23-30. https://doi.org/10.1109/CIT.2017.53

Vaswani, A., Brain, G., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. https://doi.org/https://doi.org/10.48550/arXiv.1706.03762

Wang, K., Huang, J., Liu, Y., Cao, B., & Fan, J. (2021). Combining Feature Selection Methods with BERT: An In-depth Experimental Study of Long Text Classification. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, *349*, 567-582. https://doi.org/10.1007/978-3-030-67537-0_34

...