Comparativa de Algoritmos de Aprendizaje Supervisado para la Clasificación Multiclase de Noticias: Un Estudio Empírico sobre Precisión y Rendimiento

Manzanares Peña Jorge Luis, Reyes Avila David, Salazár Domínguez Jesús Eduardo, Salinas Romero Daniel, Shen Shuai

Facultad de Ingeniería, Universidad Nacional Autónoma de México

Ciudad de México, México

jorgeluismanp@outlook.com
david.reyesafi@gmail.com
jesusesd5@gmail.com
daniel.salinas@ingenieria.unam.edu
shenshauai@gmail.com

***Abstract*— Este estudio compara el rendimiento de los algoritmos Naive Bayes, Árboles de Decisión, Máquinas de Soporte Vectorial (SVM), K-Vecinos más Cercanos (KNN), Bosque Aleatorio y Redes Neuronales en la clasificación multiclase de textos. Se utiliza un conjunto de datos representativo y se evalúan métricas como precisión, recall, F1-score y tiempos de ejecución. Los resultados destacan el rendimiento sobresaliente de las Redes Neuronales y Naive Bayes en términos de precisión y tiempo. La elección del algoritmo más apropiado depende del contexto y los requisitos específicos del análisis de textos, considerando también factores como complejidad computacional y tiempo de entrenamiento. Este estudio amplía la comprensión de los algoritmos de clasificación multiclase en el análisis de textos, permitiendo a investigadores y profesionales seleccionar el algoritmo más adecuado para sus aplicaciones.**

I. INTRODUCCIÓN

El presente artículo tiene como objetivo comparar la precisión y el rendimiento de diferentes algoritmos de aprendizaje supervisado para realizar clasificación multiclase. Los algoritmos que se tomarán en cuenta son:

- Naive Bayes
- Árboles de Decisión
- Máquinas de Soporte Vectorial (SVM)
- K-Vecinos más Cercanos (KNN)
- Bosque aleatorio
- Redes Neuronales

La clasificación se llevará a cabo sobre artículos de noticias, las cuales corresponden a una de cuatro posibles categorías: internacional, deportes, negocios y ciencia/tecnología. El conjunto de datos a utilizar consta de 127600 registros con tres atributos cada uno: título, descripción y categoría.

Las noticias fueron originalmente recopiladas por Antonio Gulli, *Senior Director* en Google, a través de la herramienta *Come to my Head.* El conjunto de datos de clasificación fue elaborado por Xiang Zhang a partir del conjunto de noticias anterior. Este último fue utilizado en el artículo *Character-level Convolutional Networks for Text Classification.*

A priori, se espera que las redes neuronales tengan la mejor precisión de entre todos los algoritmos. No obstante, también se espera que tengan el peor rendimiento en términos del tiempo necesario para el entrenamiento del modelo. Por otro lado, Naive Bayes debería ser el algoritmo más rápido, al tiempo de mantener una precisión decente. El resto de algoritmos deberían tener un rendimiento y precisión intermedios.

Además, el uso de análisis TF-IDF (*Term Frequency-Inverse Document Frequency*) como parte del proceso de clasificación agrega un enfoque más sofisticado al considerar la importancia de las palabras en el contexto de los documentos en los que aparecen. Esto proporciona una mejor comprensión y representación de los artículos de noticias, lo que puede influir en el rendimiento y la precisión de los algoritmos de aprendizaje supervisado.

La motivación para realizar este proyecto radica en la importancia y utilidad del análisis inteligente de textos en el campo de la clasificación multiclase de artículos de noticias. En la era de la información digital, la cantidad de noticias y contenido textual disponible es abrumadora, lo que dificulta el proceso de encontrar y organizar la información relevante. La capacidad de automatizar este proceso a través de algoritmos de aprendizaje supervisado permite ahorrar tiempo y recursos, y brinda la oportunidad de obtener información valiosa de manera rápida y precisa.

La clasificación de artículos de noticias tiene aplicaciones prácticas en diversos ámbitos, como la industria periodística, donde la capacidad de categorizar y etiquetar automáticamente los artículos puede mejorar la eficiencia en la recopilación de noticias y la creación de contenido temático. Además, las organizaciones pueden utilizar estas técnicas para realizar un seguimiento en tiempo real de las tendencias y opiniones en diferentes áreas, lo que les permite tomar decisiones informadas y adaptar sus estrategias en consecuencia.

En las siguientes secciones del documento se explicará a detalle las características de cada algoritmo, así como la metodología seguida para implementarlos y el tratamiento dado al conjunto de datos. Además, se mostrarán los resultados obtenidos, comparándolos con las hipótesis previamente realizadas y se generarán conclusiones con base en ellos.

II. ESTADO DE LA CUESTIÓN

El análisis inteligente de textos y la clasificación multiclase de artículos de noticias han sido objeto de investigación en diversos estudios y trabajos previos. La comprensión de las investigaciones existentes en este campo resulta fundamental para situar y contextualizar el presente estudio, así como para identificar las contribuciones y limitaciones de las investigaciones previas. Se destacarán los enfoques más exitosos y los algoritmos de aprendizaje supervisado que han demostrado un rendimiento destacado en la clasificación multiclase de artículos de noticias.

Asimismo, se mencionan los desafíos encontrados en los trabajos previos, identificando el contraste con el estudio planteado en el presente proyecto.

1. "*Text Categorization with Support Vector Machines: Learning with Many Relevant Features*" (Joachims; 1998).

- Este estudio pionero se centra en la clasificación de textos utilizando Máquinas de Soporte Vectorial (SVM). Proporciona una base teórica sólida y presenta resultados prometedores en la clasificación de documentos. Sin embargo, no se analizan específicamente categorías de noticias.

2. "*A re-examination of text categorization methods*" (Yang, Liu; 1999).

- Este trabajo compara el rendimiento de varios algoritmos de clasificación, incluyendo Naive Bayes, SVM, K-Vecinos más Cercanos (KNN) y redes neuronales, en la clasificación de textos. Ofrece una evaluación comparativa detallada y análisis de los diferentes enfoques.
- Evaluación comparativa detallada, pero no se considera el análisis TF-IDF.

3. "*Machine learning in automated text categorization*" (Sebastiani; 2002).

- Este estudio examina la utilización de redes neuronales en la clasificación de textos. Explora diferentes arquitecturas de redes neuronales y algoritmos de aprendizaje, y analiza su rendimiento en comparación con otros métodos de clasificación.
- No se aborda la clasificación multiclase de artículos de noticias específicamente.

4. *"Improving Text Categorization Methods for Event Tracking*" (Yang, Ault, Pierce, Lattimer; 2004).

- Este trabajo se enfoca en la clasificación de noticias para el seguimiento de eventos. Examina enfoques avanzados de clasificación, como el uso de K-Vecinos más Cercanos (KNN), y propone mejoras en el rendimiento de los sistemas de clasificación de textos.
- Propuestas de mejora, pero no se incluye análisis comparativo de diferentes algoritmos.

5. *"A Comparative Study on Feature Weight in Text Categorization" (Deng, Tang, Yang, et al.; 2004).*

- Resumen: Este estudio compara cuatro métodos de ponderación de características (tf-idf, tf-CRF, tf-OddsRatio, y tf-CHI) en la categorización de texto usando Máquinas de Soporte Vectorial. Los resultados muestran que tf-CHI es el más efectivo, logrando una alta precisión de clasificación. El método tf-idf, aunque ampliamente utilizado, no resultó ser tan efectivo como tf-CHI y tf-OddsRatio.

6. *"Character-level Convolutional Networks for Text Classification.*" (Zhang, Zhao, LeCun; 2016).

- El estudio explora el uso de redes convolucionales a nivel de caracteres para la clasificación de texto, construyendo varios conjuntos de datos grandes para este propósito. Los resultados demuestran que estas redes pueden ofrecer rendimientos excelentes, incluso superiores en comparación con modelos tradicionales como la bolsa de palabras y sus variantes TF-IDF.

III. MARCO TEÓRICO

Antes de explicar la metodología seguida en el proyecto, conviene recordar algunos conceptos teóricos necesarios para el entendimiento de los experimentos realizados. Cabe resaltar que, si bien algunos de los algoritmos explicados pueden utilizarse para diferentes propósitos, en este trabajo se emplearán únicamente para clasificación.

A. TF-IDF

TF-IDF (por sus siglas en inglés, *Term Frequency-Inverse Document Frequency*) es una técnica estadística que se utiliza para determinar qué tan importante es una palabra para un documento en un corpus. Consta de dos componentes principales:

Term Frequency (TF): Es la frecuencia de una palabra en un documento. Es decir, cuántas veces una palabra aparece en un documento.

Inverse Document Frequency (IDF): Es la medida de cuán importante es una palabra. Aunque una palabra puede aparecer muchas veces en un documento, puede aparecer muchas veces en otros documentos también. Por lo tanto, esta palabra puede ser irrelevante. IDF es una medida de qué tan única es una palabra para un documento.

El valor TF-IDF de una palabra aumenta proporcionalmente a la cantidad de veces que una palabra aparece en el documento, pero se compensa con la frecuencia de la palabra en el corpus, lo que ayuda a ajustar el hecho de que algunas palabras aparecen con más frecuencia en general.

TF-IDF es una de las formas más populares de ponderación de palabras y es extremadamente útil en muchas aplicaciones, especialmente en el procesamiento del lenguaje natural y la recuperación de información.

B. Stemming y lematización

Stemming y lematización son dos enfoques fundamentales en el procesamiento del lenguaje natural destinados a reducir las formas morfológicas y, a veces, las formas flexionadas de una palabra a una base común. Estos procedimientos se utilizan para normalizar los textos, lo que es fundamental para que las tareas de recuperación de información y análisis de texto sean efectivas.

Stemming es el proceso heurístico que recorta el final de las palabras para lograr mejor normalización. El stemming se enfoca en eliminar los sufijos de una palabra para obtener su raíz. Por ejemplo, las palabras "running", "runner", "ran" se derivan de la raíz "run". Sin embargo, este enfoque puede ser problemático, ya que, a veces, las raíces derivadas no son palabras léxicas reales en un idioma.

Por otro lado, la lematización es un enfoque más estructurado y se considera una parte más profunda del análisis del lenguaje natural. Este proceso tiene en cuenta el contexto y la parte de la oración en la que se encuentra la palabra para convertirla en su forma base, conocida como "lema". Por ejemplo, la lematización reconoce que las palabras "mejor" y "bueno" son formas derivadas del mismo lema "bueno", aunque "mejor" no se obtendría a través de la técnica de stemming.

Aunque el stemming es computacionalmente más barato y más rápido, puede ser inexacto debido a su heurística. La lematización, por otro lado, es más precisa ya que considera el análisis morfológico completo de las palabras, sin embargo, es computacionalmente costosa y puede ser más lenta.

C. Naive Bayes

El algoritmo Naive Bayes es un método probabilístico supervisado de clasificación de máquina de aprendizaje que se basa en el teorema de Bayes. Este algoritmo adquiere su adjetivo "ingenuo" de la suposición de que las características de un objeto de estudio son mutuamente independientes, es decir, la presencia de una característica particular no está de ninguna manera relacionada con la presencia de otra característica.

El teorema de Bayes proporciona una forma de calcular la probabilidad posterior P(C|X), que es la probabilidad de que un objeto caiga en una clase C dadas sus características X. La fórmula para el teorema de Bayes es:

$$P(C|X) = \frac{P(X|C)*P(C)}{P(X)}$$

donde:

- P(C|X) es la probabilidad posterior, es decir, la probabilidad de la clase dadas las características.
- P(X|C) es la verosimilitud, es decir, la probabilidad de las características dada la clase.
- P(C) es la probabilidad previa de la clase.
- P(X) es la probabilidad previa de las características.

El algoritmo Naive Bayes es ampliamente utilizado en diversas aplicaciones, incluyendo la clasificación de documentos, el análisis de sentimientos, el filtrado de spam y los sistemas de recomendación, debido a su simplicidad, velocidad y rendimiento eficiente, especialmente en conjuntos de datos de alta dimensión. Además, puede manejar tanto características categóricas como numéricas y proporciona estimaciones de probabilidad en lugar de simplemente decisiones de clasificación. Sin embargo, su suposición de independencia puede limitar su rendimiento en casos donde esta suposición se viola significativamente.

D. Árboles de Decisión

Los árboles de decisión son una importante familia de algoritmos de aprendizaje supervisado utilizados en la clasificación y regresión. La idea central detrás de los árboles de decisión es la segmentación recursiva del espacio de características para llegar a decisiones sobre los datos de entrada.

Un árbol de decisión es una estructura de árbol en la que cada nodo interno representa una condición en una característica, cada rama representa el resultado de la prueba y cada nodo hoja representa una predicción de clase o un valor continuo, dependiendo de si la tarea es de clasificación o regresión.

La segmentación del espacio de características se realiza de manera que se maximice la pureza de los nodos. Diversas medidas de pureza o impureza se utilizan, incluyendo el índice de Gini, la entropía y la ganancia de información.

Los árboles de decisión son muy apreciados debido a su interpretabilidad, ya que las decisiones del árbol se pueden visualizar y entender fácilmente. Sin embargo, tienden a ser inestables, ya que pequeñas variaciones en los datos pueden resultar en un árbol de decisión diferente.

E. Bosque Aleatorio

El algoritmo de Bosque Aleatorio (*Random Forest*) es un método de aprendizaje supervisado que opera construyendo una multitud de árboles de decisión en el momento del entrenamiento y generando la clase o la predicción media de los árboles individuales. Los bosques aleatorios son una combinación de aprendices de árboles que se entrenan con el método de "bagging".

El concepto central del método de bagging es que la combinación de modelos de aprendizaje aumenta el resultado general. La idea es que varios árboles débiles trabajen juntos para proporcionar una predicción robusta.

En un Bosque Aleatorio, solo un subconjunto aleatorio de las características se considera para la división de cada nodo de los árboles de decisión. También se introduce aleatoriedad en los datos de entrenamiento que se utilizan para construir los árboles, donde cada árbol se entrena con una muestra de bootstrap del conjunto de datos de entrenamiento.

El algoritmo de Bosque Aleatorio tiene varias ventajas. Tiene una alta precisión y funciona bien en muchas configuraciones y tipos de datos, incluso con características categóricas y valores faltantes. También puede manejar muy bien los conjuntos de datos de alta dimensión, y no tiende a sobreajustar en la misma medida que, por ejemplo, los árboles de decisión no restringidos.

F. K-Vecinos más Cercanos

El algoritmo de K-Vecinos más Cercanos (*K-Nearest Neighbors*, K-NN) es un método no paramétrico utilizado tanto para la clasificación como para la regresión. Es un método basado en instancias, lo que significa que el modelo no aprende explícitamente un modelo; en cambio, se basa en la memoria de las instancias de la base de datos de entrenamiento.

La premisa de K-NN es que los objetos similares se encuentran en la proximidad unos de otros en el espacio de características. En la clasificación, un objeto se clasifica según la mayoría de votos de sus vecinos, siendo el objeto asignado a la clase más común entre sus K vecinos más cercanos.

La elección del número K de vecinos es un factor crucial que puede influir significativamente en la calidad del algoritmo K-NN. Un K pequeño puede hacer que el modelo sea sensible al ruido, mientras que un K grande puede suavizar demasiado las fronteras de decisión. En la práctica, se utiliza la validación cruzada para encontrar un valor óptimo de K.

Además, la elección de la métrica de distancia es otro aspecto esencial en K-NN. La distancia euclidiana es la más comúnmente utilizada, pero otras métricas, como la distancia de Manhattan, la distancia de Minkowski, o la distancia de Mahalanobis, también pueden ser apropiadas dependiendo del contexto.

G. Máquinas de Soporte Vectorial

Las Máquinas de Soporte Vectorial (*Support Vector Machines*, SVM) son un conjunto de métodos de aprendizaje supervisado utilizados para la clasificación y regresión. Sin embargo, se utilizan principalmente en problemas de clasificación. El algoritmo de SVM crea un hiperplano o conjunto de hiperplanos en un espacio de alta o infinita dimensión.

En un problema de clasificación binaria, el objetivo de una SVM es encontrar el hiperplano que tiene el margen más grande, es decir, la distancia máxima entre los puntos de datos de ambas clases. Los puntos de datos que están más cerca del hiperplano se conocen como vectores de soporte.

SVM también puede manejar la clasificación no lineal utilizando un kernel. *Kernel* es una función que transforma los datos de entrada en un espacio de características de alta dimensión donde se puede encontrar un hiperplano que separe los datos. Los kernels comunes incluyen el kernel lineal, polinomial y de base radial gaussiana (RBF).

Las SVM pueden ser sensibles a la elección del kernel y los parámetros del kernel, y el entrenamiento puede ser costoso en términos de tiempo y memoria para conjuntos de datos grandes. Además, aunque la formulación de optimización de SVM garantiza la obtención del mínimo global, la interpretación de los modelos puede ser difícil, especialmente en el caso de los kernels no lineales.

H. Redes neuronales

Las redes neuronales artificiales son una familia de modelos de aprendizaje automático inspirados en las redes neuronales biológicas que constituyen los cerebros animales. Se utilizan para estimar o aproximar funciones que pueden depender de una gran cantidad de entradas y son generalmente desconocidas.

Una red neuronal está compuesta por una colección de unidades de procesamiento, llamadas neuronas. Estas unidades están organizadas en capas: una capa de entrada que recibe las señales (es decir, las características de entrada), una o más capas ocultas, donde se realiza el procesamiento real, y una capa de salida que produce el resultado.

Cada nodo en una capa está conectado a todos los nodos en la

capa siguiente. Estas conexiones no son todas iguales: cada una tiene un peso que determina la importancia de la entrada correspondiente. Estos pesos son los parámetros que la red aprende durante el entrenamiento.

El entrenamiento de una red neuronal implica ajustar los pesos de las conexiones de manera iterativa para minimizar una función de pérdida. Este proceso se realiza a través de un algoritmo de optimización

Las redes neuronales pueden modelar relaciones complejas y no lineales entre las entradas y la salida, y son particularmente útiles para tareas donde las características de entrada se encuentran en formas de matriz naturales.

Sin embargo, las redes neuronales pueden ser computacionalmente intensivas, requieren grandes cantidades de datos para entrenarse eficazmente, y pueden ser difíciles de interpretar debido a su naturaleza de "caja negra". Además, la elección de la arquitectura de la red (número de capas ocultas, número de nodos en cada capa, etc.) y los parámetros de entrenamiento (tasa de aprendizaje, número de épocas, etc.) pueden influir significativamente en el rendimiento del modelo.

H. Sobreajuste y dropout

El sobreajuste es un fenómeno que se produce cuando un modelo aprende demasiado bien los detalles y el ruido en el conjunto de datos de entrenamiento, hasta el punto de que se desempeña mal al predecir resultados en nuevos conjuntos de datos no vistos. Es un problema común en las redes neuronales dada su capacidad para aprender representaciones de datos complejas y de alta dimensión.

La técnica de *dropout* es una medida para contrarrestar el sobreajuste. Aleatoriamente desactiva las salidas de algunas neuronas durante la fase de entrenamiento. La probabilidad de desactivación de cada neurona es un hiper parámetro que debe definirse antes del entrenamiento.

Durante el proceso de entrenamiento, en cada iteración, ciertas neuronas son seleccionadas al azar y eliminadas temporalmente de la red, junto con todas sus conexiones de entrada y salida. Esto se asemeja a entrenar un conjunto de redes neuronales distintas, y se ha demostrado que mejora la robustez de la red al reducir la dependencia de la red en cualquier neurona o conjunto de neuronas específicas.

IV. METODOLOGÍA

A. Recopilación de datos.

La recopilación de datos se realizó utilizando el *dataset* disponible en Kaggle titulado "*AG News Classification Dataset*" [5]. Este dataset proporciona una colección de artículos de noticias, escritas en inglés, de diferentes categorías que son relevantes para nuestro estudio de clasificación multiclase de artículos de noticias.

El *dataset* contiene 127,600 artículos de noticias etiquetados en cuatro categorías: internacional, deportes, negocios y ciencia/tecnología. Existe una división preestablecida de conjuntos para entrenamiento, validación y evaluación, los cuales corresponden con los archivos train.csv, val.csv y test.csv respectivamente. Cada artículo de noticia está representado por un título y un texto asociado, así como por su categoría. Para garantizar la calidad y relevancia de los datos, se realizaron los siguientes pasos de selección y preprocesamiento:

1. Descarga del *dataset*: El *dataset* fue descargado directamente desde el enlace proporcionado en Kaggle. Posteriormente, se subieron los archivos al repositorio del proyecto, lo cual facilita su importación cada vez que se requiere trabajar con ellos.
2. Exploración y comprensión: Se realizó una exploración inicial del *dataset* para familiarizarse con su estructura y contenido. Se examinaron las columnas disponibles, como *Title* y *Description*, que contienen la información textual relevante para la clasificación de noticias.
3. Selección de columnas: Para nuestro estudio, utilizaremos las columnas *Title* y *Description* como características para la clasificación. Posteriormente, estas columnas se concatenaron en un atributo *Text,* lo cual permite emplear únicamente esta última columna en la implementación de modelos.
4. Limpieza y preprocesamiento: Se aplicaron técnicas de preprocesamiento para limpiar los textos y prepararlos para el análisis. Esto incluyó la definición y muestra de los *dataframes* para entrenamiento, la eliminación de registros duplicados según el campo *Description*, así como su ordenamiento ascendente. Además, se realizaron técnicas de eliminación de cadenas de espacios, reconfiguración de índices, validación de tamaños, la eliminación de puntuación y conversión a minúsculas.

B. Preparación de datos.

Se realizó una limpieza de los textos con el objetivo de eliminar elementos innecesarios y ruido que pudieran afectar el rendimiento de los algoritmos. Para esto, se aplicaron los siguientes procesos:

1. Tokenización y lematización: Se utilizó la biblioteca NLTK (*Natural Language Toolkit*) [6], para llevar a cabo la tokenización de los textos, dividiéndolos en palabras individuales. Posteriormente, se aplicó la lematización a las palabras utilizando el *WordNet Lemmatizer* de NLTK. Esto ayudó a reducir las palabras a su forma base o lema, considerando el contexto gramatical de las oraciones.
2. Eliminación de *stopwords*: Se utilizó el conjunto de *stopwords* proporcionado por NLTK para eliminar palabras comunes que no aportan información relevante para la clasificación de los artículos de noticias. Estas palabras, como pronombres, preposiciones y conjunciones, se consideran ruido y pueden afectar negativamente la precisión de los algoritmos.

Es importante destacar que, en el caso de los algoritmos de redes neuronales, no se realizaron procesos específicos de preparación de datos en esta etapa. Esto se debe a que las redes neuronales son capaces de aprender representaciones de datos más complejas y no requieren los mismos pasos de preprocesamiento que los algoritmos mencionados anteriormente. Por lo tanto, los datos preparados en el preprocesamiento se utilizarán directamente en el entrenamiento de las redes neuronales.

La preparación de datos realizada garantiza que los textos de los artículos de noticias se encuentren en una forma adecuada para el análisis TF-IDF y la aplicación de los algoritmos de aprendizaje supervisado seleccionados. Al eliminar palabras irrelevantes, lematizar y normalizar los textos, se mejora la calidad de los datos de entrada, lo que puede influir positivamente en el rendimiento de los algoritmos en la tarea de clasificación multiclase de los artículos de noticias.

C. Elección de algoritmos.

Los algoritmos seleccionados para el análisis de rendimiento de la clasificación multiclase se escogieron con base en las siguientes bondades:

- Relevancia y aplicabilidad: Los algoritmos seleccionados son relevantes y adecuados para abordar la problemática

de la clasificación multiclase de artículos de noticias. Estos algoritmos han sido ampliamente utilizados y estudiados en tareas de clasificación de textos y su aplicabilidad en campos relacionados como la minería de opiniones, el procesamiento del lenguaje natural y la clasificación de documentos.

- Efectividad demostrada: Los estudios previos e investigaciones mencionadas en la sección "Estado de la cuestión" han demostrado la efectividad de los algoritmos seleccionados en la clasificación de textos.
- Características de los algoritmos: Como se mencionó en el primer punto, los algoritmos son adecuados para el presente estudio. Por ejemplo, el algoritmo Naive Bayes es conocido por su simplicidad y capacidad para manejar grandes volúmenes de datos, mientras que los Árboles de Decisión ofrecen interpretabilidad y la capacidad de manejar características no lineales. También, los algoritmos Máquinas de Soporte Vectorial (SVM) y K-Vecinos más Cercanos (KNN), son efectivos en espacios de alta dimensionalidad, ambos métodos están basados en instancias que pueden ser útiles para clasificar textos similares. Por último, el algoritmo Bosques Aleatorios es conocido por su capacidad de manejar características no lineales y la reducción del sobreajuste.
- Complementariedad y diversidad: La combinación de diferentes algoritmos puede aportar una perspectiva más completa y diversa al análisis. Al utilizar varios algoritmos, se pueden aprovechar las fortalezas individuales de cada uno y mitigar las posibles limitaciones. Por ejemplo, algunos algoritmos pueden ser mejores para la clasificación de ciertos tipos de noticias, mientras que otros pueden ser más efectivos en diferentes escenarios.
- Comparabilidad y análisis comparativo: La elección de múltiples algoritmos permite realizar un análisis comparativo exhaustivo y objetivo. Al evaluar y comparar el rendimiento de cada algoritmo en términos de métricas de evaluación como precisión, *recall*, *F1-score* y *support* se pueden identificar las fortalezas y debilidades de cada enfoque.

D. Implementación y evaluación.

Para llevar a cabo las pruebas de eficiencia de los algoritmos criptográficos, se utilizó *Google Colab*, una plataforma en la nube que proporciona acceso gratuito a tiempos de ejecución de *GPU* y *TPU* por hasta 12 horas. Durante las pruebas, se utilizó el tiempo de ejecución de *GPU* de Colab, que cuenta con CPU *Intel Xeon* a 2.20 GHz, 13 GB de *RAM*, acelerador *Tesla K80* y 12 GB de *VRAM GDDR5*; además de 78 GB de almacenamiento en disco. Con respecto al software, el lenguaje de programación es Python 3.10. Adicionalmente, se utilizaron múltiples bibliotecas de desarrollo y procesamiento de lenguaje natural, entre las más destacadas están *Scikit-learn*, *Tensorflow*, nltk, numpy y pandas.

Scikit-learn es una biblioteca de *Python* de código abierto que proporciona una variedad de herramientas supervisadas y no supervisadas para el análisis de datos y la modelización predictiva. Es conocida por su claridad, eficiencia y calidad del código, y es ampliamente utilizada en la comunidad de aprendizaje automático por su amplia gama de algoritmos y utilidades convenientes para el preprocesamiento de datos, la selección de modelos y la evaluación.

TensorFlow es una biblioteca de software de código abierto desarrollada por *Google Brain Team*. Se utiliza principalmente para tareas de aprendizaje profundo y proporciona un conjunto de primitivas de nivel bajo y de alto nivel para la definición y optimización de modelos de aprendizaje automático. *TensorFlow* permite el cálculo de tensores con múltiples dimensiones, lo que lo hace adecuado para implementar redes neuronales.

Natural Language Toolkit (NLTK) es una biblioteca de *Python* destinada al procesamiento de lenguaje natural (PLN). Proporciona interfaces fáciles de usar para más de 50 cuerpos y léxicos de texto, así como bibliotecas para el procesamiento de texto para la clasificación, tokenización, derivación, etiquetado, análisis y razonamiento semántico.

NumPy es una biblioteca fundamental para la computación científica en *Python*. Proporciona un objeto de matriz multidimensional, diversos objetos derivados (tales como matrices enmascaradas y matrices) y un surtido de rutinas para operaciones rápidas en matrices, incluyendo matemáticas, lógica, manipulación de formas, entre otras.

Pandas es una biblioteca de software escrita para la manipulación y el análisis de datos en *Python*. Ofrece estructuras de datos y operaciones para manipular tablas numéricas y series temporales. *Pandas* es de uso libre bajo la licencia de tres cláusulas *BSD*.

Una vez implementados los algoritmos, el siguiente paso es evaluar su rendimiento. Como se mencionó previamente, las métricas de evaluación a utilizar serán precisión, *recall*, *F1-score* y *support:*

- La precisión es una métrica de evaluación que indica la proporción de identificaciones positivas que fueron realmente correctas. Se calcula como el número de verdaderos positivos dividido por la suma de verdaderos positivos y falsos positivos.
- El *recall* (o sensibilidad) es una métrica de evaluación que indica la proporción de positivos reales que se identificaron correctamente. Se calcula como el número de verdaderos positivos dividido por la suma de verdaderos positivos y falsos negativos.
- El *F1-score* es una métrica de evaluación que proporciona una medida única de la precisión y el *recall* de un modelo. Es la media armónica de la precisión y el *recall*, lo cual significa que para obtener un *F1-score* alto, tanto la precisión como el *recall* deben ser altos.
- El *support* es el número de ocurrencias de cada clase en el conjunto de datos verdaderos. En la evaluación de la clasificación, el *support* para cada etiqueta proporciona un contexto sobre cómo se distribuyen las instancias verdaderas entre las clases.

Por otro lado, se tomará el tiempo de entrenamiento requerido por cada uno de los algoritmos. En muchos casos, el tiempo de entrenamiento está directamente relacionado con la eficiencia computacional del algoritmo. Un algoritmo que tarda mucho en entrenarse puede ser prohibitivamente costoso en términos de recursos computacionales, lo cual es especialmente relevante cuando se trabaja con grandes volúmenes de datos. Además, un tiempo de entrenamiento más corto permite ciclos de iteración más rápidos, lo que puede permitir un desarrollo y una mejora más rápidos del modelo.

También es importante considerar los costos de infraestructura. En general, el entrenamiento de modelos de aprendizaje automático requiere recursos computacionales que pueden ser costosos, especialmente cuando se utiliza la nube o *hardware* especializado, como es el caso de este proyecto. Por lo tanto, el tiempo de entrenamiento puede tener un impacto directo en los costos de infraestructura.

E. Análisis comparativo.

Para llevar a cabo el análisis comparativo de los algoritmos se tomarán en cuenta las métricas de evaluación previamente descritas. Con el objetivo de visualizarlas de forma sencilla, se utilizará el método *Classification Report* para cada una de las implementaciones. Esta función genera una tabla con la precisión, *recall*, *F1-score* y *support* global de los algoritmos, así como un desglose de las mismas para cada una de las categorías de noticias.

En adición a los reportes, se generaron matrices de confusión. Una matriz de confusión es una herramienta de visualización y evaluación de los resultados obtenidos por un modelo de clasificación. Esta matriz proporciona información detallada sobre el rendimiento del modelo al distinguir entre cada una de las clases en el conjunto de datos.

Para tareas de clasificación multiclase, la matriz sería de tamaño n x n, siendo n el número de clases (n = 4 para el presente trabajo). En este caso, los elementos diagonales representan las instancias que fueron correctamente clasificadas para cada clase, mientras que los elementos fuera de la diagonal representan las instancias que fueron incorrectamente clasificadas.

La matriz de confusión proporciona una visión más detallada del rendimiento del modelo que las métricas de evaluación estándar como la precisión, el recall o el F1-score, ya que permite identificar no solo cuántas instancias se clasificaron incorrectamente, sino también qué tipos de errores de clasificación se están cometiendo.

V. RESULTADOS

A. Muestra de resultados.

A continuación, se muestran los reportes de clasificación para cada uno de los algoritmos en forma de tablas. Posteriormente, se incluyen gráficas de los tiempos de entrenamiento y predicción, para una mejor visualización. Adicionalmente, se incluyen las matrices de confusión, tanto de las redes neuronales como del resto de algoritmos. En la siguiente sección se realiza el análisis de los resultados presentados.

	Red Neuronal sin Dropout			
TIPO	**precision**	**recall**	**f1-score**	**support**
Internacional	0.91	0.92	0.91	1865
Deportes	0.97	0.96	0.96	1906
Negocios	0.87	0.87	0.87	1882
Ciencia	0.90	0.88	0.89	1941
accuracy	- -	- -	0.91	7594
macro avg	0.91	0.91	0.91	7594
weighted avg	0.91	0.91	0.91	7594

Tabla 1. Métricas de la red Neuronal sin Dropout.

	Red Neuronal con Dropout			
TIPO	**precision**	**recall**	**f1-score**	**support**
Internacional	0.9	0.95	0.92	1796
Deportes	0.98	0.96	0.97	1933
Negocios	0.89	0.87	0.88	1947
Ciencia	0.9	0.89	0.89	1918
accuracy	- -	- -	0.92	7594
macro avg	0.92	0.92	0.92	7594
weighted avg	0.92	0.92	0.92	7594

Tabla 2. Métricas de la red neuronal con Dropout

	Naive Bayes			
TIPO	**precision**	**recall**	**f1-score**	**support**
Internacional	0.91	0.9	0.9	7448
Deportes	0.95	0.98	0.96	7445
Negocios	0.87	0.87	0.87	7357
Ciencia	0.89	0.87	0.88	7342
accuracy	- -	- -	0.91	29692
macro avg	0.9	0.91	0.9	29692
weighted avg	0.9	0.91	0.9	29692

Tabla 3. Métricas de Naive Bayes.

	Árbol de Decisión			
TIPO	**precision**	**recall**	**f1-score**	**support**
Internacional	0.8	0.8	0.8	7448
Deportes	0.85	0.87	0.86	7445
Negocios	0.75	0.75	0.75	7457
Ciencia	0.75	0.73	0.74	7342
accuracy	- -	- -	0.79	29692
macro avg	0.79	0.79	0.79	29692
weighted avg	0.79	0.79	0.79	29692

Tabla 4. Métricas de Árbol de decisión.

	Support Vector Machines			
TIPO	**precision**	**recall**	**f1-score**	**support**
Internacional	0.93	0.9	0.92	7448
Deportes	0.96	0.98	0.97	7445
Negocios	0.89	0.89	0.89	7457
Ciencia	0.89	0.9	0.9	7342
accuracy	- -	- -	0.92	29692
macro avg	0.92	0.92	0.92	29692
weighted avg	0.92	0.92	0.92	29692

Tabla 5. Métricas de Support Vector Machines.

	K-Nearest Neighbors (KNN)			
TIPO	**precision**	**recall**	**f1-score**	**support**
Internacional	0.9	0.89	0.9	7448
Deportes	0.94	0.97	0.85	7445
Negocios	0.86	0.87	0.87	7457
Ciencia	0.89	0.85	0.87	7342
accuracy	- -	- -	0.9	29692
macro avg	0.9	0.9	0.9	29692
weighted avg	0.9	0.9	0.9	29692

Tabla 6. Métricas de KNN.

	Random forest			
TIPO	**precision**	**recall**	**f1-score**	**support**
Internacional	0.92	0.88	0.9	7448
Deportes	0.92	0.97	0.95	7445
Negocios	0.87	0.86	0.86	7457
Ciencia	0.87	0.86	0.86	7342
accuracy	- -	- -	0.89	29692
macro avg	0.89	0.89	0.89	29692
weighted avg	0.89	0.89	0.89	29692

Tabla 7. Métricas de Random forest.

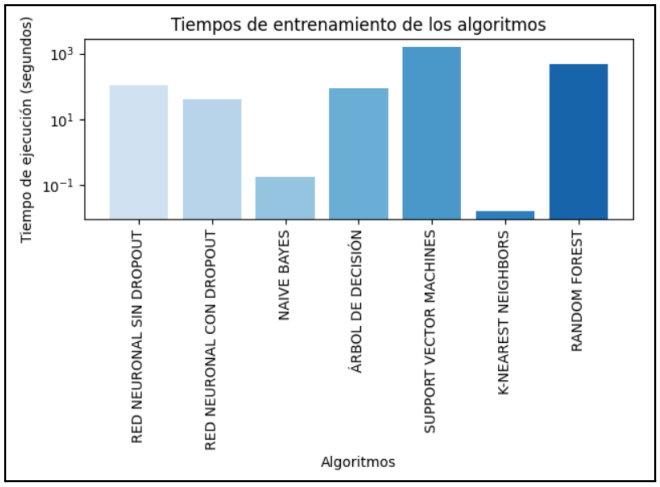

Figura 1. Tiempos de entrenamiento.

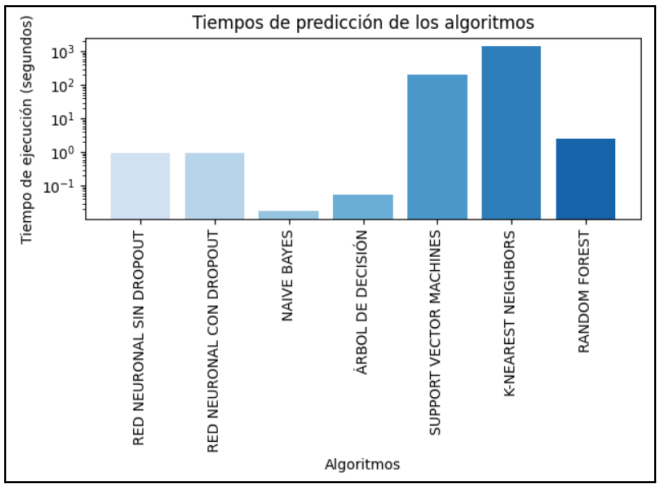

Figura 2. Tiempos de predicción.

Red Neuronal Sin Dropout

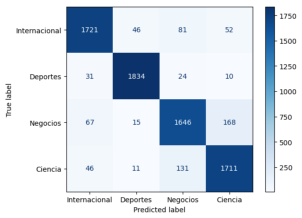

Figura 3. Matriz de confusión para Red Neuronal sin Dropout.

Red Neuronal Con Dropout

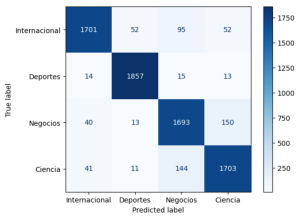

Figura 4. Matriz de confusión para Red Neuronal con Dropout.

Naive Bayes

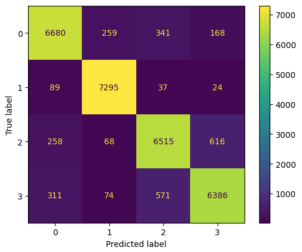

Figura 5. Matriz de confusión para Naive Bayes.

Árbol de Decisión

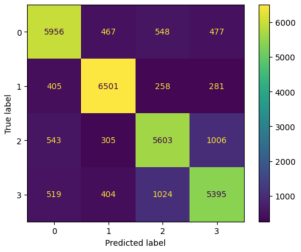

Figura 6. Matriz de confusión para Árbol de Decisión.

SVM

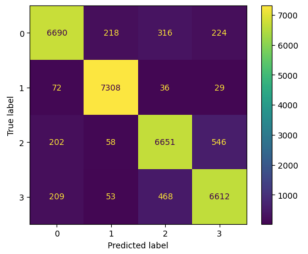

Figura 7. Matriz de confusión para SVM.

KNN

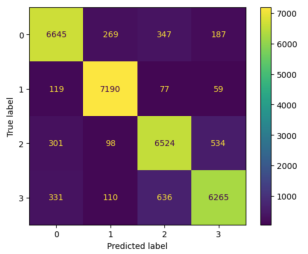

Figura 8. Matriz de confusión para KNN.

Random Forest

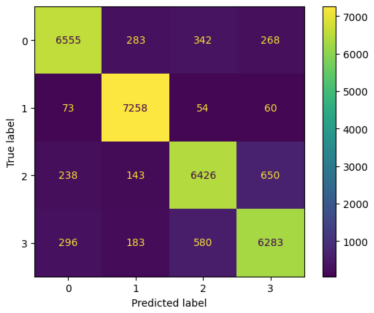

Figura 9. Matriz de confusión para Random Forest.

B. Análisis de resultados.

Las tablas y gráficas de la sección anterior nos presentan resultados sumamente interesantes. Por ejemplo, previo a la implementación de los algoritmos, se esperaba que las redes neuronales tuvieran el mayor tiempo de entrenamiento. No obstante, se encontró que estos tiempos eran muy similares, e incluso inferiores, a los de algoritmos como SVM o random forest. Además, sus tiempos de predicción o evaluación son mucho menores al de la mayoría de los algoritmos.

Continuando con el análisis de las redes neuronales, es importante destacar las diferencias que existen entre ambas. Una de ellas cuenta con la capa de *dropout,* la cual, como se explicó previamente, permite evitar el sobre ajuste del modelo. A través de la modificación y prueba de parámetros en múltiples pruebas de entrenamiento, se encontró que dicha capa sí influyó en su comportamiento. La precisión de la red neuronal sin dropout, en el conjunto de validación, fue inversamente proporcional a la cantidad de épocas o iteraciones que se aplicaron durante su entrenamiento. Por otro lado, la precisión en el conjunto de entrenamiento sí continuaba incrementando, lo cual implica un claro sobreajuste. Para mantener las métricas de evaluación tan altas como fuera posible en el conjunto de prueba, se decidió reducir la cantidad de épocas en su entrenamiento, teniendo una precisión final de 91%.

En contraste, la red neuronal con dropout mantuvo una precisión más constante en el conjunto de validación, a pesar de tener peores métricas en el entrenamiento. Tras solo dos épocas, se obtuvo una precisión del 92% en el conjunto de evaluación. Es importante destacar que ambas redes neuronales se diferencian, también, por las características de sus capas, pues la cantidad de neuronas en ellas es distinta. Esto se decidió así, tras muchas pruebas, para que ambas ofrecieran el mejor rendimiento posible según sus características.

Otro punto a tener en cuenta para las redes neuronales, es que su consumo de recursos computacionales es muy alto. Si bien sus métricas de evaluación son mejores a la mayoría de algoritmos, la diferencia no es tan grande como se esperaba. Al respecto, cabe remarcar que esto puede deberse a las características del conjunto de datos. Si bien las redes neuronales tienen aplicaciones sumamente importantes en el análisis de textos y han revolucionado el procesamiento de lenguaje natural, emplearlas para una aplicación y datasets como estos puede considerarse "matar moscas a cañonazos".

Sobre el resto de algoritmos, uno de los mejor balanceados, en cuanto a rendimiento y eficiencia se refiere, fue *Naive Bayes*. Cuenta con el promedio de tiempo de ejecución más bajo de entre todos los algoritmos implementados, al tiempo que mantiene una precisión general del 90%. Si bien se esperaba que tuviera buenas métricas, la realidad ha superado las expectativas sobre este algoritmo.

Las máquinas de soporte vectorial tuvieron, junto con las redes neuronales, las mejores métricas de evaluación. Esto tiene sentido, ya que son el algoritmo más cercano a las redes en cuanto a funcionamiento se refiere. Sin embargo, a pesar de su gran precisión, cercana al 92%, sus tiempos de entrenamiento y predicción fueron los más altos de todos. En consecuencia, se considera que su rendimiento fue peor al de sus parientes lejanos, las redes neuronales.

Los algoritmos basados en estructuras tipo árbol, árboles de decisión y bosques aleatorios, tuvieron las métricas esperadas. Dado que un bosque se conforma de múltiples árboles, 100 en nuestro caso, su precisión fue un 10% mejor, pero su tiempo de entrenamiento, al igual que el de predicción, también aumentó considerablemente. Aún cuando estos resultados demuestran que no son el mejor algoritmo para esta tarea, al mismo tiempo prueban que los árboles de decisión trabajan mejor en conjunto que en solitario.

Por último, el algoritmo que arrojó los resultados más sorprendentes fue *K-vecinos más próximos* (KNN). Su precisión estuvo muy cerca del promedio del resto, alrededor del 90%, pero sus tiempos fueron muy distintos, ya que el de predicción fue el más largo y el de entrenamiento el más corto. Esto se debe a que KNN es un algoritmo que no requiere un proceso de entrenamiento en el sentido tradicional. En lugar de aprender parámetros o estructuras de un conjunto de entrenamiento, KNN simplemente almacena todos los puntos de datos de entrenamiento en la memoria. Por lo tanto, el "tiempo de entrenamiento" para K-vecinos más próximos es corto, ya que simplemente implica almacenar los datos.

Por otro lado, el proceso de predicción con KNN puede ser costoso en términos de tiempo. Cuando se realiza una predicción, el algoritmo KNN debe comparar el nuevo punto de datos con todos los puntos de datos almacenados durante el "entrenamiento". Este proceso implica calcular la distancia entre el nuevo punto de datos y cada punto de datos de entrenamiento para identificar los k vecinos más cercanos. Luego, la predicción se realiza basándose en las etiquetas de estos k vecinos.

Por lo tanto, a medida que el tamaño del conjunto de datos de entrenamiento aumenta, el tiempo de predicción también aumenta, ya que el algoritmo necesita comparar el nuevo punto de datos con un número creciente de puntos de datos de entrenamiento. Esto es en contraste con los otros algoritmos de aprendizaje automático, donde el tiempo de entrenamiento es el que se ve más afectado por el tamaño del conjunto de datos de entrenamiento, pero el tiempo de predicción permanece relativamente constante independientemente del tamaño del conjunto de datos de entrenamiento.

Finalmente, analizando las matrices de confusión, hay una clara tendencia, en todos los algoritmos, de tener un peor desempeño en las noticias correspondientes a negocios y ciencia. Esto puede tener diferentes causas, como la calidad de los datos correspondientes a esas categorías, pero una de las más probables es la cantidad y proporción de los datos. El conjunto de datos de entrenamiento final, tras los métodos de limpieza y procesamiento, se conformó de la siguiente manera: 27871 noticias internacionales, 27911 noticias deportivas, 27642 noticias de negocios y 27749 noticias correspondientes a ciencia y tecnología. En contraste, el conjunto de evaluación tuvo una distribución prácticamente uniforme. Por lo tanto, es posible que los algoritmos no aprendieran lo suficiente sobre las últimas dos categorías o tuvieran un sesgo hacia las dos primeras.

VI. CONCLUSIONES

En conclusión, este proyecto tuvo como objetivo comparar la precisión y el rendimiento de diferentes algoritmos de aprendizaje supervisado en la clasificación multiclase de artículos de noticias. Se evaluaron los algoritmos Naive Bayes, Árboles de Decisión, Máquinas de Soporte Vectorial (SVM), K-Vecinos más Cercanos (KNN), Bosque Aleatorio y Redes Neuronales.

Los resultados obtenidos mostraron que las redes neuronales alcanzaron la mayor precisión entre todos los algoritmos evaluados, lo cual era esperado a priori. Sin embargo, también se confirmó que las redes neuronales requirieron más tiempo de entrenamiento en comparación con otros algoritmos. Por otro lado, el algoritmo Naive Bayes demostró ser rápido y mantener una precisión decente.

El uso del análisis TF-IDF como parte del proceso de clasificación permitió una mejor comprensión y representación de los artículos de noticias, lo que influyó en el rendimiento y la precisión de los algoritmos de aprendizaje supervisado.

En cuanto a mejoras para el proyecto, se podrían considerar las siguientes:

Explorar y evaluar otros algoritmos de clasificación multiclase que no se incluyeron en este estudio. Existen constantemente nuevos enfoques y técnicas en el campo del aprendizaje automático que podrían ofrecer mejores resultados.

Realizar un análisis más exhaustivo de los hiperparámetros de cada algoritmo. La configuración óptima de los hiperparámetros puede tener un impacto significativo en el rendimiento de los algoritmos y se podría realizar una búsqueda más exhaustiva para encontrar la configuración óptima en cada caso.

Ampliar el conjunto de datos utilizado. Aunque se utilizó un conjunto de datos de tamaño considerable, agregar más datos podría proporcionar una mejor representación de las categorías de noticias y mejorar la generalización de los modelos.

Considerar el uso de técnicas de aprendizaje automático profundo, como modelos de lenguaje pre-entrenados (por ejemplo, BERT, GPT) o arquitecturas avanzadas de redes neuronales. Estas técnicas han demostrado un rendimiento sobresaliente en tareas de clasificación de texto y podrían mejorar aún más los resultados obtenidos.

En general, este proyecto ofrece una visión comparativa de diferentes algoritmos de clasificación multiclase aplicados a la clasificación de artículos de noticias. Sin embargo, siempre existen posibilidades de mejora y es importante estar al tanto de los avances en el campo del aprendizaje automático para seguir mejorando la eficiencia y precisión de estos algoritmos en futuros proyectos similares.

VII. REFERENCIAS

[1] T. Joachims, “Text categorization with Support Vector Machines: Learning with many relevant features”, en Machine Learning: ECML-98, Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 137–142. https://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf

[2] Z.-H. Deng, S.-W. Tang, D.-Q. Yang, M. Z. L.-Y. Li, y K.-Q. Xie, “A comparative study on feature weight in text categorization”, en Advanced Web Technologies and Applications, Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 588–597. https://link.springer.com/chapter/10.1007/978-3-540-24655-8_64

[3] F. Sebastiani, “Machine learning in automated text categorization”, ACM Comput. Surv., vol. 34, núm. 1, pp. 1–47, 2002. https://dl.acm.org/doi/pdf/10.1145/505282.505283

[4] Y. Yang, T. Ault, T. Pierce, y C. W. Lattimer, “Improving text categorization methods for event tracking”, en Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, 2000. https://dl.acm.org/doi/pdf/10.1145/345508.345550

[5] A. Anand, “AG News Classification Dataset”, AG News Classification Dataset. 20-abr-2020. https://www.kaggle.com/datasets/amananandrai/ag-news-classification-dataset?select=train.csv

[6] “NLTK : Natural Language Toolkit”, Nltk.org. [En línea]. Disponible en: https://www.nltk.org/. [Consultado: 16-jun-2023].

[7] X. Zhang, J. Zhao, y Y. LeCun, “Character-level Convolutional Networks for Text Classification”, Junbo Zhao, 2016.