UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO

Instituto de Ciencias Básicas e Ingeniería
Área Académica de Computación y Electrónica
Licenciatura en Ciencias Computacionales

TESIS

CLASIFICACIÓN DE DOCUMENTOS DIGITALES MEDIANTE TÉCNICAS DE PROCESAMIENTO DEL LENGUAJE NATURAL Y APRENDIZAJE AUTOMÁTICO

Para obtener el grado en:

Licenciada en Ciencias Computacionales

Presenta:

DIANA DENHY CENTENO GARCÍA

Directores:

Dra. Rosa María Ortega Mendoza

Dr. Félix Agustín Castro Espinoza

Abril, 2026

Universidad Autónoma del Estado de Hidalgo
Instituto de Ciencias Básicas e Ingeniería
School of Engineering and Basic Sciences
Dirección
Office of the Director

Mineral de la Reforma, Hgo., a 4 de mayo de 2026

Número de control: ICBI-D/887/2026
Asunto: Autorización de impresión.

MTRA. OJUKY DEL ROCÍO ISLAS MALDONADO
DIRECTORA DE ADMINISTRACIÓN ESCOLAR DE LA UAEH

Con Título Quinto, Capítulo II, Capítulo V, Artículo 51 Fracción IX del Estatuto General de nuestra Institución, por este medio, le comunico que el Jurado asignado a la egresada de la Licenciatura en Ciencias Computacionales **Diana Denhy Centeno García,** quien presenta el trabajo de titulación **"Clasificación de documentos digitales mediante técnicas de procesamiento del lenguaje natural y aprendizaje automático",** ha decidido, después de revisar fundamento en lo dispuesto en el Título Tercero, Capítulo I, Artículo 18 Fracción IV; dicho trabajo en la reunión de sinodales, **autorizar la impresión del mismo**, una vez realizadas las correcciones acordadas.

A continuación, firman de conformidad los integrantes del Jurado:

Presidente:	Dr. Ricardo Calderón Suárez
Secretario:	Dra. Guadalupe Carmona Arroyo
Vocal:	Dra. Rosa María Ortega Mendoza
Suplente:	Dr. Félix Agustín Castro Espinoza

Sin otro particular por el momento, reciba un cordial saludo.

Atentamente
"Amor, Orden y Progreso"

Dr. Gabriel Vergara Rodríguez
Director del ICBI

GVR/MMM

"Amor, Orden y Progreso"

Ciudad del Conocimiento, Carretera Pachuca-Tulancingo
Km. 4.5 Colonia Carboneras, Mineral de la Reforma, Hidalgo,
México. C.P 42184
Teléfono: 52 (771) 71 720 00 Ext. 40001
direccion_icbi@uaeh.edu.mx, vergarar@uaeh.edu.mx

Resumen

La expansión del entorno digital ha provocado un aumento masivo en la generación de documentos digitales, lo que ha planteado diversos retos en su organización. Este escenario resulta especialmente complejo en repositorios de documentos personales, donde suelen presentar una gran diversidad temática y se organizan y clasifican según criterios propios de cada usuario.

Para afrontar este desafío, se han desarrollado diversas estrategias orientadas a la organización de documentos según su contenido. En este contexto, las disciplinas del procesamiento del lenguaje natural y el aprendizaje automático ofrecen enfoques de clasificación tanto supervisada como no supervisada. En la práctica también se emplean herramientas comerciales de clasificación automática de documentos; sin embargo, estas no suelen adaptarse a las características de los archivos personales, además de implicar costos económicos y, en muchos casos, depender de servidores externos.

El presente estudio propone evaluar y comparar distintas técnicas de representación textual, desde métodos tradicionales como TF-IDF hasta modelos avanzados basados en transformadores, combinados con algoritmos de clasificación no supervisada (K-medias, HDBSCAN y modelado de tópicos) y supervisada (bosques aleatorios, máquinas de vectores de soporte y perceptrón multicapa). La evaluación se realizó utilizando colecciones de texto de referencia y documentos personales con el objetivo de simular un entorno real, empleando métricas como exactitud, F1-macro y coeficiente de silueta.

A partir de los experimentos realizados, se observó que BERTopic destacó por su capacidad para modelar tópicos, mientras que la combinación de TF-IDF con máquinas de vectores de soporte obtuvo el mejor desempeño en la clasificación de documentos. Estos modelos se integraron en una interfaz gráfica de usuario que automatiza la organización documental, permitiendo clasificar nuevos documentos, agrupar aquellos sin categoría y entrenar modelos de clasificación personalizados, manteniendo la estructura del repositorio del usuario.

Finalmente, al probar la interfaz gráfica se obtuvieron resultados favorables. En el caso del agrupamiento con BERTopic, se lograron identificar seis de los siete grupos originales del conjunto de datos personales, generando temáticas coherentes. Por otra parte, en la clasificación de documentos, se alcanzaron valores de exactitud superiores a 0.9. En conjunto, estos resultados respaldan el potencial de las técnicas seleccionadas para la organización de documentos personales en un ambiente local.

Palabras clave: *repositorio de documentos personales, aprendizaje automático supervisado, aprendizaje automático no supervisado, clasificación de documentos*

Abstract

The expansion of the digital environment has led to a massive increase in the generation of digital documents, which has posed various challenges for their organization. This scenario is especially complex in personal document repositories, where documents often exhibit a wide thematic diversity and are organized and classified according to each user's own criteria.

To address this challenge, various strategies have been developed to organize documents based on their content. In this context, the fields of natural language processing and machine learning offer both supervised and unsupervised classification approaches. In practice, commercial automatic document classification tools are also used; however, they do not usually adapt well to the characteristics of personal files, and they involve economic costs and, in many cases, depend on external servers.

This study proposes to evaluate and compare different text representation techniques, ranging from traditional methods such as TF-IDF to advanced transformer-based models, combined with unsupervised classification algorithms (K-means, HDBSCAN, and topic modeling) and supervised methods (random forests, support vector machines, and multilayer perceptron). The evaluation was carried out using benchmark text collections and personal documents with the aim of simulating a real-world environment, employing metrics such as accuracy, macro F1-score, and silhouette coefficient.

Based on the experiments conducted, it was observed that BERTopic stood out for its ability to model topics, while the combination of TF-IDF with support vector machines achieved the best performance in document classification. These models were integrated into a graphical user interface that automates document organization, allowing new documents to be classified, uncategorized ones to be grouped, and custom classification models to be trained, while preserving the user's repository structure.

Finally, testing the graphical interface yielded favorable results. In the case of clustering with BERTopic, six of the seven original groups in the personal dataset were successfully iden-

tified, generating coherent themes. On the other hand, in document classification, accuracy values above 0.9 were achieved. Overall, these results support the potential of the selected techniques for organizing personal documents in a local environment.

Key words: *personal document repositories, supervised machine learning, unsupervised machine learning, document classification*

Índice general

Lista de figuras

Lista de tablas

CAPÍTULO 1

Introducción

La creación de documentos ha permitido a la humanidad preservar sus saberes y pensamientos, así como transmitir su conocimiento a lo largo del tiempo. Gracias al salto tecnológico que hemos vivido, la generación de documentos digitales se ha convertido en una vía esencial para transferir y almacenar contenidos. Sin embargo, esta práctica conlleva una problemática implícita: la producción masiva y descontrolada de información documental, fenómeno que Deleón (2007) denomina con el término *explosión de documentos*. Una situación común tanto en organizaciones como en usuarios individuales, donde generalmente no existe un orden establecido ni un procedimiento definido para la gestión documental. Por lo tanto, la clasificación de los archivos digitales que los usuarios generan en su repositorio personal se ha convertido en un desafío.

Para afrontar esta situación, se han desarrollado mecanismos que buscan mejorar el acceso y la organización de la información. Entre ellos, destaca una técnica conocida como clasificación, la cual implica identificar, agrupar y sistematizar los elementos. En este sentido, los documentos se agrupan según el asunto, tema o contenido, y posteriormente se integran en expedientes, como fólderes o carpetas (Deleón, 2007).

En el área de la Inteligencia Artificial, y particularmente del Procesamiento del Lenguaje Natural (PLN) y aprendizaje automático, se han desarrollado y probado diversas técnicas y algoritmos que han permitido automatizar la clasificación documental. Existen dos enfoques principales aplicados en este contexto: la clasificación supervisada, que se basa en documentos previamente etiquetados, y la clasificación no supervisada, que busca identificar patrones en la información sin necesidad de contar con etiquetas (Antonelli & Guarracino, 2023).

Aunque ambas técnicas han sido ampliamente estudiadas en investigaciones y actualmente existen herramientas comerciales que las implementan, la mayoría no se adapta a las

características organizativas de los archivos digitales de un usuario, donde la cantidad producida suele ser menor y las temáticas muy variadas. Además, algunas de estas soluciones requieren pagos recurrentes y operan mediante servicios externos en la nube, lo que puede representar una limitación para ciertos usuarios.

Por ello, surge el interés de analizar cómo se comportan diferentes técnicas de clasificación documental en un entorno más cercano al uso cotidiano. Se busca responder a la siguiente pregunta: ¿Qué técnicas y algoritmos de aprendizaje automático presentan un mejor desempeño en la clasificación de documentos digitales recopilados del repositorio personal de los usuarios?

Con el propósito de responder a esta interrogante y trasladar los avances en procesamiento del lenguaje natural y aprendizaje automático al contexto planteado, este trabajo propone un estudio exploratorio para evaluar y comparar el desempeño de diversas técnicas de representación textual y algoritmos de aprendizaje supervisado y no supervisado, para abordar la tarea de clasificación documental. Se consideran tanto representaciones tradicionales, como TF-IDF, como enfoques recientes que capturan un significado más profundo del texto, como los modelos basados en transformadores.

Estas representaciones se combinarán con algoritmos de clasificación supervisada, como bosques aleatorios, máquinas de vectores de soporte (SVM) y perceptrón multicapa (MLP), así como con técnicas de agrupamiento no supervisado, como K-medias y HDBSCAN, además de un método de modelado de tópicos. El objetivo es identificar cuáles de estas combinaciones ofrecen un mejor desempeño al aplicarse en la estructura documental de los usuarios.

Se desarrollarán modelos de clasificación de textos utilizando un conjunto de datos externo. Posteriormente, los modelos generados se evaluarán mediante las métricas de desempeño adecuadas para cada tipo de clasificación, tales como exactitud y F1-macro, coeficiente de silueta, entre otros.

Al final, los modelos con mejores resultados se integrarán en una interfaz gráfica de usuario (GUI). El objetivo de esta interfaz es clasificar automáticamente los documentos digitales del usuario, permitiéndole acceder a funcionalidades adaptadas a los distintos tipos de documentos que posee, tanto para ejemplares sin una categoría definida como para aquellos que pueden integrarse a carpetas ya establecidas, manteniendo la estructura interna del repositorio documental. La GUI será probada con una colección reducida de documentos personales con el fin de simular un entorno real de uso. De esta manera, el presente trabajo no solo busca analizar el desempeño de diversas técnicas y modelos de aprendizaje automático

para abordar la tarea de clasificación documental, sino también ofrecer una solución práctica que facilite la organización de documentos digitales en el ámbito personal.

1.1 Planteamiento del problema

La organización manual de documentos personales en formato digital se vuelve una tarea complicada y propensa a errores. A medida que el volumen de archivos crece, un usuario individual puede generar documentos como contratos, facturas, artículos académicos, entre otros, lo que implica una acumulación progresiva de archivos y hace que su organización sea cada vez más desafiante.

Por lo general, una persona que desea organizar los documentos digitales en su computadora personal debe revisarlos uno por uno, identificar su contenido y moverlo manualmente a la carpeta correspondiente, lo cual resulta en un proceso lento e impráctico. Además, la búsqueda de la carpeta adecuada puede volverse compleja, especialmente cuando la estructura del directorio no está bien definida o no existe una identificación clara de los archivos. Estos desafíos han motivado el desarrollo de algunas soluciones disponibles para la resolución de esta tarea.

En el ámbito académico, estas soluciones suelen explorarse a partir entrenar modelos de aprendizaje automático utilizando grandes volúmenes de datos, como aquellos que se encuentran en plataformas institucionales, lo que permite a los modelos capturar patrones más detallados y mejorar su capacidad de generalización (Uddin & Lu, 2024).

Sin embargo, en escenarios con conjuntos de datos a menor escala, los modelos supervisados tienden a presentar dificultades, como el sobreajuste a los datos empleados por falta de ejemplares por clase (Uddin & Lu, 2024). En este sentido, el comportamiento de estos modelos en entornos con poca disponibilidad de archivos ha sido poco explorado.

Algunas herramientas disponibles en el mercado implementan modelos preentrenados mediante servicios web externos que no consideran la información temática ni las categorías propias del usuario. Esto no solo limita la personalización en la organización de documentos, sino que también puede generar desconfianza respecto al manejo y la protección de los datos personales. Un ejemplo que aborda estas inquietudes lo presentan Molina y Tapia (2020) en su artículo: *La tecnología y los riesgos sobre la privacidad y protección de datos*, donde se analiza el impacto de registrar datos personales en plataformas digitales, como el uso indebido y las filtraciones masivas de información.

Además, el público objetivo al que suelen estar dirigidos estos servicios está conformado generalmente por empresas que manejan grandes volúmenes de documentos, por lo que el costo de dichos servicios se puede llegar a justificar dado ese contexto; caso contrario al

volumen en menor escala que un usuario individual genera en tareas cotidianas como la escuela, trabajo, trámites personales, etc.

En resumen, la problemática se define en los siguientes puntos:

1. El proceso de revisar manualmente cada archivo digital para etiquetarlo y clasificarlo requiere tiempo y esfuerzo.
2. La dificultad para identificar categorías adecuadas para cada documento. En ocasiones, las carpetas y los archivos se suelen nombrar de manera intuitiva o arbitraria, lo que contribuye a la desorganización.
3. La falta de herramientas disponibles que permitan proteger la privacidad del usuario y sobre todo, que se adapten a la estructura documental del mismo y sean capaces de encontrar patrones de clasificación.

Para abordar esta problemática, en este trabajo se propone construir modelos de PLN y aprendizaje automático con el fin de crear una herramienta capaz de clasificar documentos según los patrones de clasificación definidos por los usuarios en su repositorio de documentos digitales.

1.2 Antecedentes

En México, la administración de documentos surgió como respuesta ante el grave problema de la producción explosiva de documentos. Con la llegada de la era digital, diversas instituciones públicas empezaron a modernizar este proceso mediante la digitalización de archivos físicos y el uso de programas para la automatización de la gestión documental (Deleón, 2007).

Deleón (2007) explica que la implementación de sistemas digitales favoreció la capacidad del almacenamiento y la preservación de la información. Sin embargo, siguieron persistiendo problemáticas similares en la gestión de documentos tanto en formato físico como digital. Por ejemplo:

- Dificultades para localizar la información.
- Problemas para integrar los documentos dentro de los expedientes correspondientes.
- Insuficiencia de mecanismos para la identificación, clasificación y organización sistemática de la información.
- Riesgo de pérdida de información valiosa, así como limitaciones en su almacenamiento y acceso controlado.

Para abordar esta problemática, diversos autores han aplicado técnicas de aprendizaje automático y PLN para lograr la clasificación de texto, implementando dos enfoques principales: la clasificación supervisada y por otro lado, la clasificación no supervisada.

Entre estos trabajos destaca Sebastiani (2002), quien realizó un compendio de los principales acercamientos en la clasificación textual, en donde discutió problemáticas acerca de la representación documental, la construcción de un clasificador y su evaluación.

En años recientes, se han desarrollado técnicas más avanzadas en el área del PLN, como los modelos basados en transformadores (Vaswani, 2017), los cuales han sido ampliamente usados en tareas de clasificación textual, debido a su capacidad de crear representaciones numéricas que capturan información contextual del texto. Otra técnica muy explorada es el modelado de tópicos, un método que proporciona información valiosa sobre el contenido de un texto (Zhang & Shafiq, 2024).

A pesar de todos estos avances, el comportamiento de los algoritmos de aprendizaje automático en entornos de pequeña escala ha sido menos explorado en comparación con

escenarios de gran escala, lo cual es relevante en contextos como los repositorios de documentos digitales de un usuario individual.

1.3 Justificación

La justificación de este trabajo se sustenta en su aporte académico, su utilidad práctica para usuarios finales y su relevancia como una solución viable para mantener la privacidad del usuario. En el ámbito académico, este proyecto contribuye al área del PLN mediante la exploración y comparación del desempeño de diversos algoritmos de aprendizaje automático aplicados a un dominio específico: documentos digitales en un entorno local. El mayor desafío es enfrentarse con diversos tipos de documentos que un usuario puede manejar en su día a día, lo que implicaría evaluar la robustez de los algoritmos y su capacidad de adaptación a estos dominios.

Desde una perspectiva práctica, el proyecto permite abordar la necesidad de gestionar archivos digitales de diversas temáticas de interés para los usuarios, y surge como una alternativa a la falta de soluciones accesibles y gratuitas en el mercado, lo cual permitiría agilizar los procesos manuales repetitivos (Domínguez et al., 2024), y así reducir la carga de trabajo organizativo documental para cualquier persona que lo desee.

Con la creciente popularidad de herramientas que implementan Inteligencia Artificial, los debates éticos acerca de las políticas de privacidad y el manejo de datos sensibles, han aumentado cuando se trabaja con documentos personales. Dado este contexto, este trabajo resulta relevante, ya que propone el desarrollo de una solución automatizada que funcione localmente, es decir, que un usuario pueda implementarlo desde su propia máquina, sin la necesidad de usar servicios externos, lo cual permitiría mantener la privacidad de la información y preservar la estructura de clasificación que un usuario emplea para organizar sus documentos digitales.

1.4 Propuesta de solución

Se propone un estudio exploratorio de técnicas para la clasificación automática de documentos, combinando diferentes representaciones vectoriales y métodos de aprendizaje automático. El objetivo es identificar las fortalezas y limitaciones de estos algoritmos mediante una evaluación de su desempeño, utilizando métricas acordes al tipo de clasificación. Finalmente, los modelos seleccionados se integrarán en una GUI que opera localmente y que prioriza la privacidad, adaptabilidad y accesibilidad para usuarios.

Se propone la combinación de enfoques supervisados y no supervisados debido a la naturaleza de los documentos personales, donde algunas categorías pueden definirse de forma previa (por ejemplo, facturas o contratos), mientras que otras pueden emerger dinámicamente a partir de las tareas específicas del usuario.

Por ello, la solución propuesta contempla la exploración y evaluación sistemática de las siguientes tareas: preprocesamiento, extracción de características, entrenamiento y evaluación de algoritmos de clasificación supervisada y no supervisada. Estas evaluaciones se realizarán sobre un conjunto de documentos de referencia exclusivo para la etapa de experimentación, con el fin de identificar las configuraciones más adecuadas.

Posteriormente, las técnicas que presenten mejor desempeño, serán integradas en una GUI que permita ejecutar todo el flujo de procesamiento de forma interna. La GUI también será probada utilizando un conjunto de documentos distinto al empleado en la fase experimental; esta colección de dato permitirá simular un entorno real de uso, donde los documentos disponibles son limitados.

Se espera que esta solución reduzca el esfuerzo manual y que el usuario tenga control total sobre su información, sin depender de servicios externos. Sin embargo, los resultados dependen de la calidad de los datos, el adecuado procesamiento del texto y la correcta configuración de los algoritmos utilizados.

De esta forma, el proyecto busca facilitar la recuperación de información en la clasificación documental, aprovechando la diversidad de técnicas existentes en el campo del PLN e integrándolas en una GUI que permita personalizar dichos clasificadores.

1.5 Objetivos

1.5.1 Objetivo general

Desarrollar modelos computacionales para clasificar documentos formales en un entorno local, mediante técnicas de procesamiento de lenguaje natural y algoritmos de aprendizaje automático.

1.5.2 Objetivos específicos

- Construir representaciones textuales que permitan extraer las características más relevantes de un documento para alimentar modelos de aprendizaje automático.
- Implementar modelos de clasificación que sean capaces de aprender el contexto de un conjunto de ejemplares previamente etiquetados y generar predicciones sobre nuevos textos.
- Construir modelos de clasificación no supervisada de textos para identificar grupos de documentos que cumplan características similares usando colecciones de documentos de diversas témáticas.
- Desarrollar una interfaz gráfica de usuario que permita la integración de las técnicas seleccionadas para su uso práctico, garantizando que el procesamiento se realice en el dispositivo del usuario.

1.6 Organización de la tesis

En el Capítulo 2 se expone el marco teórico necesario para familiarizar al lector con los temas que se abordarán en el presente trabajo. Se inicia con una introducción al PLN y las técnicas fundamentales que se suelen emplear para el análisis del texto. Posteriormente, se presenta un contexto general acerca de los distintos tipos de representaciones de texto que existen, donde se incluye su preprocesamiento y la vectorización usando métodos básicos y avanzados. En esta sección también se abordan técnicas de agrupamiento, clasificación y las métricas empleadas para su evaluación. Por último, se incluye una explicación general relacionada al desarrollo de aplicaciones y el diseño de interfaces gráficas.

El Capítulo 3 analiza las investigaciones realizadas por la comunidad científica acerca de la clasificación y el agrupamiento de documentos digitales. Se discuten las ideas planteadas en los estudios y se determina cuales podrían tener un aporte significativo en la experimentación y evaluación de esta tesis.

En el Capítulo 4 se explica la metodología implementada para los algoritmos de agrupamiento, donde se describen de manera general las etapas que se siguieron, así como la configuración experimental implementada, consistente en la selección de los algoritmos de agrupamiento, los valores utilizados en sus parámetros, las métricas de evaluación y los resultados obtenidos.

El Capítulo 5 describe un proceso similar al agrupamiento, pero enfocado en la clasificación. Se presenta la configuración experimental implementada, que incluye la selección de los algoritmos de clasificación, la optimización de parámetros, las métricas de evaluación y el análisis de los resultados obtenidos.

El Capítulo 6 desarrolla una explicación detallada de la GUI que integra los modelos de agrupamiento y clasificación. Primero se analizan los casos de uso que el usuario podrá realizar. Después, se describe el patrón de diseño que se siguió, así como la forma en que se implementó el flujo de pasos necesarios para la ejecución de los algoritmos. Al final, se muestran los resultados de la GUI, desde el diseño final de las páginas hasta su demostración de funcionamiento para el entrenamiento de modelos de agrupamiento y clasificación. En este capítulo también se presentan las conclusiones de este trabajo, así como el trabajo a futuro.

CAPÍTULO 2

Marco teórico

En este apartado, se abordan los diferentes temas teóricos necesarios para describir el proyecto de tesis. Estos temas se enfocan principalmente en la teoría relacionada al proceso general para tratar el texto mediante enfoques de PLN.

2.1 Procesamiento del lenguaje natural

Uno de los principales problemas del PLN nace de la siguiente pregunta: ¿cómo lograr que una computadora no solo perciba las palabras, sino que comprenda sus significados y sea capaz de asimilar el trasfondo, tal como lo haría una persona? (Chowdhary, 2020). Para abordar este dilema, se ha desarrollado una serie de procesos y técnicas que le permiten a una computadora llevar a cabo estas tareas. Comúnmente, el proceso empieza con el preprocesamiento de texto y continúa con la vectorización del texto para crear una representación numérica; posteriormente, se aplican algoritmos de aprendizaje automático adecuados para resolver la tarea de interés. Finalmente, se realiza una evaluación de resultados mediante métricas específicas (Jurafsky & Martin, 2026). En las siguientes secciones se explicará con más detalle en qué consisten estos pasos.

2.2 Representaciones de texto

Existen distintas formas de transformar datos textuales a numéricos, las cuales generalmente implican una etapa inicial de limpieza y preparación de los datos, seguida de técnicas que permiten convertir el texto a representaciones vectoriales. Estas representaciones per-

miten que los algoritmos puedan identificar patrones, similitudes y relaciones dentro de los documentos.

2.2.1 Preprocesamiento

Los autores Lamba y Madhusudhan (2022) señalan que, no todos los problemas de PLN requieren el mismo nivel de preprocesamiento; depende del tipo de aplicación y del algoritmo a usar. Algunas técnicas básicas son:

1. **Normalización.** Generalmente, el texto utiliza una mezcla de letras minúsculas y mayúsculas. Sin embargo, mantener una estandarización del texto a través de convertir todo a minúsculas o todo a mayúsculas permite que algunos algoritmos de aprendizaje automático generalicen mejor los patrones provenientes del texto (Lamba & Madhusudhan, 2022).

2. **Eliminación de ruido.** Como se mencionó antes, cada problema es diferente, y es cuestión de un analista determinar qué cosas no aportan nada a la resolución del problema. Se suele eliminar *URLs*, *hashtags*, números, caracteres especiales y *stop-words*. Las *stop-words* son *tokens* que no aportan información relevante y ocupan memoria, como, por ejemplo, conectores, preposiciones, artículos, adverbios, etc. Para la mayoría de los casos, remover las *stop-words* es de gran utilidad, siempre y cuando no tengan un impacto en los resultados finales o no represente pérdidas de información. Para este proyecto se implementa una lista de *stop-words* en español ya predefinida en la librería de *spacy*[1] integrada en Python.

3. **Normalización morfológica.** La normalización morfológica busca representar las palabras de una forma más general, es decir, reconocer las variantes que un término puede tener y aplicarlas en un solo elemento. Se aplican a través de técnicas como el *Stemming* o lematización. El *stemming* es el proceso de transformar palabras a su forma base, eliminando sufijos, prefijos o pluralización. En cambio, la lematización es una técnica avanzada del *stemming*, donde se mapea la forma verbal de un término a su forma infinitiva o los sustantivos plurales a su forma singular, por medio del entendimiento del contexto en una oración (Lamba & Madhusudhan, 2022). En este proyecto de tesis, estas técnicas fueron probadas y evaluadas en los Capítulos 4 y 5; sin embargo, no produjeron cambios significativos en los resultados de las métricas, por lo que se decidió no presentarlas en los experimentos.

[1] https://spacy.io/

El adecuado preprocesamiento del texto va a garantizar que la extracción de características o las representaciones vectoriales, tengan una reducción significativa del ruido, lo que cual puede influir directamente en la obtención de buenos resultados en la fase de aplicación de algoritmos de aprendizaje supervisado y no supervisado.

2.2.2 Vectorización

Un vector es, en esencia, un arreglo de números. En enfoques clásicos como *Bag-of-Words* o TF-IDF, la representación vectorial de documentos se define en un espacio de dimensión V, donde cada dimensión corresponde a un término único del corpus. Entonces, un documento queda representado como un vector $\mathbf{x} \in \mathbb{R}^V$, cuyos componentes o características pueden ser frecuencias, ponderaciones TF-IDF u otras representaciones vectoriales (Jurafsky & Martin, 2026). En años recientes se han desarrollado representaciones vectoriales más robustas, donde se manejan vectores contextuales que permiten un mayor manejo de la semántica del texto.

En este proyecto se experimentarán y analizarán tres técnicas de vectorización, incluyendo técnicas básicas y recientes. Esto permitirá comparar las ventajas y desventajas que cada técnica conlleva:

- TF–IDF
- Word2Vec
- BERT

A continuación se describen los fundamentos básicos de cada una de estas representaciones vectoriales.

2.2.2.1. TF–IDF

El esquema TF–IDF es en una de las primeras técnicas propuestas para la representación vectorial de textos. Algunos autores como Manning et al. (2008) la definen como una técnica estadística que mide qué tan relevante es una palabra dentro de un documento en relación con una colección de documentos. De acuerdo con Manning et al. (2008), TF-IDF se compone de dos partes principales:

- *TF (Term Frequency)*. Mide cuántas veces aparece el término t en el documento d:

$$\mathrm{TF}(t,d) = \frac{\text{Número de veces que aparece } t \text{ en } d}{\text{Total de tokens en } d} \tag{1}$$

- *IDF (Inverse Document Frequency)*. La frecuencia inversa de los documentos mide la importancia de un término t en el corpus de N documentos, donde df es el número de documentos que contienen t:

$$\mathrm{IDF}(t) = \log \frac{N}{1 + \mathrm{df}} \tag{2}$$

Entonces, la ponderación TF–IDF se define como:

$$\text{TF-IDF}(t,d) = \mathrm{TF}(t,d) \times \mathrm{IDF}(t) \tag{3}$$

TF-IDF se convirtió en una herramienta comúnmente usada para medir la relación entre palabras y documentos (Zhang et al., 2020).

2.2.2.2. Incrustaciones de palabras

En las representaciones clásicas, los vectores son dispersos (muchos ceros) y de alta dimensionalidad, como en el caso de TF-IDF. De ahí que han surgido nuevas representaciones por ejemplo, las basadas en *word embeddings*. Los *word embeddings* se consideran vectores densos, o sea, tienen una dimensión reducida (por ejemplo, 50, 100, 300 dimensiones). Su principal diferencia contra TF-IDF es su capacidad de capturar patrones semánticos, es decir, palabras que comparten significados parecidos, tienden a representarse cerca en un espacio vectorial (Jurafsky & Martin, 2026).

Los *word embeddigns* son entrenados mediante redes neuronales que aprenden representaciones vectoriales de millones de palabras. Entre los modelos más implementados se encuentra Word2Vec, propuesto en el artículo de Mikolov (2013). La idea principal es que para cada palabra w se genera un vector fijo $\mathbf{v}_w$, el cual no varía con el contexto (Jurafsky & Martin, 2026).

Por ejemplo:

- *El banco al lado del río se desgastó.*
- *Él fue al banco a sacar dinero.*

En ambos ejemplos el significado de *banco* es completamente distinto, para Word2Vec no existe diferencia, genera el mismo vector para la palabra *banco* sin distinción del contexto (Haris et al., 2026). Uno de los algoritmos que utiliza es *Skip-Gram*, una arquitectura basada en una red neuronal poco profunda que predice las palabras de contexto dada una palabra central (Mikolov, 2013). En la Figura 1 se muestra el diagrama que representa la funcionalidad de *Skip-Gram*.

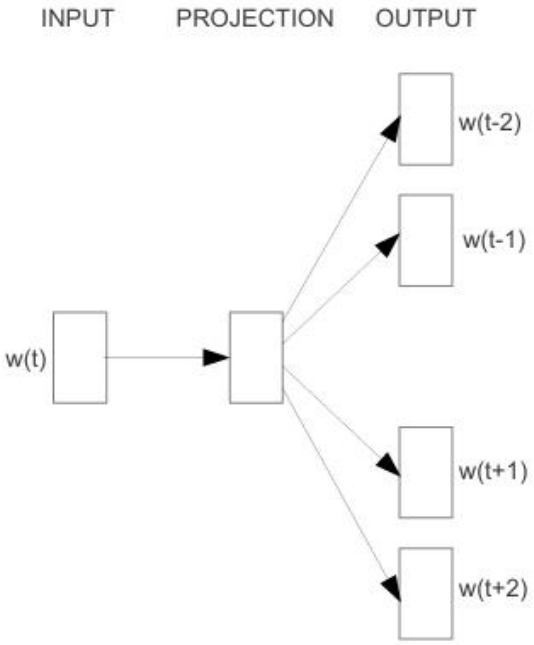

Figura 1. Arquitectura de Skip-Gram. Imagen tomada de Mikolov (2013)

El modelo ajusta sus parámetros y asigna a cada palabra un vector numérico, de manera que las palabras con un significado similar terminan teniendo vectores cercanos entre sí. Por ejemplo, después del entrenamiento, los vectores de *gato* y *perro* estarán próximos en el espacio multidimensional porque suelen aparecer en contextos similares.

2.2.2.3. Transformadores (BERT)

El transformador original, propuesto por Vaswani (2017), es una arquitectura basada en redes neuronales que se compone de dos partes esenciales: el codificador y el decodificador. El codificador se encarga de comprender el texto de entrada y genera representaciones continuas, mientras que el decodificador utiliza esas representaciones para generar el texto de respuesta (Vaswani, 2017).

Con el tiempo, la comunidad académica encontró que ambas partes podían utilizarse de manera independiente en la resolución de ciertas tareas, donde no es necesario generar una

respuesta textual; únicamente con utilizar el bloque del codificador o el decodificador resulta suficiente. Para este proyecto se utiliza la parte del codificador, específicamente el modelo *Bidirectional Encoder Representations from Transformers* (BERT) y basado en la arquitectura *Transformer*.

Los modelos como BERT utilizan una atención bidireccional, lo que significa que puede enfocarse en palabras anteriores y posteriores de la palabra que esté procesando y generar relaciones a partir de eso (Jurafsky & Martin, 2026). Esta característica le permite generar representaciones más completas del contexto. El funcionamiento general del *encoder* se puede expresar de forma simplificada como una secuencia de entrada transformada a una secuencia de salida contextualizada:

$$(x_1, x_2, \ldots, x_n) \rightarrow (h_1, h_2, \ldots, h_n) \tag{4}$$

donde cada h_i representa el token x_i pero contextualizado, usando toda la oración, es decir, cada palabra se representa considerando las demás palabras (Jurafsky & Martin, 2026).

Antes de ingresar al modelo, el texto debe transformarse en un formato entendible por BERT. Para ello, se emplean algoritmos de tokenización como *WordPiece*, los cuales dividen las palabras en *subtokens*. En la Figura 2 se muestra el proceso del *encoder*, el cual inicia después de convertir el texto a un formato entendible e identificar los *subtokens* resultantes con números.

Después de obtener los identificadores numéricos, estos se transforman en vectores densos mediante la capa inicial del modelo. BERT admite secuencias de 512 tokens como máximo, y cada token se representa como un vector de 768 posiciones. Por lo tanto, la entrada del modelo se representa así:

$$X \in \mathbb{R}^{N \times d} \tag{5}$$

donde N corresponde al número de tokens (con $N \leq 512$) y $d = 768$ es la dimensión del modelo.

Después de generar estos vectores, se le suman *positional embeddings*, que le permiten al modelo saber el orden de las palabras dentro de la secuencia. Posteriormente, la representación resultante pasa por múltiples capas de *self-attention*, el mecanismo que permite que cada token evalúe la relevancia de los demás tokens dentro de la misma oración y decida qué tan importante es cada una para su propio significado. Por ejemplo, en la frase "*El banco al lado del río se desgastó*", la palabra banco mirará a las palabras río y aprenderá que, en este contexto, probablemente se refiere a un lugar físico, no a una institución financiera. En

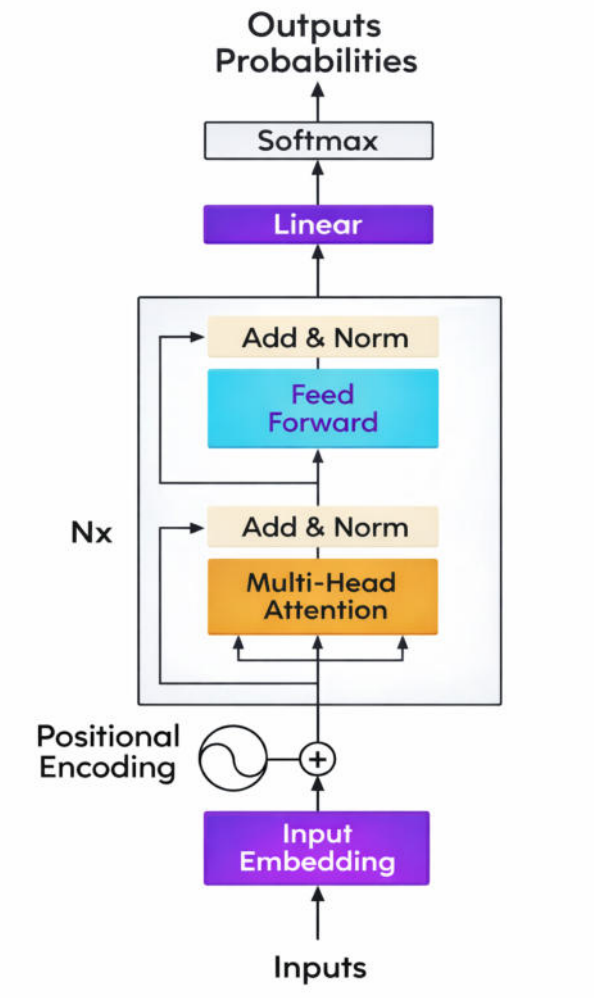

Figura 2. Encoder de la arquitectura transformador. Imagen tomada de Mary (2026)

otra frase, como "*Él fue al banco a sacar dinero*", la misma palabra observará sacar y dinero, y entenderá que ahora se trata de una entidad comercial (Haris et al., 2026).

Su capacidad de relacionar palabras entre sí es una característica que distingue a modelos como BERT, produciendo representaciones o *embeddings* que cambian conforme al corpus que se esté usando, a diferencia de Word2Vec, el cual asigna un vector fijo por palabra. Esto implica que cada capa añade información gramatical, sintáctica y semántica, lo que convierte a BERT en un modelo sumamente usado actualmente para el campo del PLN, en tareas como la extracción de información, clasificación, reconocimiento de entidades (NER), etc. (Darji et al., 2023; Jawahar et al., 2019).

2.3 Aprendizaje automático

El aprendizaje es uno de los procesos más fundamentales en los seres humanos, así como en los sistemas artificiales. El aprendizaje es una actividad importante que ayuda a los organismos vivos a adaptarse al ambiente. En el ámbito computacional, el aprendizaje se enfoca en crear teorías, procedimientos y algoritmos que permitan a las máquinas aprender por sí mismas (Chowdhary, 2020).

Algunos autores como Russell y Norvig (2004) refieren que el tipo de información disponible para el aprendizaje suele ser el factor más importante para determinar la naturaleza del problema de aprendizaje. Existen múltiples tipos de aprendizaje automático, pero en esta tesis únicamente se explorará el aprendizaje supervisado y no supervisado.

2.3.1 Aprendizaje no supervisado

El aprendizaje no supervisado a menudo se realiza como parte de un análisis exploratorio de datos. A diferencia del aprendizaje supervisado, no se devuelve un valor etiquetado al ingresar un valor de entrada. Por lo tanto, no existe forma de evaluar las predicciones del modelo, dado que se desconoce la verdadera respuesta. Debido a esta característica, es difícil evaluar el desempeño de los algoritmos de aprendizaje no supervisado, ya que no existe un mecanismo universalmente aceptado para validar resultados en un conjunto de datos independientes (James et al., 2023).

Una de sus técnicas más conocidas dentro de este enfoque es el agrupamiento, en el cual los datos son divididos en grupos de acuerdo con alguna métrica de disimilaridad o características compartidas. Los autores Cormen et al. (2022) explican que los puntos dentro de un mismo grupo son aquellos que comparten el mismo centroide más cercano, mientras que puntos en distintos grupos tienen centroides más cercanos diferentes.

2.3.1.1. K-Medias

El algoritmo de K-Medias es un método de agrupamiento donde, a partir de un conjunto de datos, se forman k grupos o *clusters*. Los datos se agrupan de forma en que los puntos en el mismo grupo sean más similares entre sí que los puntos pertenecientes a otros *clusters* (Chowdhary, 2020).

Cada grupo está representado por su centroide; este se refiere a la media aritmética de

los puntos de datos asignados al grupo, es decir, es el promedio de todos los objetos del conjunto. De esta manera, el algoritmo funciona a través de un proceso iterativo hasta que cada punto de datos esté más cerca del centroide de su propio grupo que de los centroides de otros grupos, minimizando la distancia dentro del grupo en cada paso (Chowdhary, 2020).

Desde una perspectiva matemática, Cormen et al. (2022) definen el algoritmo de *K-Means* de la siguiente manera: Dado un conjunto S de n puntos y un entero positivo k, tenemos una secuencia $C = \langle c^{(1)}, c^{(2)}, \ldots, c^{(k)} \rangle$ de k puntos centro que minimicen la suma $f(S, C)$ de la distancia al cuadrado desde cada punto a su centro más cercano, donde

$$f(S, C) = \sum_{x \in S} \min\{\delta(x, c^{(j)}) \mid 1 \leq j \leq k\} = \sum_{j=1}^{k} \sum_{x \in S^{(j)}} \delta(x, c^{(j)}). \tag{6}$$

es decir,

$$f(S, C) = \sum_{j=1}^{k} \sum_{x \in S^{(j)}} \|x - c^{(j)}\|^2. \tag{7}$$

Desglosando cada parte de la función, encontramos que:

- $S^{(j)}$ representa el conjunto de puntos que están asignados al *cluster* j, cuyo centroide es $c^{(j)}$.
- $\sum_{j=1}^{k}$: es la suma que recorre todos los k *clusters*.
- $\sum_{x \in S^{(j)}}$: dentro de cada *cluster* j, se suman las distancias cuadradas entre cada punto x y su centroide correspondiente $c^{(j)}$.
- $\|x - c^{(j)}\|^2$: esta es la distancia euclidiana al cuadrado entre el punto x y el centroide $c^{(j)}$.

Las expresiones matemáticas sobre este algoritmo fueron tomadas de (Cormen et al., 2022).

Para explicar de una forma más gráfica al modelo *K-Means*, la Figura 3 representa cómo, en los datos de entrada no etiquetados (a), se colocan dos centroides aleatoriamente (b); entonces, los puntos de datos serán marcados midiendo sus distancias a cada centroide, ya sea que se encuentren más cerca del centroide 1 o del 2 (c), y así sucesivamente se irá ajustando el modelo hasta que los movimientos de los puntos sean inexistentes.

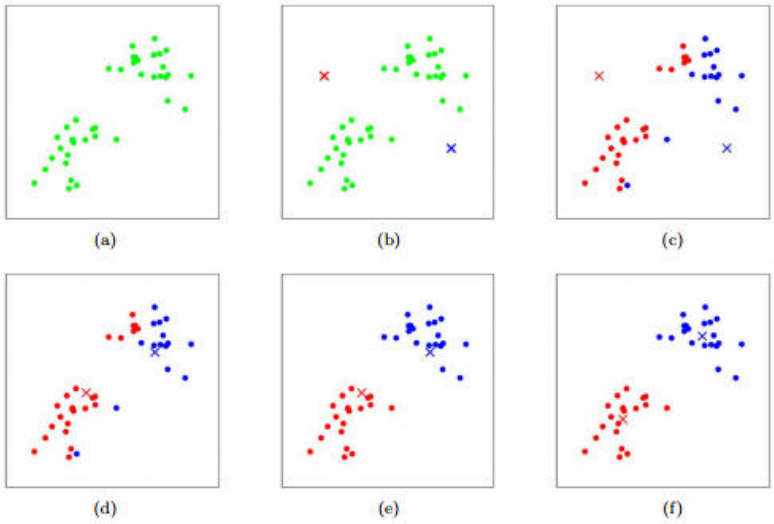

Figura 3. Representación gráfica del algoritmo K-Medias. Imagen tomada de Chowdhary (2020)

La ventaja principal de K-Medias radica en que es fácil de implementar. Sin embargo, su desventaja es su sensibilidad ante elegir la partición inicial; si no se elige adecuadamente, puede encontrar una solución aceptable, pero no necesariamente la mejor (Chowdhary, 2020).

2.3.1.2. HDBSCAN

El algoritmo *Hierarchical Density-Based Spatial Clustering of Applications with Noise* (HDBSCAN) propuesto por los autores Campello et al. (2013, 2015), es un algoritmo de agrupamiento no supervisado que extiende y mejora el algoritmo DBSCAN.

El objetivo de HDBSCAN es descubrir agrupamientos (*clusters*) de datos con densidad variable, es decir, grupos de puntos más densos separados por regiones menos densas, como se muestra en la Figura 4. También es capaz de detectar ruido o puntos atípicos (*outliers*) (Campello et al., 2015).

Entre sus ventajas se encuentra que HDBSCAN es capaz de identificar grupos con densidades variables, esto le permite tener una mayor robustez ante la selección de parámetros (McInnes et al., 2017). En el trabajo de Campello et al. (2015) se define el procedimiento general del algoritmo, descrito de la siguiente manera:

1. Se calcula la distancia núcleo (*core distance*) de cada punto, en función del parámetro $mpts$ (número mínimo de vecinos).

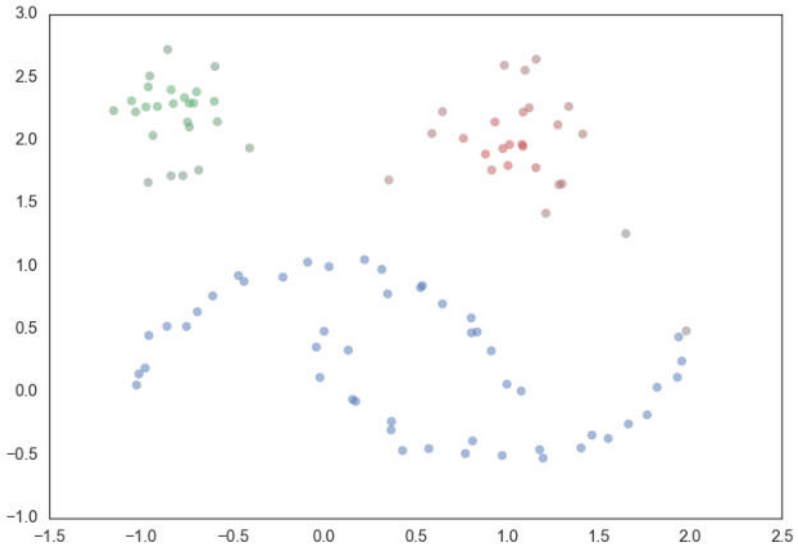

Figura 4. Ejemplo de agrupamiento con el algoritmo HDBSCAN. Imagen tomada de McInnes et al. (2016)

2. Se construye un grafo utilizando estas distancias y se obtiene su *Minimum Spanning Tree* (MST), el cual conecta todos los puntos del conjunto de datos minimizando la suma total de las distancias..

3. Se extiende el MST agregando auto-conexiones (*self-loops*) a cada punto, con un peso igual a su distancia núcleo. Esto permite identificar cuándo un punto deja de pertenecer a un grupo y pasa a ser considerado ruido.

4. Se eliminan las aristas del árbol en orden descendente de peso:

    - Al eliminar aristas, el árbol se divide en componentes conexas.
    - Cada componente representa un grupo en un determinado nivel de densidad.
    - Si un componente queda aislado sin conexiones, se considera ruido.

2.3.1.3. Evaluación de los algoritmos de agrupamiento

En problemas de aprendizaje no supervisado, no se dispone de etiquetas reales que permitan comparar las predicciones con su clase verdadera. La evaluación debe realizarse utilizando únicamente la información contenida en los propios datos. Estas métricas proporcionan una visión de las propiedades geométricas de los agrupamientos descubiertos (como su separación y compactación), permitiendo cuantificar la calidad de los grupos generados de acuerdo con una noción intuitiva de cómo debería ser un buen grupo (Chicco et al., 2025).

Entre las métricas internas más utilizadas en la literatura se encuentran el Coeficiente de Silueta (*Silhouette Score*), el índice de Calinski-Harabasz y el índice de Davies-Bouldin.

Coeficiente de Silueta. El coeficiente de silueta es una de las métricas más populares para medir la consistencia de un agrupamiento. Refleja qué tan bien asignado está cada punto a su grupo; para cada punto calcula la distancia entre los demás puntos de su propio cluster (intra-cluster); luego calcula que tan lejos se encuentra un punto respecto a los puntos de otro cluster más cercano. Sus valores oscilan entre -1 y +1, donde valores cercanos a +1 indican una buena asignación, es decir, si un punto está bien ubicado en su cluster o si estaría mejor en otro (Chicco et al., 2025).

Para una función de distancia d y un conjunto de datos particionado en k grupos $\{C_t : t = 1, \ldots, k\}$, dado un punto de datos i perteneciente a un grupo no unitario C_I, se define:

$$a(i) = \frac{1}{|C_I| - 1} \sum_{\substack{j \in C_I \\ j \neq i}} d(i, j) \tag{8}$$

como la medida de disimilitud intra-grupo, y

$$b(i) = \min_{J \neq I} \left(\frac{1}{|C_J|} \sum_{j \in C_J} d(i, j) \right) \tag{9}$$

como la medida de disimilitud inter-grupo. A partir de estas, el coeficiente de Silhouette se define como:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{10}$$

Para el caso de un grupo unitario $C_I = \{i\}$, se establece $s(i) = 0$.

El coeficiente de Silhouette toma valores en el rango $[-1, 1]$, donde valores cercanos a $+1$ indican una buena asignación al grupo, valores cercanos a 0 indican solapamiento entre grupos, y valores cercanos a -1 indican una posible asignación incorrecta.

Índice Calinski–Harabasz. También llamado criterio de razón de varianza (Chicco et al., 2025), mide la calidad del agrupamiento comparando qué tan distintos y alejados están los grupos entre sí y qué tan compactos son los puntos dentro de cada grupo. La fórmula del

índice Calinski–Harabasz, se describe a continuación:

$$CH = \frac{\text{BCSS}/(k-1)}{\text{WCSS}/(n-k)} \tag{11}$$

En donde,

- k: número de grupos.
- n: número total de observaciones.
- $BCSS$ (*Between-Cluster Sum of Squares*): suma de cuadrados entre grupos. Representa la variabilidad explicada por la separación entre los diferentes grupos. Cuanto mayor sea, más alejados estarán los grupos entre sí.
- $WCSS$ (*Within-Cluster Sum of Squares*): suma de cuadrados dentro de los grupos. Mide la compacidad interna de los grupos; valores pequeños indican que los elementos dentro de cada grupo están más cerca entre sí.

El numerador $\text{BCSS}/(k-1)$ mide la dispersión promedio entre grupos, ajustada por el número de grupos. El denominador $\text{WCSS}/(n-k)$ mide la dispersión promedio dentro de los grupos, ajustada por los grados de libertad. Valores altos indican agrupaciones densas y bien separadas.

Índice Davies–Bouldin. Mide el coeficiente entre la dispersión intra-grupo y la separación inter-cluster (Chicco et al., 2025). Es decir, si los grupos son compactos (baja dispersión) y estás bien separados (alta distancia), el cociente será pequeño, lo que indica que existe una buena calidad en el agrupamiento. Esto se puede interpretar como valores cercanos a 0 indican alta calidad, y valores grandes indican baja calidad. Esta métrica no tiene un rango limitado de valores (Chicco et al., 2025).

$$DBI = \frac{1}{N}\sum_{i=1}^{N} \max_{j \neq i} R_{ij} \quad \text{donde} \quad R_{ij} = \frac{s_i + s_j}{d_{ij}} \tag{12}$$

Dónde:

- N = número de clusters
- s_i = la distancia media entre cada muestra en el cluster i y el centroide del cluster i
- d_{ij} = la distancia entre los centroides de los clusters i y j

El índice Davies-Bouldin, a diferencia del coeficiente de silueta, cuanto más bajo sea su valor, más particionadas estarán las muestras, siendo cero la puntuación mínima.

En la Tabla 1 se muestra un resumen de los rangos en que se debe encontrar cada métrica para considerar si la división de grupos fue exitosa.

Tabla 1. Rangos óptimos en las métricas de evaluación para agrupamiento

Métrica	**Rango teórico**	**Valor preferible**
Silhouette Score	$[-1, 1]$	Cercano a 1
Índice de *Calinski-Harabasz*	$[0, +\infty)$	Valores altos
Índice de Davies-Bouldin	$[0, +\infty)$	Valores bajos

Nota. Tabla adaptada de Chicco et al. (2025).

En resumen, aunque las tres métricas comparten el objetivo general de medir la calidad de los agrupamientos generados, lo hacen mediante ecuaciones matemáticas distintas y tienen sensibilidad a diferentes características como la forma, densidad y distribución de los datos. El coeficiente de silueta evalúa la cohesión entre los puntos pertenecientes a un mismo cluster (intra-cluster), y la separación entre los clusters resultantes (inter-grupo) punto a punto; el índice de *Calinski–Harabasz* introduce una relación entre la distancia global que existe entre los centroides y la distancia entre grupos, cercano al análisis de varianza; mientras que el índice de Davies–Bouldin enfatiza la relación entre la compactación y proximidad entre centroides, penalizando especialmente el solapamiento (Miller et al., 2024).

2.3.2 Aprendizaje Supervisado

El aprendizaje supervisado se refiere a construir un modelo estadístico con base a conjuntos de datos ya etiquetados para la predicción o estimación de un valor de salida, es decir, a partir de alimentar una función con ejemplos de entrada se infieren resultados concretos, los cuales también son valores etiquetados (James et al., 2023).

A continuación, se describirán los algoritmos de aprendizaje supervisado utilizados en este trabajo.

2.3.2.1. Bosques aleatorios

Formalmente, el autor Breiman (2001) define un bosque aleatorio como una colección de clasificadores con estructura de árbol $\{h(\mathbf{x}, \Theta_k),\ k = 1, 2, \ldots\}$, donde $\{\Theta_k\}$ represen-

tan vectores aleatorios independientes e idénticamente distribuidos. El vector Θ_k controla la construcción del k-ésimo árbol, o sea, que subconjunto de datos se considera en cada nodo. Al final, cada árbol emite un voto unitario para la clase más popular dado el vector de entrada $\mathbf{x}$. En la Figura 5 se muestra gráficamente el funcionamiento general de este modelo.

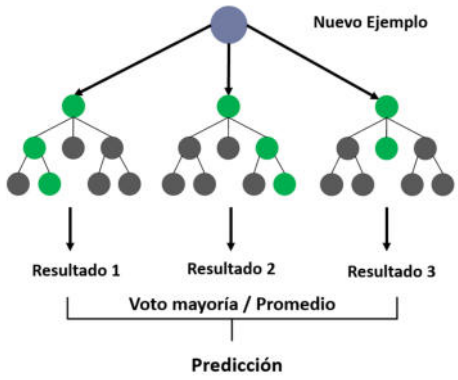

Figura 5. Construcción de bosques aleatorios. Imagen tomada de Yehoshua (2023)

En este sentido, el autor resume el procedimiento para la construcción de bosques aleatorios, denominado Forest-RI (*Random Input*), consistente en lo siguiente:

1. Para cada árbol $k = 1, 2, \ldots, K$:

   a) Obtener un conjunto de entrenamiento T_k, lo que refiere a un muestreo del conjunto original.

   b) Crecer un árbol de decisión sin poda. La regla de división en cada nodo es: seleccionar al azar F variables de las M disponibles y elegir la mejor división únicamente entre esas F variables, donde F representa el número de variables candidatas y M el número total de variables de entrada. Al restringir la búsqueda a solo F variables, los árboles resultan distintos entre sí, lo que reduce su correlación.

2. Una vez construidos los K árboles, para clasificar una nueva observación $\mathbf{x}$, cada árbol $h(\mathbf{x}, \Theta_k)$ emite solo un voto a favor de la clase que predice. La predicción que tenga un mayor número de votos entre los K árboles será la elegida.

A diferencia de un árbol de decisión individual, un bosque de cientos de arboles es difícilmente interpretable, lo que podría representar una desventaja para la implementación del algoritmo. Aún asi, los bosques aleatorios se consideran modelos altamente eficientes gracias a su adaptabilidad ante el ruido de los datos, no presenta sobreajuste y suele ser rápido computacionalmente, etc., (Breiman, 2001).

2.3.2.2. Máquinas de vectores de soporte

Las Máquinas de Vectores de Soporte constituyen un modelo de aprendizaje supervisado, cuyo objetivo es resolver problemas de clasificación binaria mediante la construcción de un hiperplano óptimo que separa los datos en el espacio de características (Cortes & Vapnik, 1995).

La Figura 6 ilustra el concepto fundamental del algoritmo SVM en un problema de clasificación binaria dentro de un espacio bidimensional (X1, X2). El hiperplano de máximo margen representa la frontera de decisión óptima, ya que maximiza la distancia entre las dos clases (Cortes & Vapnik, 1995).

A cada lado del hiperplano, se encuentran dos hiperplanos paralelos, uno corresponde a la clase positiva y otro a la negativa. Los puntos que se encuentran sobre estos hiperplanos se denominan vectores de soporte, estos son los elementos fundamentales del conjunto de entrenamiento, pues si el resto de los puntos fuera removido, la solución del clasificador no cambiaría (Burges, 1998).

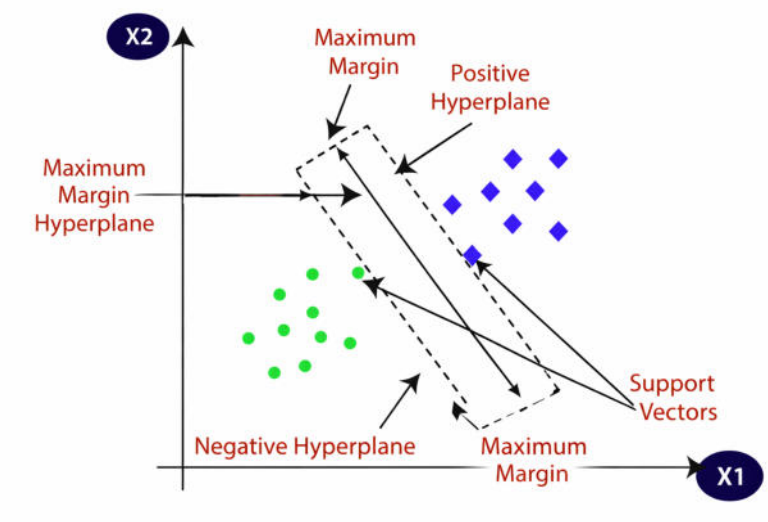

Figura 6. Ejemplo de SVM en un problema de clasificación binaria. Imagen tomada de Kamble (2024)

El hiperplano de separación puede representarse mediante una función lineal en el espacio de características, la cual se expresa como $y(x) = w^T\phi(x) + b$, donde $\phi(x)$ es una transformación del dato de entrada al espacio de características, w es un vector de pesos que define la orientación del hiperplano y b es un término de sesgo que determina su posición. De esta forma, el hiperplano corresponde al conjunto de puntos que satisfacen $y(x) = 0$, mientras que el signo de $y(x)$ indica a qué lado del hiperplano pertenece cada instancia (Bishop, 2006).

Este algoritmo destaca principalmente porque maneja eficazmente espacios de alta dimensionalidad (como el texto), al asumir que la mayoría de las características son irrelevantes. Además, muchos problemas de clasificación de texto son linealmente separables. Estás razones convierten a las SVM en una herramienta ampliamente usada en el campo del PLN y la clasificación textual (Joachims, 1998).

2.3.2.3. Redes neuronales artificiales

Las redes neuronales artificiales (RNA) son modelos matemáticos inspirados en el funcionamiento del cerebro humano, capaces de aprender y adaptarse a partir de los datos. En términos formales, se describen como un grafo dirigido compuesto por nodos (neuronas) y conexiones (aristas), cuyos pesos sinápticos determinan la influencia de una neurona sobre otra (Haykin, 2009). En la Figura 7 se muestra un perceptrón simple, el modelo matemático más simple de una neurona artificial, la cual se compone de los siguientes elementos fundamentales de acuerdo a Haykin (2009):

- Enlaces de conexión: donde $x_1, x_2, \ldots, x_m$ son las señales de entrada; $w_1, w_2, \ldots, w_m$ son los pesos sinápticos correspondientes a la neurona k; la entrada es multiplicada por el peso sináptico generando así, una conexión entre las entradas y la neurona.
- Sesgo (*bias*), ajusta la entra de la función de activación, que tiene el efecto de aumentar o disminuir dependiendo si el valor es positivo o negativo.
- Suma ponderada: expresada como v, donde se suma el resultado de multiplicar las entradas x_m por la intensidad sináptica de la neurona, el peso w_m; esto constituye a un combinador lineal.
- Función de activación: definida como $\varphi(\cdot)$, transforma la salida de la neurona, permitiendo modelar relaciones no lineales en un valor finito. Típicamente, el rango de salida de una neurona se expresa en un intervalo cerrado unitario $[0, 1]$ o, alternativamente, $[-1, 1]$.
- Salida: la predicción final de la neurona definida como y.

Posteriormente, surgió el perceptrón multicapa. Un modelo de red neuronal compuesto por múltiples capas de neuronas interconectadas. Cada conexión entre una entrada y una neurona posee un peso que se ajusta durante el entrenamiento para minimizar el error del modelo. En el interior de cada neurona, se calcula la suma ponderada de las entradas y posteriormente se aplica una función de activación (Goodfellow et al., 2016), lo que permite

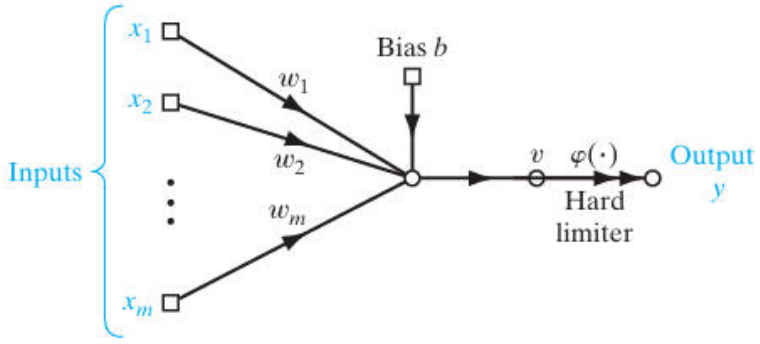

Figura 7. Modelo de un perceptrón simple. Imagen tomada de Haykin (2009).

que la red capture relaciones no lineales presentes en los datos, especialmente útiles en el procesamiento del lenguaje natural.

El resultado final de todas las neuronas se agrupan en un vector numérico que representa la salida de la red. Este resultado se compara con las etiquetas reales mediante una función de error, comúnmente la entropía cruzada, que mide la discrepancia entre la predicción del modelo y los valores esperados. A partir de esta diferencia, se aplica el proceso de retropropagación del error (*backpropagation*), que utiliza el método del descenso de gradiente para ajustar los pesos en dirección contraria al error (Goodfellow et al., 2016) .

2.3.2.4. Evaluación de los algoritmos de clasificación

Las métricas de evaluación permiten conocer el desempeño de los modelos de aprendizaje supervisado. Según Miller et al. (2024), escoger un tipo de métrica, depende de la distribución de los datos y el objeto de estudio. Este mismo autor, hace un resumen de las métricas más usadas para clasificadores: Exactitud, Precisión, *Recall*, *F1-Score*.

Matriz de confusión. La matriz de confusión es una tabla que permite visualizar el rendimiento de un clasificador respecto a datos de prueba. En la Figura 8 se presenta los componentes de la matriz de confusión. Cada fila representa las instancias de la clase real, mientras que cada columna corresponde a las predicciones. La diagonal principal refleja las clasificaciones correctas, mientras que fuera de ella se identifican los errores (falsos positivos, falsos negativos, etc.) (Ting, 2024).

Los componentes son:

- VP – Verdaderos Positivos: instancias positivas correctamente clasificadas. Ejemplo: detectar enfermos que realmente lo están.

Figura 8. Matriz de confusión. Imagen tomada de Markoulidakis et al. (2021)

- VN – Verdaderos Negativos: instancias negativas correctamente clasificadas. Ejemplo: personas sanas identificadas como sanas.
- FP – Falsos Positivos: negativas clasificadas erróneamente como positivas. Ejemplo: diagnosticar sano como enfermo.
- FN – Falsos Negativos: positivas clasificadas erróneamente como negativas. Ejemplo: no detectar una enfermedad presente.

A continuación, se definen las fórmulas utilizadas para calcular las métricas de evaluación derivadas de la matriz de confusión, de acuerdo con el autor Markoulidakis et al. (2021).

Exactitud. Es la proporción de predicciones correctas respecto al total, ya sean positivas o negativas:

$$\text{Exactitud} = \frac{\text{VP} + \text{VN}}{\text{VP} + \text{VN} + \text{FP} + \text{FN}} \tag{13}$$

Es útil en conjuntos balanceados, pero puede resultar engañosa si existe desbalance en las clases.

Precisión. Mide la proporción de predicciones positivas que realmente son positivas:

$$\text{Precision} = \frac{\text{VP}}{\text{VP} + \text{FP}} \tag{14}$$

Es relevante cuando los falsos positivos disminuyen.

Sensibilidad (*recall*). Indica la fracción de positivos reales correctamente identificados:

$$\text{Sensibilidad} = \frac{\text{VP}}{\text{VP} + \text{FN}} \tag{15}$$

En un conjunto de datos desequilibrados, esta métrica es más significativa que la exactitud, ya que mide la capacidad del modelo para identificar correctamente todas las instancias positivas.

F1-score

El *F1-score* es la media armónica entre precisión y *recall*:

$$F1 = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{16}$$

Comúnmente, se emplea cuando existe un desbalance de clases y se busca medir un equilibrio entre precisión y *recall*.

2.4 Desarrollo de aplicaciones

El desarrollo de aplicaciones computacionales requiere una serie de elementos a considerar, no se trata únicamente de la lógica de programación, sino también de la experiencia del usuario con el sistema y su usabilidad. Las aplicaciones modernas se enfocan en implementar algoritmos complejos, manteniendo al mismo tiempo, una interfaz intuitiva que facilite la interacción con el usuario (Galitz, 2007). Para la etapa final del proyecto, se desarrolló una Interfaz Gráfica de Usuario (GUI, por sus siglas en inglés) que permitiera desplegar los modelos explicados en puntos anteriores.

2.4.1 Interfaces Gráficas de Usuario

Una GUI es una colección de técnicas y mecanismos para facilitar la interacción humano-computadora a través de componentes audiovisuales (Chicala et al., 2021). Estas interacciones ocurren mediante dispositivos (como el ratón o *mouse*) que permiten manipular elementos referidos como objetos. Los objetos son siempre visibles y sirven para realizar tareas, son percibidos como entidades independientes (Galitz, 2007) y pueden representar archivos, botones, ventar o cualquier otro objeto con el que un usuario puede interactuar.

De acuerdo con el autor Galitz (2007), en el diseño de interfaces gráficas destacan características como:

- Presentación visual: la GUI se basa en una representación visual que permite mostrar elementos como ventanas, menús, iconos, botones y cuadros de texto, así como gráficos, tipografías y colores. El objetivo es presentar la información de manera clara y sencilla, permitiendo que el usuario identifique los elementos necesarios para llevar a cabo una tarea.
- Interacción mediante apuntar y hacer clic: el usuario identifica un elemento y ejecuta la

acción de seleccionar y hacer clic mediante el curso. Esto permite tener una interacción visual en relación con la acción manual.

- Visualización: esta característica le permite al usuario entender la información que es difícil de percibir, ya sea por ser muy abstracta o porque es mucha información. Requiere representar gradualmente la estructura o funciones del sistema.

En una GUI, los componentes a considerar en una ventana incluyen: título, controles de pantalla (botones, menús, campos), encabezados, contenido principal y mensajes de ayuda o instrucciones. El objetivo principal del diseño de interfaces gráficas es reducir el esfuerzo del usuario. Para esto es importante disminuir la carga cognitiva mediante la eliminación del ruido o información innecesaria, la adecuada organización de los elementos y crear mecanismos de reconocimiento en lugar de fomentar la memorización. También es importante reducir el número de acciones y movimientos necesarios para complementar una tarea y automatizar procesos repetitivos (Galitz, 2007)

2.4.2 Patrones para el desarrollo de interfaces gráficas de usuario

Para el desarrollo de la GUI en este proyecto, no se siguió una metodología compleja de diseño, ya que la interfaz únicamente funge como un prototipo funcional cuyo objetivo es integrar modelos computacionales. Por ello se optó por usar un patrón de diseño conocido como Modelo-Vista-Controlador (MVC), el cual establece una separación entre los objetos del dominio, los cuales moldean la idea del mundo real, y los objetos de presentación, que muestran los elementos de la interfaz gráfica en pantalla (Fowler, 2003).

El autor Fowler (2003) explica que dentro del patrón MVC, el objeto de dominio se conoce como Modelo, que es completamente independiente a la interfaz de usuario. La capa de presentación se compone por la Vista y el Controlador. La vista se encarga de mostrar el estado del modelo al usuario y todos los elementos visuales para llevar a cabo las tareas; el controlador es el que permite la comunicación entre la vista y el modelo, es decir, maneja todas las entradas de usuario y las traduce en acciones para el modelo. Cada elemento de la interfaz puede tener su propio par vista-controlador; cuando un usuario interactúa con la interfaz, los controladores determinan el componente que recibió la acción, se ejecutan la lógica interna para posteriormente, coordinar la actualización de la GUI.

En la Figura 9 se observa un diagrama propuesto por Fowler (2003), en el cual enfatiza las dependencias esenciales entre el modelo, la vista y el controlador. También resalta la importancia de conectar la vista y el controlador dado que se vinculan directamente entre sí,

pero los desarrolladores generalmente no utilizan este hecho.

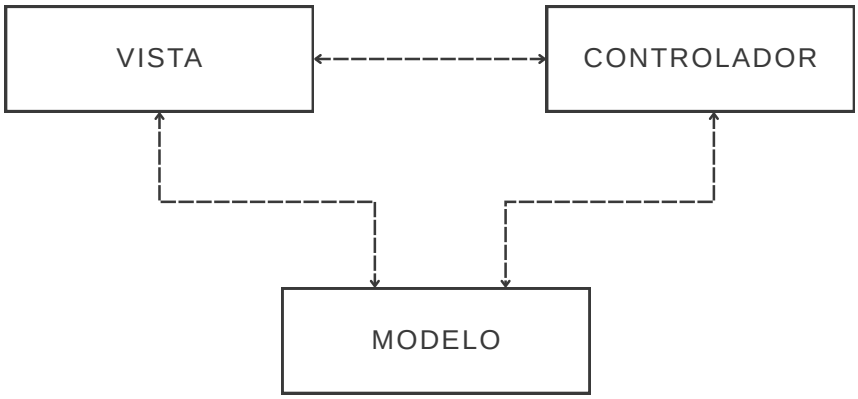

Figura 9. Diagrama de interacción del patrón de diseño Modelo-Vista-Controlador. Imagen obtenida de Fowler (2003)

CAPÍTULO 3

Estado del arte

A través de los años, diversos autores han explorado una serie de métodos y técnicas dentro del aprendizaje automático y el procesamiento del lenguaje natural enfocados esencialmente a la organización automática de documentos digitales. Estos métodos se basan tanto en agrupar documentos sin categorías previas como en clasificarlos dentro de etiquetas ya definidas. Recientemente, dichos métodos se han integrado en sistemas computacionales que cumplen con la tarea planteada.

Dado el contexto descrito, este capítulo se estructura en los siguientes enfoques: soluciones aplicadas, métodos basados en agrupamiento, métodos basados en clasificación. De igual manera, se describe la colección de datos utilizada para los Capítulos 4 y 5. Esta organización permite analizar las fortalezas y limitaciones de los algoritmos empleados en los artículos revisados, y sustentar las decisiones que se tomaron en el presente proyecto.

3.1 Aplicaciones para clasificación de documentos

Existen diversas herramientas en línea que permiten automatizar el procesamiento de documentos digitales, de las cuales destaca *Document AI*[1]. De acuerdo con la documentación oficial de *Document AI* se trata de un servicio en la nube que permite extraer, clasificar o dividir documentos digitales mediante técnicas de aprendizaje supervisado.

En cuanto al primer apartado, la plataforma es capaz de extraer información estructurada a partir de documentos no estructurados o semiestructurados, como facturas, formularios, contratos e identificaciones, mediante modelos entrenados. También cuenta con la opción de

[1] https://cloud.google.com/document-ai

reentrenar modelos para clasificar documentos a partir de etiquetas definidas por el usuario, el cual funciona a través del ajuste fino (*fine-tuning*) de dichos modelos con los datos proporcionados. Asimismo, permite el entrenamiento de modelos personalizados desde cero, empleando conjuntos de datos propios. La plataforma también incluye un apartado para dividir documentos en conjuntos de temas, es decir, identifica secciones de documentos que tienen diferentes tipos de datos estructurados como solicitudes, comprobantes de ingresos e identificaciones oficiales.

Sin embargo, el uso de este servicio implica un costo que se incrementa de acuerdo con la cantidad de documentos procesados, lo que podría resultar inviable para un usuario individual que solo desea organizar sus documentos personales. Además, sus datos personales tendrían que ser expuestos a servidores externos.

3.2 Enfoques basados en agrupamiento de textos

Los algoritmos de agrupamiento por definición no requieren de datos previamente etiquetados, esto produce que las agrupaciones formadas, estén basadas en la estructura interna de los datos y a la proximidad de sus características. Existen diferentes tipos de métodos de agrupamiento como, por ejemplo, los métodos en centroides, como K-Means; los basados en densidad como DBSCAN y HDBCAN; los distribuidos y jerárquicos. La literatura ha mostrado que el desempeño de estos métodos no depende solo del algoritmo, sino también de la forma en que el texto es representado numéricamente (Subakti et al., 2022).

Por ejemplo, el autor Saha (2023) analiza la influencia de diferentes representaciones vectoriales en el desempeño de los algoritmos de agrupamiento para reseñas de productos, implementado los algoritmos: K-Means (basado en centroides), *agglomerative single-linkage* (jerárquicos), DBSCAN y HDBSCAN (basados en densidad). El autor evalúa los resultados implementando una métrica interna (coeficiente de silueta) y algunas externas (*Adjusted Rand Index (ARI)* y Pureza). También examina el uso de la distancia euclidiana y una variación de hiperparámetros, como es el número de clústeres para K-Medias, *agglomerative* para DBSCAN y el tamaño mínimo del clúster para HDBSCAN.

El autor concluye que, los resultados obtenidos varían tanto por el tipo de representación textual como por la familia del algoritmo utilizado ya que no existe una combinación universalmente óptima. Aunque K-Medias mostró valores aceptables de acuerdo con las estimaciones de las métricas internas, su descubrimiento de las clases reales fue limitada. Por otro lado, los métodos de densidad lograron identificar estructuras más finas dentro de los datos, pero

a costa de clasificar muchos documentos como ruido. Esto implica que, aunque los grupos obtenidos pueden ser más coherentes semánticamente, una parte del corpus queda fuera del análisis.

Otra conclusión importante de Saha (2023) es que, los resultados estimados con las métricas internas y externas pueden diferir; por ello, es conveniente analizarlas en conjunto y considerar también el número de valores atípicos al interpretar los resultados. El autor señala, además, las limitaciones y líneas de trabajo. Por ejemplo, extender la diversidad del conjunto de datos, ampliar la selección de hiperparámetros y explorar el ajuste de modelos para alinear mejor la geometría del espacio con la tarea de agrupamiento.

En años recientes se ha propuesto un flujo de trabajo para el modelado de tópicos que aprovecha la semántica a través de transformadores, como BERT, combinado con técnicas de agrupamiento (generalmente HDBSCAN) y una variante de TF-IDF adaptada a clústeres (c-TF-IDF), que permite encontrar las palabras más representativas de cada grupo (Maarten, 2022). Este modelo es conocido como BERTopic y genera representaciones de tópicos a partir de tres pasos: primero, la incrustación de documentos, donde cada documento se transforma en un vector numérico mediante un modelo preentrenado (BERT), lo que permite identificar similitudes semánticas; posteriormente, el agrupamiento de documentos, en el cual se emplea UMAP para reducir la dimensionalidad de los *embeddings* y, sobre estos, HDBSCAN para identificar grupos densos y manejar ruido y valores atípicos; finalmente, la representación de tópicos, donde se aplica c-TF-IDF considerando cada grupo como una categoría, lo que permite obtener un conjunto de palabras representativas que funcionan como etiquetas de tópicos interpretables por humanos.

Se realizaron experimentos para evaluar el desempeño de BERTopic mediante dos métricas que miden la coherencia del tema y la diversidad del tema. Los resultados muestran que BERTopic produce tópicos más coherentes e interpretables que otros algoritmos más clásicos como la *Asignación Latente de Dirichlet* (LDA, por sus siglas en inglés). No obstante, se muestra un cambio en los resultados dependiendo del algoritmo de agrupamiento empleado. Cuando HDBSCAN clasifica muchos documentos como ruido, la generación de tópicos puede verse afectada. Aunque el artículo no enfatiza el costo computacional como una limitación principal, es conocido que el uso de modelos profundos conlleva mayores requerimientos de memoria y tiempo de procesamiento en comparación con otros enfoques.

Más adelante, Groot et al. (2022) amplió el análisis anterior al evaluar el desempeño de BERTopic en textos cortos y provenientes de múltiples dominios. Esta evaluación fue relevante, ya que muchos conjuntos de datos reales no están compuestos por documentos largos

y homogéneos, como podría ser el caso de usuarios comunes que desean organizar sus documentos personales.

En el apartado experimental, Groot et al. (2022) hace una comparación del desempeño de HDBSCAN y K-Medias. Los resultados muestran que, aunque HDBSCAN permite generar tópicos coherentes, puede llegar a clasificar más del 70 % de los documentos como valores atípicos. Al sustituir HDBSCAN por K-Means, se reduce significativamente la cantidad de documentos excluidos, aunque la coherencia temática disminuye. Este trabajo refiere que el algoritmo de agrupamiento no debe considerarse algo fijo, su elección del método debe ajustarse a la naturaleza del corpus. Una de las conclusiones a las que llega Groot et al. (2022) es que, en aplicaciones reales, un modelo que descarta la mayoría de los documentos puede resultar poco funcional, incluso si los grupos restantes son muy coherentes.

3.3 Enfoques basados en clasificación de textos

A diferencia de los métodos de agrupamiento, la clasificación requiere de datos etiquetados y categorías establecidas. En el trabajo de Bilski (2011), se lleva a cabo una exploración de los algoritmos de aprendizaje supervisado aplicados a la tarea de clasificación de documentos. Este estudio, publicado en el *International Journal of Electronics and Telecommunications* en el año 2011, busca solventar la creciente necesidad de organizar grandes volúmenes de datos digitales.

En el inicio de su artículo, Bilski discute la representación vectorial del texto, que es esencial para cualquier actividad de clasificación. Esta sección señala que la técnica llamada Bolsa de palabras (BoW) es la más utilizada, en donde los documentos se modelan como vectores de palabras, sin considerar el contexto ni la semántica del contenido, lo que da lugar a vectores con alta dimensionalidad. Es por ello que se suelen emplear métodos de reducción de dimensionalidad, además de técnicas de preprocesamiento como la tokenización, la eliminación de *stop-words* y la implementación de algoritmos de *stemming*. Posteriormente, el autor explora los principales algoritmos de aprendizaje automático utilizados en el manejo de texto para crear clasificadores de documentos.

El primer método que analiza son las redes neuronales artificiales, donde el autor resalta la capacidad de este algoritmo para gestionar datos de alta dimensionalidad y de adaptación al ruido, aunque su principal desventaja es el alto consumo de recursos computacionales. El artículo prosigue discutiendo los árboles de decisión, que proporcionan una estructura jerárquica, avanzando desde la parte superior hacia abajo y realizando una serie de decisiones

sucesivas sobre las características del documento. Estos funcionan mejor con un pequeño número de variables, pero pueden sobreajustarse al modelo al memorizar los documentos en lugar de generalizar a partir de ellos. Finalmente, Bilski analiza SVM, que son vistas por muchos como los clasificadores supervisados más precisos. Este algoritmo tiene la capacidad de procesar datos de alta dimensionalidad y combatir el sobreajuste, lo que las convierte en una opción preferida para el autor, a pesar de que su entrenamiento puede resultar computacionalmente exigente. En resumen, el artículo de Adrian Bilski ofrece un compendio de los métodos más relevantes, así como sus aplicaciones, beneficios y limitaciones de los algoritmos de clasificación.

Otra contribución importante para esta investigación es el artículo de Luna et al. (2025). Su estudio presenta pruebas a gran escala de documentos digitales, clasificando más de 600,000 documentos PDF mediante un proceso que combina reconocimiento de caracteres y aprendizaje automático. Se llevó a cabo una evaluación entre métodos clásicos como TF-IDF y representaciones basadas en transformadores, así como clasificadores tales como Bosques Aleatorios, SVM y BERT.

La combinación de TF-IDF con bosques aleatorios resultó en el mejor rendimiento, logrando una precisión que supera el 84 % y opacando al modelo más avanzado, qu es BERT. Los autores indican que BERT mostró un buen desempeño en documentos cortos y con bastante contexto. Sin embargo, mostró fallos al trabajar con clases similares o con textos en dos idiomas, además de requerir más capacidad de procesamiento. En este contexto, la representación TF-IDF combinada con bosques aleatorios demostró ser más efectiva al lidiar con el ruido que genera la extracción de texto.

Estas aportaciones son significativas para este trabajo de tesis, ya que permite señalar que los modelos más complejos no aseguran mejores resultados en todas las situaciones.

3.4 Colecciones de datos para clasificación

La clasificación de datos textuales se convirtió en una de las tareas más exploradas dentro del PLN, lo que motivó a la comunidad académica a crear y publicar conjuntos de datos para la experimentación y evaluación de modelos. Para la etapa de experimentación, la cual será explicada en el Capítulo 4 y 5del presente trabajo, se utilizó un conjunto de datos o *dataset* en español llamado *Spanish News Classification*, publicado por Morgado (2022)

y disponible públicamente en la plataforma Kaggle[2]. Este conjunto de datos fue construido con una herramienta de *web scraping* para entrenar modelos supervisados para su uso en recomendación de noticias. En la Figura 10 se muestra la distribución de cada una de las categorías. Está conformado por 1217 instancias distribuidas en siete categorías: Macroeconomía, Sustentabilidad, Innovación, Regulaciones, Alianzas, Reputación y Otros. El conjunto de documentos se encuentra estructurado de forma tabular y es posible descargarlo en un archivo CSV. Se compone de tres columnas conformadas por la URL de origen, el cuerpo de la noticia y su etiqueta temática.

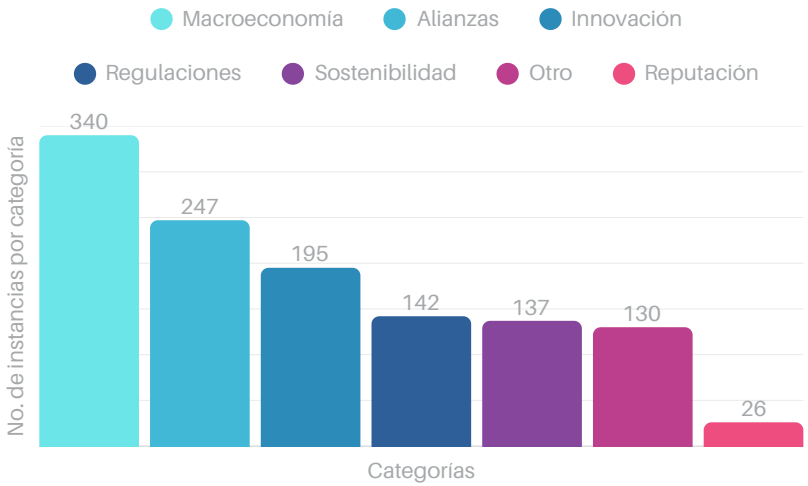

Figura 10. Distribución del conjunto de datos *Spanish News Classification*. Realizado por Morgado (2022)

La distribución de clases presenta un desbalance, predominando la clase de Macroeconomía sobre las demás. Asimismo, el texto contiene signos de puntuación, caracteres especiales, etc., lo que indica que es necesario una etapa de preprocesamiento antes de implementar los modelos.

Este conjunto de datos ha sido reutilizado en la misma plataforma de Kaggle, por un proyecto el cual consiste en la clasificación automática de noticias mediante la implementación de modelos multilingües como XLM-RoBERTa, lo que evidencia su utilidad en la evaluación de técnicas modernas en el aprendizaje automático y el PLN.

La selección de esta colección de datos para este trabajo, se justifica principalmente porque los documentos se encuentran en español, lo que resulta oportuno dado que la mayoría de los datos para clasificación textual publicados se encuentran en inglés y este proyecto se enfoca en la comunidad hispanohablante. Además, al contener siete categorías, el cor-

[2]https://www.kaggle.com/

pus ofrece suficiente variabilidad para evaluar la capacidad los algoritmos de adaptarse ante diversas temáticas. Finalmente, el hecho de que incluya etiquetas previamente definidas permite su utilización tanto en algoritmos de agrupamiento como de clasificación. En el caso de los modelos de aprendizaje no supervisado, las etiquetas definidas sirven como referencia para evaluar si los modelos están generando agrupaciones correctamente, comparando los resultados obtenidos con las categorías originales del corpus.

3.5 Discusión

Recapitulando los trabajos presentados en este capítulo, se puede concluir que no existe una única solución para llevar a cabo la organización automática de documentos digitales. Los resultados demuestran que la variación de ciertos factores, tales como la representación textual utilizada, la selección del algoritmo y la propia naturaleza del corpus, afecta significativamente los valores de las métricas empleadas.

En el trabajo de Saha (2023) se pueden extraer algunas ideas principales que justifican las decisiones tomadas para el Capítulo 4 de esta tesis. Una de ellas consiste en que las representaciones textuales influyen directamente en el desempeño de los algoritmos de agrupamiento y, como se plantea en el artículo de Luna et al. (2025), también repercuten en los resultados de los algoritmos de clasificación. El autor Saha (2023) hizo especial énfasis en que no existe una combinación ganadora ni una que sea universalmente aceptada como la mejor; además, para analizar los resultados, es importante observar el desempeño de acuerdo con varías métricas, no necesariamente hay que optar por un solo indicador. En concreto, es valioso experimentar con diversas técnicas de vectorización y modelos de aprendizaje automático para encontrar una solución más adaptable al problema que se desea resolver.

Por otro lado, la propuesta de Maarten (2022) con BERTopic demuestra que la coherencia semántica mejora sustancialmente cuando se utilizan representaciones más profundas, como las empleadas en los transformadores. Sin embargo, tanto en el artículo original como en el estudio ampliado de Groot et al. (2022), se coincide en que HDBSCAN, a pesar de ser un algoritmo más reciente que K-Medias, aún presenta ciertas limitaciones. Por ejemplo, en algunos casos clasifica más del 70 % de los documentos como valores atípicos, especialmente aquellos documentos más cortos. Esto afecta escenarios en los que el tamaño del corpus es pequeño y el usuario espera que sus documentos sean organizados y no descartados. La sustitución de HDBSCAN por K-Medias, como sugiere Groot et al. (2022), reduce la pérdida de documentos, pero conlleva una disminución en la coherencia temática de los grupos. Esto

resalta la importancia de experimentar con ambos algoritmos para determinar cuál funciona mejor con el conjunto de datos seleccionado, y si es preferible tener menos grupos o una mayor cobertura de los documentos, aunque con menor precisión semántica. Por ello, resulta importante experimentar tanto HDBSCAN como K-Medias.

Pasando a la clasificación supervisada, el trabajo de Bilski (2011) presenta una comparación diversa de varios algoritmos de clasificación. Esto permitió seleccionar tres de los algoritmos más destacados e implementarlos en la fase de experimentación del Capítulo 5. Los algoritmos escogidos fueron SVM, bosques aleatorios y redes neuronales. El objetivo no es repetir técnicas clásicas, sino emplear métodos respaldados empíricamente en el contexto de trabajar con texto. Además, en el artículo de Luna et al. (2025) se demostró que los mejores resultados se obtuvieron al combinar TF-IDF con bosques aleatorios. Esta evidencia justifica cuestionar si modelos más complejos, como BERT, garantizan un mejor desempeño en comparación con modelos más sencillos. Por ello, en el presente proyecto se incorpora también la utilización de técnicas tradicionales para la representación textual, como TF-IDF.

En conclusión, este trabajo aporta la combinación de algoritmos supervisados y no supervisados, lo que permite abordar el problema desde diferentes perspectivas. Por un lado, se aborda la dificultad de ordenar documentos que no cuentan con una etiqueta o nombre; por otro lado, y de manera complementaria, se ofrece la opción de clasificar nuevos documentos que ya cuentan con una categoría establecida. Además, se lleva a cabo una exploración con lo discutido en esta sección, donde se aborda desde la utilización de técnicas básicas de representación textual hasta métodos más avanzados, así como la implementación de diversos algoritmos de clasificación y agrupamiento. Esto permite observar el comportamiento de dichas combinaciones y contrastar o verificar las hipótesis planteadas por los autores en sus investigaciones. Además, se propone una solución que facilita el despliegue de los modelos con mejor desempeño mediante una interfaz gráfica que opera localmente, evitando así la necesidad de confiar documentos personales y posiblemente sensibles a servidores externos, garantizando la protección y accesibilidad del usuario en todo momento.

CAPÍTULO 4

Agrupamiento de textos

En esta sección experimental, se seleccionaron modelos de agrupamiento, así como una herramienta para el modelado de tópicos, la cual integra técnicas recientes del PLN. Con esto se busca encontrar algoritmos que sean capaces de identificar grupos representativos y útiles en un conjunto de documentos digitales.

4.1 Metodología experimental

La estructura implementada para realizar los experimentos sigue un enfoque comparativo basado en la combinación de técnicas de representación textual junto con algoritmos de agrupamiento. La Figura 11 presenta un diagrama que ilustra los pasos necesarios para aplicar los modelos computacionales de agrupamiento.

El proceso comienza con la selección de un conjunto de documentos digitales no etiquetados, es decir, textos que no cuentan con una categoría o clasificación previa. En la siguiente etapa, se realiza una limpieza y normalización del texto, donde se eliminan los elementos irrelevantes que podrían afectar el desempeño de los algoritmos. Posteriormente, cada documento se transforma en una representación numérica mediante tres técnicas de vectorización, las cuales permiten extraer las características de cada documento para que puedan ser procesados por los modelos.

Una vez obtenidas las representaciones textuales, se aplican algoritmos de agrupamiento y una técnica de modelado de tópicos. Estas técnicas identifican la similitud entre los documentos y organizan automáticamente en grupos aquellos que tienen contenido similar.

Al final, los conjuntos obtenidos se evalúan utilizando métricas especificas que permiten medir la separación entre grupos y la coherencia interna entre sus elementos.

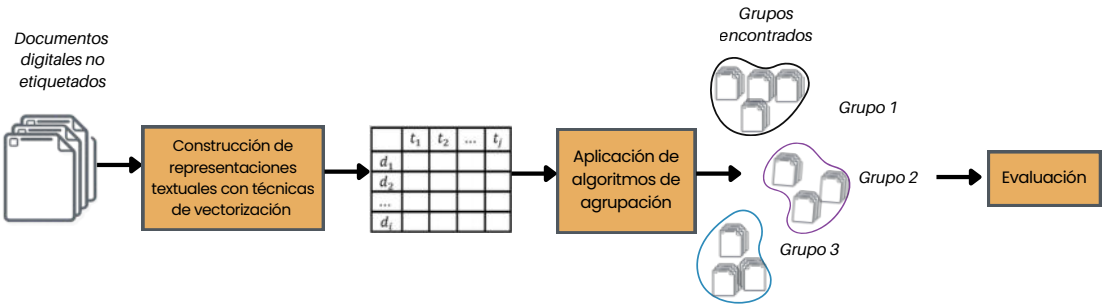

Figura 11. Flujo para el agrupamiento de documentos

4.2 Configuración

En esta sección se detallan las configuraciones que fueron implementadas para experimentar y evaluar los algoritmos de agrupamiento.

Colección de datos. El conjunto de datos utilizado se describe en la Sección 3.4.

Preprocesamiento del texto. Después de seleccionar el conjunto de datos, fue necesario hacer un preprocesamiento para retirar ruido del texto. Las tareas realizadas incluyen:

- Eliminación de caracteres no alfabéticos (números, signos especiales y puntuación innecesaria).
- Eliminación de espacios extras. Después de la eliminación de caracteres no alfabéticos, se generaron espacios en blanco redundantes entre los tokens, por lo que se eliminaron las secuencias de espacios por un único espacio.
- Normalización del texto, donde se hizo una conversión a minúsculas de todo el corpus para evitar duplicidades en el vocabulario.
- Eliminación de *stopwords* en español. Tras un análisis exploratorio del texto, se detectó que las palabras más frecuentes correspondían a *stopwords* en español, por ejemplo: *de, es, la, que, el, en, y, a*.

Representaciones textuales. Los documentos fueron transformados en representaciones numéricas utilizando las siguientes técnicas y configuraciones.

- **TF-IDF:** Se implementó un vocabulario de 5,000 términos y se eliminaron palabras con menos de dos repeticiones.
- **Word2Vec:** Se utilizaron *embeddings* entrenados en español mediante el algoritmo *Skip-gram*. Estas incrustaciones permitieron representar aproximadamente el 82.20 % de las 29, 341 palabras únicas presentes en la colección de datos empleada en la etapa de experimentación, mientras que el resto no se encontraba en el vocabulario del modelo.
- **Basadas en el uso de transformadores (BERT):** Se empleó un modelo ajustado sobre *bert-base-spanish-wwm-cased* basado en la arquitectura BERT, el cual ha sido entrenado específicamente para el idioma español (Cañete et al., 2020). Este modelo permite generar *embeddings* de 768 dimensiones que capturan el significado contextual de los textos en español.

Estas representaciones fueron utilizadas como entrada para los algoritmos de aprendizaje automático considerados a continuación.

Algoritmos de agrupamiento. Para la tarea de agrupamiento se evaluaron los modelos K-Medias, HDBSCAN y BERTopic. K-Medias se incluyó por ser un algoritmo ampliamente utilizado en PLN debido a que su implementación es sencilla y eficiente, además de permitir evaluar el impacto de las representaciones textuales bajo un número predefinido de grupos. HDBSCAN se eligió por su capacidad para determinar automáticamente el número de grupos y manejar ejemplares atípicos como ruido, lo que lo hace más adecuado para colecciones de datos con una estructura desconocida.

BERTopic se seleccionó por integrar técnicas recientes de PLN para el modelado de tópicos, como la reducción de dimensionalidad mediante UMAP, el agrupamiento basado en densidad con HDBSCAN y la representación de tópicos a través de c-TF-IDF, lo que permite generar etiquetas representativas para cada grupo.

Evaluación. Para HDBSCAN y BERTopic se emplearon sus configuraciones predeterminadas las cuales son detallados en el Anexo A. Para K-medias se probaron valores de $k \in \{5, 7, 10\}$, considerando que el corpus original contiene siete categorías conocidas.

Los modelos fueron evaluados con métricas internas de validación, como índice de silueta, índice Calinski-Harabasz y el índice Davies-Bouldin, interpretando los resultados de

manera general y no únicamente con el valor máximo de una métrica. También se tuvo en cuenta el número de grupos que los algoritmos identificaban para evaluar su desempeño, es decir, se hace un balance interpretativo tanto de los valores obtenidos en las métricas, como del número de grupos encontrados.

4.3 Experimentos

A continuación, se muestran los resultados derivados de evaluar las técnicas de representación textual y los algoritmos de agrupamiento considerados en este trabajo.

4.3.1 Evaluación del algoritmo K-Medias

En la Tabla 2 se presentan los valores obtenidos al aplicar el algoritmo de K-Medias a los tres tipos de representaciones textuales empleadas.

Tabla 2. Resultados obtenidos con el algoritmo K-Medias

Representación	Núm. de grupos	Silueta	Calinski-Harabász	Davies-Bouldin
TF-IDF	5	0.013	12.005	7.659
	7	0.013	10.289	6.474
	10	0.016	9.110	5.655
Word2Vec	5	0.115	107.516	2.690
	7	**0.070**	**87.720**	**2.732**
	10	0.071	68.922	2.609
Transformador	5	0.067	66.240	3.188
	7	0.059	52.871	3.376
	10	0.030	42.234	3.365

En el caso de TF-IDF, los valores del coeficiente de silueta varían entre 0.013 y 0.016, lo cual indica que existe muy poca separación entre los grupos y los documentos no se encuentran correctamente asignados a sus grupos.

En contraste, Word2Vec presenta una mejora en sus resultados. El coeficiente *Silhouette* alcanza 0.115 para $k = 5$ y el índice de Calinski-Harabász llega a 107.516, valores considerablemente superiores a los obtenidos con TF-IDF. Esto indica que las representaciones densas capturan mejor las temáticas del corpus, generando grupos más compactos y mejor separados. No obstante, al incrementar el número de grupos a 7 o 10, el coeficiente de silueta disminuyo a 0.070 y 0.071 respectivamente, por lo que los grupos encontrados se encuentran más cercanos entre sí, pero sus elementos internos más separados.

Por su parte, el Transformador mejora claramente los resultados respecto a TF-IDF, aunque no supera a Word2Vec en este algoritmo. Con un coeficiente de silueta de 0.067 para $k = 5$ refleja una separación aceptable, pero menor que la obtenida con Word2Vec. Esto sugiere que, representaciones profundas generadas a partir de un transformador, no necesariamente resultan en grupos con mayor separación entre grupos y cohesión en los elementos de cada grupo, al trabajar con un algoritmo basado en distancias como es K-Medias.

En resumen, *K-Means* presenta una mejora progresiva al pasar de representaciones basadas en frecuencia a representaciones semánticas densas, siendo Word2Vec la combinación que ofrece la separación más clara dentro de este algoritmo. Sin embargo, su necesidad de fijar un número de agrupamiento k cuando no se tiene gran certeza, podría ser una limitante para su implementación en problemas reales.

4.3.2 Evaluación del algoritmo HDBSCAN

En la Tabla 3 se muestran los resultados obtenidos al emplear los valores predeterminados de HDBSCAN en el conjunto de datos de evaluación.

Tabla 3. Resultados obtenidos con el algoritmo HDBSCAN

Representación	**Silueta**	**Calinski–Harabász**	**Davies–Bouldin**	**Clústeres**
TF-IDF	-0.031136	3.454990	5.894452	4
Word2Vec	0.029471	14.284424	5.385376	3
Transformador	-0.050992	18.166293	2.549734	7

Con TF-IDF, el índice de Silhouette es negativo (-0.031), lo que indica que, en promedio, los documentos están más cerca de grupos vecinos que del propio. Además, las demás métricas evidencian un solapamiento entre las agrupaciones, dado que el índice de Calinski–Harabasz presenta un valor menor en comparación. En comparación con los resultados obtenidos mediante K-Medias y TF-IDF, se observa que el conjunto de datos, tras ser procesado mediante representaciones textuales, no generó agrupaciones densas, por lo que a HDBSCAN le resultó más difícil identificar grupos representativos. El índice de Davies–Bouldin arrojó valores similares a los de K-Medias, lo que indica que los datos están dispersos y presentan baja separación. Lo interesante de la representación TF-IDF es que, aunque los resultados obtenidos no fueron óptimos, el algoritmo HDBSCAN logró identificar siete grupos representativos, coincidiendo con el número real de clases.

Con Word2Vec, el coeficiente de Silhouette mejora hasta 0.029, lo que indica un avance respecto a los demás experimentos, pero aún está lejos de alcanzar un valor ideal. El algorit-

mo de HDBSCAN con esta representación textual, identificó solo tres grupos, lo que sugiere que las densidades que se generaron a partir de este corpus, no fueron lo suficientemente densas para que el algoritmo de agrupamiento pudiera encontrar con claridad las estructuras.

El caso más relevante es el transformador. Aunque el valor de Silhouette es negativo (-0.050), lo que indica que en promedio, los puntos están más cerca de grupos vecinos que del propio, el índice de Davies–Bouldin es el más bajo (2.549), lo que indica una baja dispersión intra-grupo y una alta separación inter-grupo. Asimismo, se identificaron correctamente siete clústeres, lo cual coincide con la estructura temática presente en la colección de datos empleada para este capítulo. Esta combinación de resultados se interpreta en qué, los documentos individuales no siempre están cercanos al centroide de su grupo, pero se logra identificar una estructura temática coherente y que no cae en lo general.

En comparación con las demás configuraciones evaluadas, se observa que HDBSCAN, junto con una representación contextual, fue capaz de identificar siete grupos representativos, aunque estos no necesariamente pertenecen al grupo al que fueron asignados.

4.3.3 Evaluación de BERTopic

En la Tabla 4 se muestran las temáticas o tópicos que BERTopic encontró al ser implementado con los valores predeterminados, además del número de documentos pertenecientes a cada grupo.

Tabla 4. Resultados del modelado de tópicos mediante BERTopic

Tópico	Temática encontrada	Núm. documentos
0	0_millones_clientes_país_000	872
1	1_inflación_precios_alimentos_mayor	307
2	2_2019_gasto_consumo_frente	23
3	3_suiza_activos_blockchain_digitales	15

Como se puede observar, BERTopic no fue capaz de identificar las siete temáticas originales de la colección de datos empleada, sino que únicamente logró identificar cuatro, lo que sugiere que el modelo identificó las temáticas generales del corpus, pero no subtemas más específicos. Esto no necesariamente indica un bajo desempeño del modelo, sino que refleja las características del conjunto de datos utilizado, tanto en términos de desbalance entre clases, como de una falta de separación temática entre grupos.

Dado que BERTopic no cuenta con métricas internas de evaluación como las utilizadas en los experimentos con HDBSCAN o K-Medias, este ofrece la funcionalidad de estimar para

cada documento, un puntaje de pertenencia a su grupo asignado. En la Figura 12 se presenta el promedio de dichos puntajes por grupo, con el objetivo de analizar la pertenencia interna de los tópicos generados. Se puede observar que en tres de cuatro de los grupos, el puntaje promedio es superior a 0.8, lo que indica qué, en general, los documentos se encuentran correctamente asignados en su mayoría.

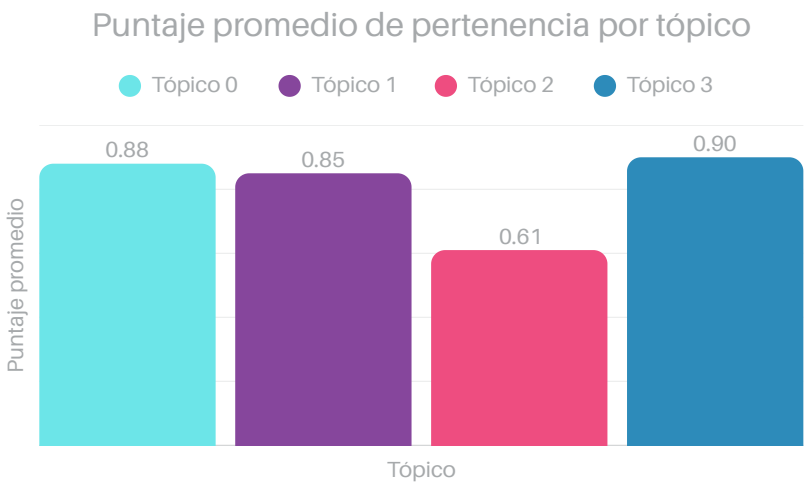

Figura 12. Puntaje de pertenencia promedio por tópico

En conjunto, estas características posicionan a BERTopic como una alternativa sostenible para el análisis de texto no supervisado, ya que no solo permite identificar agrupamientos, sino también interpretar y comprender la estructura temática del corpus.

4.4 Discusión

En el caso de K-Means, los resultados muestran una mejora al cambiar de representaciones basadas en frecuencia a representaciones semánticas densas. Sin embargo, este algoritmo presenta una clara desventaja respecto a HDBSCAN, debido a su necesidad de definir previamente un número fijo de grupos. En escenarios reales, esto podría ser una limitación significativa si se desconoce la estructura real del corpus. Por el contrario, HDBSCAN es capaz de adaptarse a la distribución de los datos y determinar patrones distintivos que le permiten encontrar una aproximación al número de agrupaciones presentes. Esto lo convierte en la opción más adecuada para escenarios reales donde se desconocen las clases existentes.

En este sentido, BERTopic reúne lo mejor del PLN. La combinación que ofrece BERTopic

al integrar embeddings contextuales, reducción dimensional mediante UMAP y agrupamiento con HDBSCAN logra proyectar los documentos en un espacio compacto donde se preservan las relaciones semánticas y, en su mayoría, se identifica cada temática. Es importante señalar que, en los experimentos realizados, BERTopic no logra identificar con exactitud las siete categorías reales del corpus. Sin embargo, fue elegido como el método preferido debido a su capacidad para modelar tópicos mediante la aplicación de c-TF-IDF por grupo. Con esta técnica, BERTopic identifica automáticamente las palabras más representativas de cada grupo y les asigna una etiqueta semántica coherente. Esta característica resulta especialmente relevante para el presente proyecto, dado que el objetivo no es solo agrupar documentos, sino generar nombres descriptivos y representativos de manera automática para las carpetas o categorías que el modelo detecte, eliminando la necesidad de que el usuario lo haga manualmente.

En resumen, para el escenario no supervisado, se seleccionó BERTopic, que integra representaciones profundas junto con los algoritmos HDBSCAN, UMAP y c-TF-IDF para el modelado de temas.

CAPÍTULO 5

Clasificación de textos

El objetivo principal de esta fase es entrenar modelos capaces de encontrar patrones distintivos en un conjunto de documentos digitales ya etiquetados y poder inferir sobre nuevos ejemplares.

5.1 Metodología experimental

En esta sección también se realizó una serie de pasos ordenados que permitieron la experimentación de estos modelos de clasificación. La idea central de este enfoque es contar con documentos que ya cuentan con una etiqueta o categoría conocida, lo que permite al modelo reconocer patrones asociados a cada clase.

En la Figura 13 se observa el flujo de trabajo específico para desarrollar un enfoque de aprendizaje supervisado para la clasificación de documentos digitales. El proceso inicia después de procesar el conjunto de datos de referencia etiquetado, en donde se realiza una limpieza del texto. Después de que el texto se encuentra limpio, se procede a convertir cada documento en una representación vectorial, lo que permite capturar características del contenido textual. Una vez obtenidas estas representaciones, se procede al entrenamiento de un modelo de clasificación. Durante esta etapa, el algoritmo aprende los patrones presentes en los vectores y hace una relación con las etiquetas correspondientes, generando así un modelo capaz de realizar predicciones.

Finalmente, se evalúa el desempeño del modelo a través de clasificar documentos desconocidos y comparando las predicciones generadas con las etiquetas reales. Para ello, se emplean métricas de evaluación que permiten medir qué tan bien el modelo ha aprendido a clasificar correctamente los documentos.

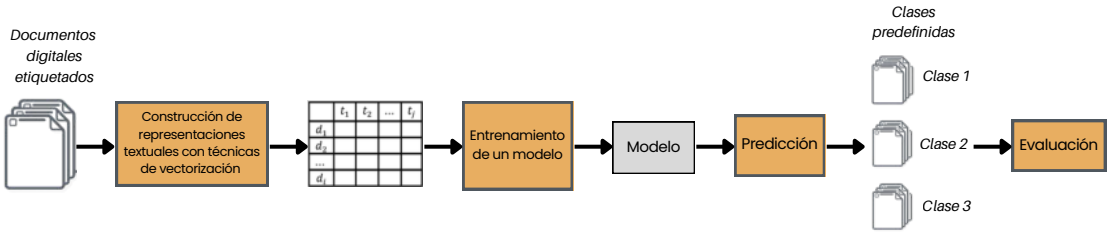

Figura 13. Flujo para la clasificación de documentos

5.2 Configuración

En esta sección se presenta la configuración de los modelos de clasificación para la evaluación de los experimentos.

Colección de datos. Se utilizó el mismo conjunto de datos utilizado con los algoritmos de agrupamiento, descritos en la Sección 3.4.

Representaciones textuales. Se utilizaron las mismas representaciones textuales mencionadas en la Sección 4.2.

Algoritmos de clasificación. Se seleccionaron tres clasificadores: bosques aleatorios, SVM y MLP. Estos clasificadores fueron elegidos por su uso frecuente en PLN y por representar distintas estrategias de trabajo con texto. Los bosques aleatorios aportan interpretabilidad y adaptabilidad ante ruido; el SVM lineal destaca por su desempeño en espacios de alta dimensionalidad; y las redes neuronales permiten modelar relaciones no lineales y patrones complejos presentes en el corpus.

Optimización de parámetros. Para llevar a cabo los experimentos de manera metódica, se empleó una estrategia de optimización de parámetros basada en *GridSearch*. En el Tabla 5 se resumen los parámetros implementados para cada modelo con dicha búsqueda inteligente. La estrategia consistió en que, para cada tipo de representación textual se exploraron diversas configuraciones de parámetros para los modelos clasificadores, con el objetivo de identificar aquella configuración con un mejor desempeño por representación textual. Como criterio principal de selección, se utilizó la métrica F1-macro.

Para el clasificador de bosques aleatorios, se ajustó el número de árboles con valores de 200 y 400, ya que un mayor número de árboles suele mejorar la estabilidad del modelo, pero

aumenta el costo computacional. También se evalúo la profundidad máxima de los árboles con valores restringidos (20 y 40) y una profundidad sin limite con el objetivo de reducir el riesgo de sobreajuste. Por último, se ajustó el parámetro de número mínimo de muestras para la división de nodos, buscando evitar particiones poco significativas.

En el caso de SVM, se configuró el nivel de regularización (C), en donde valores bajos tienden a favorecer la generalización, mientras que valores altos pueden conducir a sobreajuste. El tipo de kernel (lineal o no lineal) y el parámetro Gamma, que controla la influencia de cada punto en el caso del kernel no lineal, utilizando valores como *scale*, que ajusta automáticamente este valor de acuerdo con los datos.

Tabla 5. Valores empleados en la optimización de parámetros para los modelos de clasificación

Modelo	**Parámetro**	**Valores**
Bosques aleatorios	Número de árboles	{200, 400}
	Profundidad máxima	{Sin límite, 20, 40}
	Muestras para dividir	{2, 5}
SVM	Nivel de regularización (C)	{0.5, 1, 2}
	Tipo de frontera (kernel)	{Lineal, No lineal}
	Gamma	{scale, 0.01}
MLP	Estructura	{(128), (256), (128, 64)}
	Regularización	$\{10^{-4}, 10^{-3}\}$
	Iteraciones	{200}

Finalmente, en el algoritmo de MLP se exploró la estructura con una única capa oculta de 128 y 256 neuronas, así como una arquitectura de dos capas, con el fin de capturar relaciones no lineales sin aumentar demasiado la complejidad del modelo 128 y 64 neuronas. El parámetro de regularización (alpha) y el número máximo de iteraciones del modelo. Estas configuraciones permitieron analizar cómo diferentes valores en los parámetros influyen directamente a las representaciones textuales utilizadas y al desempeño final del modelo.

Para el perceptrón multicapa, se exploraron valores del parámetro de regularización α (0.0001 y 0.001), con el propósito de disminuir el riesgo de sobreajuste al penalizar configuraciones con pesos excesivamente complejos. Finalmente, el número máximo de iteraciones se estableció en 200, un valor considerado suficiente para permitir la convergencia del modelo teniendo en cuenta el tamaño del corpus empleado.

Evaluación. Las métricas reportadas son la media y la desviación estándar de: exactitud, F1-macro, precisión macro y sensibilidad macro. La métrica F1-macro se adoptó como métrica de referencia principal para la comparación entre modelos, dado que asigna el mismo

peso a todas las clases independientemente de su frecuencia, lo que resulta adecuado ante el desbalance de clases presente en el corpus.

5.3 Experimentos

En esta sección se presentan las configuraciones que presentaron mejores resultados al combinar distintas representaciones textuales con procesos de optimización de parámetros en los clasificadores.

5.3.1 Evaluación del clasificador Bosques Aleatorios

Tras la optimización de parámetros mediante *Grid Search* para el clasificador de bosques aleatorios, la Tabla 6 indica que el mejor desempeño se obtuvo con la representación TF-IDF y con una configuración para el clasificador consistente en 400 árboles con profundidad no limitada y un criterio de división basado en un mínimo de dos muestras, logrando una exactitud de 0.836 y un F1-macro de 0.792.

Tabla 6. Resultados de la optimización de parámetros con Bosques Aleatorios

Representación	Árboles	Prof. máx.	Min. división	Exactitud	F1-macro	Precisión	Sensibilidad
TF-IDF	**400**	**Sin límite**	**2**	**0.836**	**0.792**	**0.876**	**0.763**
Word2Vec	200	Sin límite	5	0.775	0.691	0.784	0.670
Transformador	400	20	5	0.773	0.643	0.690	0.635

Estos resultados sugieren que los bosques aleatorios son capaces de adaptarse a espacios de alta dimensionalidad como el generado por TF-IDF, el cual produce representaciones vectoriales de aproximadamente cinco mil dimensiones, a diferencia de métodos como Word2Vec y modelos basados en transformadores que generan representaciones de menor dimensionalidad (300 y 768 dimensiones, respectivamente).

En este contexto, los árboles de decisión pudieron identificar patrones relevantes en representaciones dispersas y sin información contextual. Por el contrario, las representaciones densas generadas por Word2Vec y el modelo basado en transformadores obtuvieron valores inferiores en F1-macro y sensibilidad, lo que sugiere bosques aleatorios no aprovecha completamente la información existente en espacios densos y contextuales.

5.3.2 Evaluación del clasificador SVM

En la Tabla 7 se muestran los resultados de SVM después de aplicar la optimización de parámetros. Es apreciable que SVM con representación TF-IDF y kernel lineal (C = 2) obtuvieron el mejor desempeño global, alcanzando una exactitud de 0.866 y un F1-macro de 0.849, siendo el modelo con mejor rendimiento entre todos los evaluados. Este comportamiento sugiere que los espacios de alta dimensionalidad como TF-IDF funcionan adecuadamente con una separación lineal para distinguir las clases de manera efectiva.

Tabla 7. Resultados de la optimización de parámetros con SVM

Representación	Kernel	C	Gamma	Exactitud	F1-macro	Precisión	Sensibilidad
TF-IDF	**Lineal**	**2**	–	**0.866**	**0.849**	**0.881**	**0.800**
Word2Vec	No lineal	2	scale	0.797	0.716	0.796	0.700
Transformador	No lineal	2	scale	0.833	0.797	0.853	0.771

Para Word2Vec y el transformador se utilizó kernel rbf, obteniendo resultados no tan favorecedores como con TF-IDF. Al igual que en el experimento anterior, en este conjunto de datos específico no superaron el desempeño de la representación basada en frecuencias, a pesar de su falta de información contextual.

En general, SVM demostró ser el modelo más adaptable y consistente entre las distintas representaciones.

5.3.3 Evaluación del clasificador MLP

En el modelo MLP se evaluaron distintas configuraciones para las capas ocultas, un parámetro de regularización alpha y el número máximo de iteraciones. Los resultados obtenidos que se presentan en la Tabla 8, la mejor configuración con MLP se logró con TF-IDF y capas ocultas de neuronas, alcanzando una exactitud de 0.839 y un F1-macro de 0.788.

Tabla 8. Resultados de la optimización de parámetros con MLP

Representación	α	Capas ocultas	Exactitud	F1-macro	Precisión	Sensibilidad
TF-IDF	**0.0001**	**{256}**	**0.839**	**0.788**	**0.859**	**0.765**
Word2Vec	0.0001	{256}	0.824	0.796	0.823	0.785
Transformador	0.0001	{128, 64}	0.805	0.783	0.815	0.773

A diferencia de SVM y los bosques aleatorios, el MLP mostró un desempeño más equilibrado entre las distintas representaciones. Word2Vec obtuvo un F1-macro relativamente cercano al de TF-IDF, lo que indica que la red neuronal puede adaptarse a representaciones

densas.

Sin embargo, en términos generales, el MLP no superó el desempeño alcanzado por SVM. Una posible explicación a esto, es que el tamaño del conjunto de datos no es lo suficientemente grande para explotar completamente el potencial de una red neuronal.

5.4 Discusión

El modelo SVM alcanzó la mayor precisión (0.866) y el mejor F1-macro (0.849) entre todos los clasificadores evaluados, superando tanto a los bosques aleatorios como al MLP.

A diferencia de las representaciones densas como Word2Vec o basados en transformadores, cuya profundidad semántica no se traduce en una mejor clasificación con algoritmos como los bosques aleatorios o SVM, TF-IDF captura de forma directa la distribución léxica de los documentos; es decir, el vocabulario representativo que utiliza resulta suficiente para distinguir categorías temáticas. Además, SVM no depende de grandes volúmenes de datos para obtener resultados favorables, lo que lo hace más adecuado para el corpus utilizado en este trabajo. En conclusión, para el caso de la clasificación supervisada, los resultados obtenidos justifican la elección de SVM con representación textual mediante TF-IDF para su integración en el desarrollo de la GUI.

Los algoritmos seleccionados serán integrados en la GUI, donde se evaluarán utilizando un conjunto de documentos reducido, con el objetivo de simular un entorno de uso real. Esto permitirá simular un escenario en el que un usuario típico dispone de una cantidad limitada de documentos para el entrenamiento y evaluación de los modelos.

CAPÍTULO 6

Desarrollo de una GUI para integrar los modelos de clasificación

Este capítulo describe de forma integral el proceso de creación, diseño e implementación de una interfaz gráfica de usuario para el análisis de documentos PDF, su agrupamiento temático, el entrenamiento de modelos de clasificación y la posterior inferencia temática sobre nuevos documentos. Se presenta la arquitectura adoptada, el diseño de interacción con el usuario y los componentes técnicos clave.

La interfaz fue creada usando la librería PyQT5[1] . De las razones principales para su elección es la integración directa con Python, ya que este lenguaje de programación ofrece una diversidad de librerías para el desarrollo de soluciones en PLN. Asimismo, PyQt5 cuenta con una amplia variedad de componentes interactivos prediseñados, además de funcionalidades que facilitan la construcción de interfaces sin requerir un desarrollo visual complejo.

6.1 Funcionalidad

La GUI resuelve un flujo completo de trabajo sobre la organización de documentos, realizando las siguientes tareas:

- Importación de archivos en formato PDF por lotes o individuales, los cuales serán el conjunto de documentos a categorizar.
- Extracción de texto, con el fin de obtener el contenido textual de los documentos y

[1] https://pypi.org/project/PyQt5/

hacerlo apto para su análisis mediante técnicas de procesamiento del lenguaje natural.

- Generación de representaciones vectoriales, para transformar el texto en una forma numérica que pueda ser interpretada por los algoritmos de aprendizaje automático.
- Agrupamiento por tópicos de documentos no etiquetados, con el propósito de identificar estructuras o patrones temáticos de manera automática en ausencia de categorías predefinidas.
- Entrenamiento de un clasificador supervisado, con el objetivo de aprender a asignar categorías a partir de documentos previamente etiquetados.
- Clasificación de nuevos documentos, para predecir la categoría de archivos no vistos previamente utilizando el modelo entrenado.
- Reubicación automática de archivos de acuerdo con las predicciones de los algoritmos, facilitando la organización del repositorio sin intervención manual del usuario.

6.2 Requerimientos

La interacción entre el usuario y la GUI ocurre de forma intuitiva y con base en las necesidades del usuario. En el diagrama mostrado en la Figura 14 se muestran las funcionalidades principales de la interfaz, las cuales también se representan en el menú de la GUI. Este diagrama permite delimitar las tareas que realiza la interfaz desarrollada.

Como se observa en la Figura 14, el actor principal es el usuario, quien puede ejecutar las siguientes acciones dentro del sistema: importar documentos, agrupar documentos por temática, entrenar modelos de clasificación, clasificar nuevos documentos, consultar modelos previamente entrenados. Cada uno de estos casos de uso representa una funcionalidad implementada en la interfaz, integrando internamente los procesos de extracción de texto, generación de representaciones vectoriales y aplicación de algoritmos de aprendizaje automático.

6.3 Arquitectura de la aplicación

Para el desarrollo de la GUI, se adoptó un patrón de estilo Modelo–Vista–Controlador (MVC) para separar las responsabilidades, facilitar las pruebas y el mantenimiento. Este patrón fue adoptado siguiendo los componentes mostrados en la Figura 15. La capa de vista

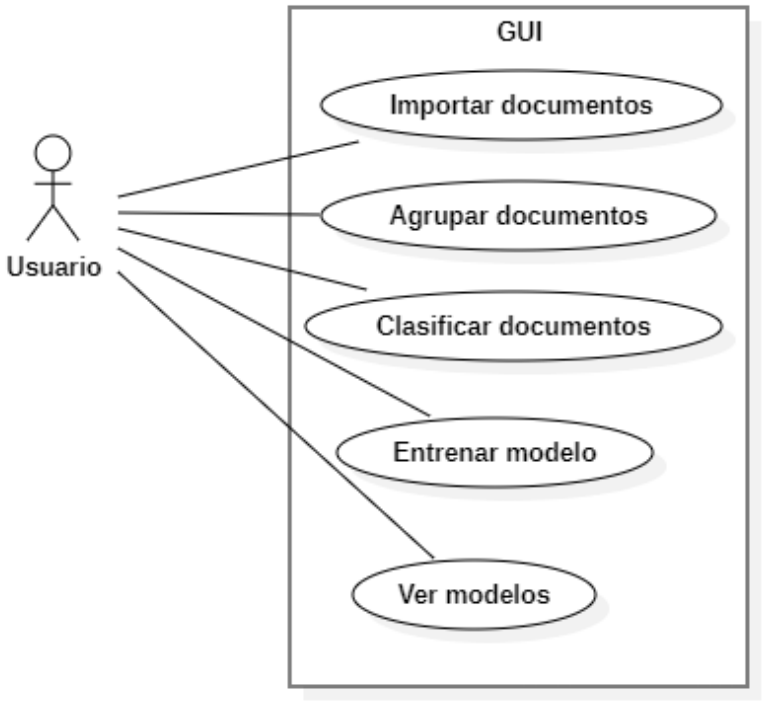

Figura 14. Diagrama de casos de uso de la GUI

está compuesta por páginas reutilizables en PyQt5, que contienen las funcionalidades visibles de la GUI de acuerdo con las tareas que puede realizar el usuario, las cuales se explicaron en el diagrama de casos de uso en la Sección 6.2; el controlador es quien orquesta las tareas, manda las señales del usuario a los apartados correspondientes, una vez que se genere el resultado, manda la señal para actualizar la GUI; la capa del modelo encapsula la lógica del *pipeline* o flujo de trabajo, que consiste en: extracción de texto, generación de *embeddings*, despliegue de los algoritmos de agrupamiento y clasificación.

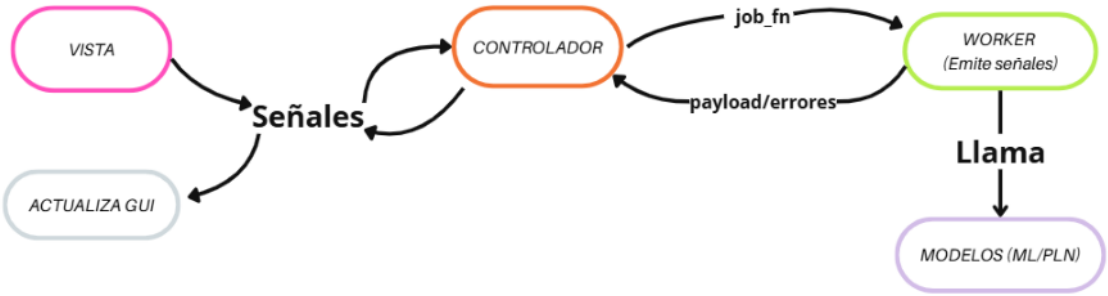

Figura 15. Relación entre componentes de la GUI

El flujo de ejecución dentro de la GUI inicia cuando un usuario interactúa con un elemento de la interfaz gráfica, lo que genera un evento que es capturado por el controlador. A partir de

esta acción, el controlador determina la operación a realizar.

Para evitar bloqueos en la interfaz, el controlador delega las tareas correspondientes de la capa modelo, como la extracción de texto, generación de representaciones vectoriales o aplicación de algoritmos de aprendizaje automático, a un componente que se ejecuta en un hilo secundario, del cual se explicará más en la Sección 6.3.3.1.

6.3.1 Capa Vista

En la capa de vista se diseñan los elementos gráficos de la aplicación. Esta capa incluye todas las páginas funcionales de la GUI: Agrupar, Entrenar Modelos, Clasificar, Ver Modelos, la ventana principal y un menú lateral para navegar entre páginas. Asimismo, se implementaron distintos diálogos para la visualización de resultados.

La vista es únicamente responsable de la presentación de la información y la captura de eventos del usuario, sin contener la lógica interna de procesamiento.

6.3.2 Capa de modelo

En parte de la arquitectura se definieron todas las funciones lógicas que permiten la aplicación de los algoritmos. Cada uno de los archivos corresponde a un paso en la integración del flujo de trabajo, los cuales se abordan en la aplicación de la metodología.

- **Extracción de texto.** La extracción de texto emplea dos librerías que detectan cuando el PDF contiene texto seleccionable. Si no hay texto, se aplica una librería especial para documentos escaneados. Para una mejor experiencia de usuario, se estableció como criterio procesar únicamente las primeras tres páginas de cada documento, lo que permite tener un contenido representativo sin necesidad de procesar en su totalidad el documento y ralentizar el proceso de la GUI. La función de extracción envía una llamada de estado para informar a la interfaz del avance y de incidencias o errores.

- **Representaciones vectoriales con aceleración por GPU.** Para ejecutar la tarea de agrupamiento, cada documento se transforma a un vector contextual basado en el transformador BERT, los cuales se generan en CUDA (*Compute Unified Device Architecture*) cuando hay GPU disponible. Esto permite acelerar el procesamiento de los documentos y realizar la tarea de agrupamiento en un menor tiempo, en comparación con no utilizar la GPU.

- **Agrupamiento temático.** Para la implementación del agrupamiento dentro de la inter-

faz se utilizó la librería BERTopic, que implementa los *embeddings* explicados en el punto anterior y realiza el proceso interno para identificar agrupamientos en los datos. Al final, se generan automáticamente representaciones temáticas interpretables para cada grupo detectado, esto permite asignar nombres descriptivos a las carpetas generadas por la aplicación, lo que facilita automatizar el proceso de organización de documentos.

- **Entrenamiento de modelos.** Para el entrenamiento del clasificador se empleó la combinación de TF-IDF como técnica de representación textual y SVM como algoritmo de clasificación. Esta decisión se fundamenta en los resultados obtenidos durante la etapa experimental del Capítulo 5, donde dicha combinación presentó el mejor desempeño en términos de exactitud.

  Las clases se derivan del nombre de la carpeta de origen de cada documento, lo que simplifica la preparación del conjunto etiquetado. Se realiza una partición de validación y se crea un reporte con las métricas de evaluación preestablecidas, para que el usuario pueda conocer el desempeño de su modelo, como se ve en la Figura 21. Los modelos se guardan en una carpeta especial que incluye el modelo, el codificador de etiquetas y un archivo *json* que documenta el entrenamiento. Este almacenamiento permite auditar y reutilizar modelos.

- **Clasificación y visualización de resultados.** Para la clasificación de nuevos documentos, se toma como base un modelo previamente entrenado, se realiza la extracción del texto para los nuevos documentos y se procede a generar la representación textual con TF-IDF; posteriormente, se predicen las categorías para cada documento de acuerdo con el modelo entrenado. Los resultados se presentan en una tabla y con opción de exportación manual. Todos los archivos que pasaron por el clasificador son movidos en automático a las carpetas predichas.

6.3.3 Capa controlador

La capa controladora actúa como intermediaria entre la vista y el modelo, permitiendo gestionar las interacciones del usuario con las distintas funcionalidades de la GUI. A partir de estas acciones, el controlador invoca los procesos necesarios de la capa del modelo, agrupar documentos, el entrenamiento de modelos o la clasificación. También se encarga de mostrar el estado en el que se encuentra la tarea específica, como la extracción de texto, generación de representaciones vectoriales y la aplicación de el algoritmo de aprendizaje automático.

Para mantener separada la vista y el modelo, se emplea un sistema de señales que

permite comunicar el progreso, estado y finalización de cada tarea. Estas señales son utilizadas para actualizar los elementos de la vista, como la barra de progreso y los mensajes informativos.

6.3.3.1. Hilos de trabajo y señales

Con el objetivo de evitar bloqueos de la interfaz durante operaciones demandantes como la generación de *embeddings*, se implementó un *worker* basado en QThread. QThread es una clase dentro de la librería PyQT5 que representa un hilo de ejecución paralelo al hilo principal: la GUI (Company, 2023). Se usa para evitar que la interfaz se bloquee al realizar tareas pesadas, como generar las representaciones textuales de cada documento.

El controlador de la GUI crea un objeto (*worker*) que contiene el código que se desea ejecutar en segundo plano. Este objeto se mueve a un hilo de ejecución mediante QThread (Company, 2023). Al finalizar su tarea, el *worker* emite señales para transmitir el resultado o la excepción al controlador.

6.4 Resultados

En esta sección se presentan los resultados del desarrollo de la interfaz gráfica, mostrando las principales pantallas del sistema y las funcionalidades implementadas. Estas vistas permiten observar la integración de los componentes descritos en la arquitectura, así como el flujo de interacción del usuario con la aplicación.

En la Figura 16 se presenta la vista principal de la GUI, donde se observa claramente el panel de opciones que ofrece.

6.4.1 Páginas funcionales

Cada operación principal se expone en una página dedicada:

- **Selección de documentos.** En la Figura 17 se muestra el desarrolló un *widget* (componente interactivo de la interfaz) para las opciones de *Agrupar*, *Entrenar Modelos* y *Clasificar* que permite arrastrar o abrir el diálogo de selección de archivos mediante doble clic. De igual manera, se cuenta con dos botones que permiten elegir entre seleccionar tandas de archivos PDF o simplemente subir carpetas completas de estos

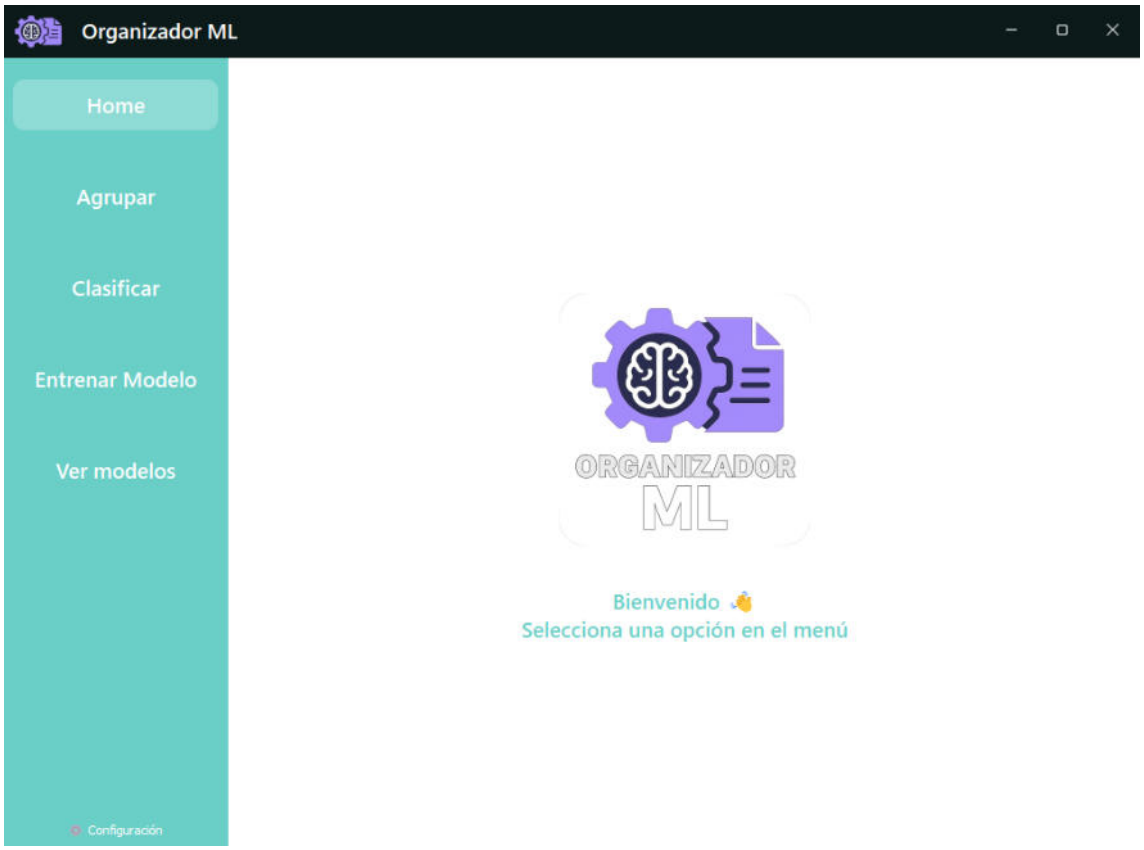

Figura 16. Ventana principal y navegación entre páginas. En la imagen se muestra la pantalla de bienvenida, donde se aprecia un menú lateral con todas las funcionalidades de la GUI.

archivos. Igualmente, se tiene un botón para limpiar el *widget* en caso de que se quiera cambiar los documentos seleccionados. Después de seleccionar los documentos deseados, se emite una señal *filesChanged* que permite habilitar las acciones que el usuario puede realizar dentro de cada página.

- **Agrupar**. En la Figura 18 se muestra un apartado para cargar documentos desde el ordenador. Cuenta con un botón de *agrupar* que ejecuta el proceso necesario para implementar BERTopic con los datos que el usuario proporciona. Durante el procedimiento, se muestra una barra de progreso y los pasos que se están realizando internamente. Cuando el algoritmo termina, se manda un mensaje de confirmación y se crean carpetas a partir de los tópicos encontrados.

- **Entrenar Modelos**. En la Figura 19 se muestra la funcionalidad de entrenar modelos personalizados. Con esta opción, el usuario nuevamente puede cargar sus documentos en el apartado correspondiente y, a través de accionar el botón, se entrena un modelo SVM que interpreta el nombre de la carpeta de origen como una etiqueta de clase. También se solicita al usuario ingresar un nombre para el modelo. Al finalizar el entre-

Figura 17. Componentes para cargar documentos a la GUI. En la pantalla se muestra el componente para cargar los documentos PDF, además de los botones que permiten elegir entre subir tandas de documentos o solo subir carpetas.

Figura 18. Página de agrupar documentos. La pantalla muestra el apartado de la GUI cuando el usuario desea agrupar sus documentos. Contiene una sección de carga de documentos y un botón para ejecutar la acción.

namiento, se generan metadatos como el número de documentos utilizados, clases y métricas de validación.

- **Clasificar.** En la Figura 20 se visualiza un selector desplegable de modelos ya entrena-

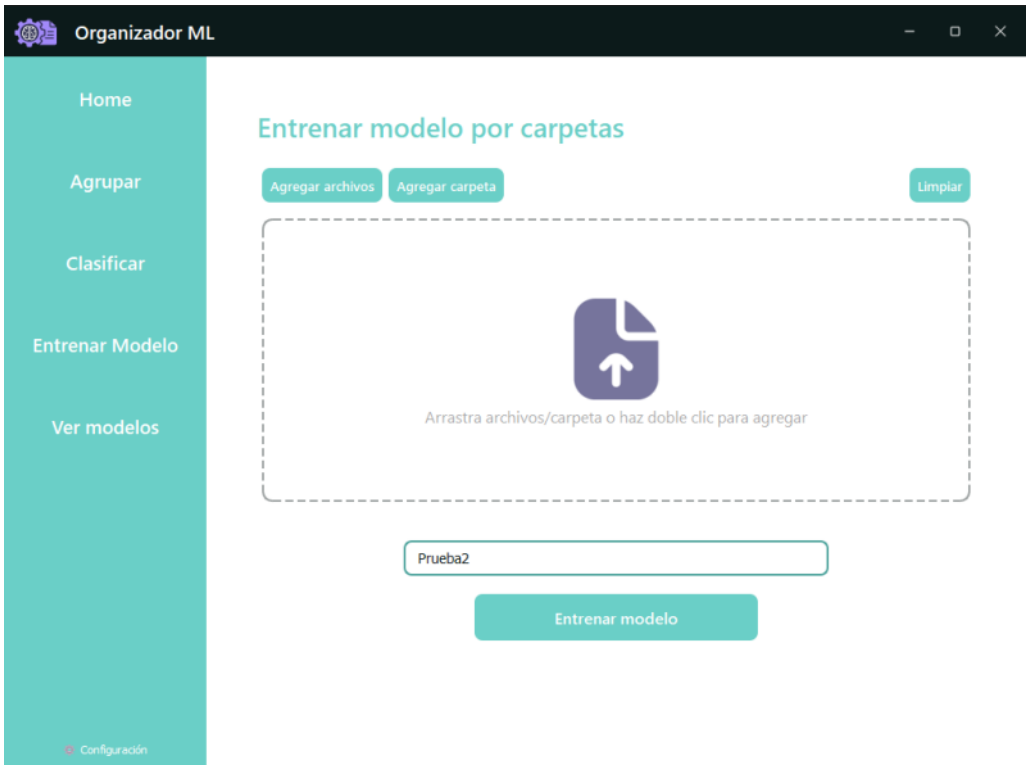

Figura 19. Página de entrenar modelos. La pantalla muestra el proceso de entrenamiento de un modelo SVM con los documentos del usuario.

dos, donde el usuario solamente puede seleccionar uno. Asimismo, se deben cargar los nuevos documentos que se deseen clasificar. Tras ejecutar el procedimiento, se abre un cuadro de diálogo con los resultados, en el cual se muestran las carpetas a las que el algoritmo asignó los documentos.

- **Ver modelos.** Recorre la carpeta de modelos y lee el archivo tipo *json* de cada modelo. Se construye una tabla con el nombre de modelo, número de documentos utilizados para entrenar el modelo, las clases empleadas, exactitud reportada, fecha de actualización y ruta. Un doble clic abre un diálogo con el informe de clasificación completo. En la Figura 21 se observan ejemplos de modelos previamente entrenados.

6.4.2 Configuración experimental

En esta sección se detalla la configuración experimental implementada para probar las funcionalidades de la GUI.

Conjunto de datos. Para probar la GUI, se recopiló un conjunto de documentos de índole personal. El conjunto de datos seleccionado consta de un total de 54 documentos, divididos en 7 categorías, las cuales son: Acuse, Almacén, Análisis, Contestación, Cotización, Formato, Resumen. En la Figura 22 se observa la distribución de instancias por clase. Es

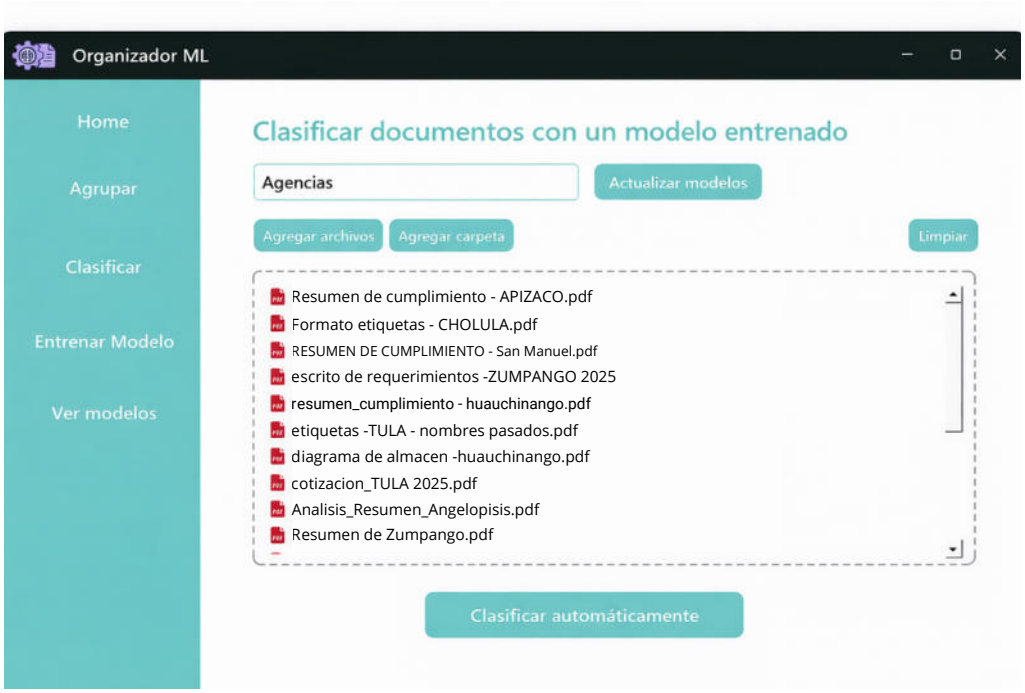

Figura 20. Página de Clasificar documentos. La pantalla muestra el apartado correspondiente para ejecutar la clasificación de nuevos documentos.

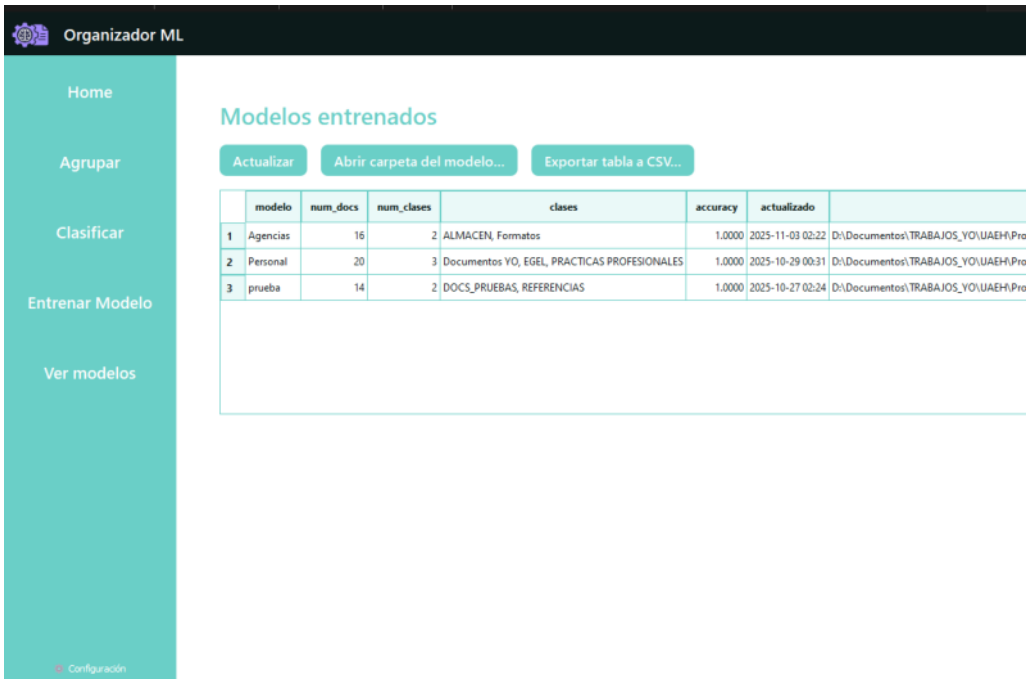

Figura 21. Página de Ver modelos. La pantalla presenta una tabla con todos los modelos entrenados por el usuario, incluyendo el nombre del modelo, el número de documentos utilizados, el número de clases y sus nombres, la exactitud reportada, la fecha de actualización y la ruta correspondiente.

necesario señalar que, los documentos empleados se encontraban previamente etiquetados, es decir, ya se conocía la categoría a la que pertenecía cada uno. Esto con el objetivo de tener una referencia sustancial y determinar si los modelos de clasificación supervisada y no supervisada tuvieron un buen desempeño.

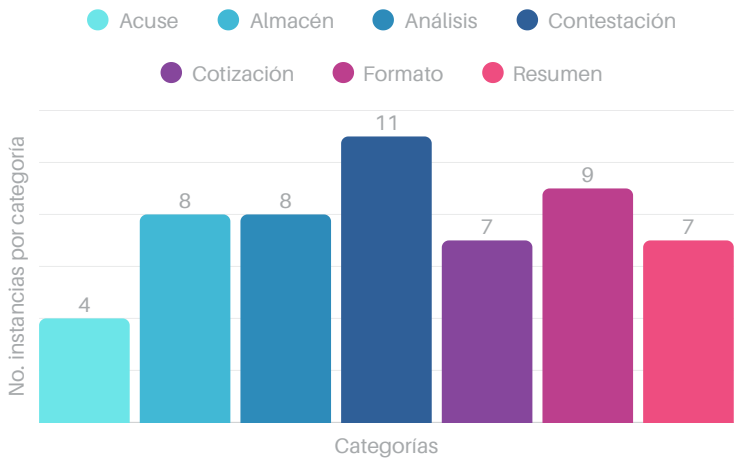

Figura 22. Distribución del conjunto de prueba para probar la interfaz

Algoritmo de agrupamiento seleccionado. De acuerdo a lo discutido en el Capítulo 4, el modelo con mayores ventajas fue BERTopic, dada su utilidad en el modelado de tópicos. Es por ello que se implementó como herramienta para el agrupamiento de documentos digitales. De igual manera, se configuró con sus valores predeterminados con un modelo preentrendo en español para la generación de *embeddings*, UMAP para la reducción de dimensionalidad y HDBSCAN como algoritmo de agrupamiento.

Algoritmo de clasificación seleccionado. Para la clasificación de nuevos documentos digitales en la GUI, se utilizaron los algoritmos discutidos en el Capitulo 5, donde se determinó que la mejor configuración fue la combinación de TF-IDF con un vocabulario de 5, 000 términos y SVM con un kernel lineal y un nivel de regularización igual a 2.

6.4.3 Demostración de la GUI

En este apartado se muestran el funcionamiento de la GUI al realizar experimentos de funcionalidad para el apartado de agrupamiento utilizando la colección de documentos digitales que se detalla en la Sección 6.4.2.

6.4.3.1. Entrenamiento de modelos de agrupamiento

Se hicieron diversos experimentos con el conjunto de datos seleccionados, donde se cargaron archivos de las 7 categorías y en diferentes combinaciones. En un experimento general, se cargaron los 54 documentos divididos en las 7 categorías. Se ejecutó la funcionalidad de agrupamiento y la GUI realizó todo el proceso interno para crear las representaciones textuales y la implementación de BERTopic.

En la Tabla 9 se hace un resumen de los resultados obtenidos después de aplicar BERTopic en el conjunto de datos.

Tabla 9. Resultados del modelado de tópicos con BERTopic en un corpus de 54 documentos

Temática encontrada	**Núm. de agruparDocumentos**
0_residuos_manejo_especial_hidalgo	16
1_caracteristicas_ingreso_cretib_residuo	9
2_hf_gg_gases_reaccion	8
3_contar_retencion_canaletas_deben	8
4_cumplen_ambiental_pendiente_analisis	7
5_ambiental_residuos_peligroso_materia	6

Es apreciable que el modelo logró encontrar seis temáticas de siete. El primer grupo fue el que presentó más problemas, ya que mezcló las clases de Acuse, Contestación y Cotización. Por el contrario, los demás grupos presentaron excelentes resultados, al lograr identificar y agrupar los documentos pertenecientes a una misma categoría. Se puede señalar que BERTopic, a través de cada uno de sus módulos, es capaz de encontrar patrones significativos en los datos y tener un entendimiento profundo del contenido textual.

6.4.3.2. Entrenamiento de modelos de clasificación

Se probó igualmente el apartado de clasificación, donde previamente se entrenó un modelo con documentos de dos categorías: Formatos y Análisis. Para probar su efectividad, se ingresaron nuevos ejemplares exclusivamente de la clase Formatos, ejemplares que el modelo entrenado desconocía. Explicado el contexto, en la Figura 23 se muestra el resultado que arrojó la GUI aplicando las técnicas de clasificación.

Como se puede observar, el clasificador fue capaz de detectar que todos los nuevos ejemplares efectivamente correspondían a la clase "Formatos", obteniendo una exactitud superior a 0.9 (el valor máximo que se puede alcanzar es 1). Esto indica que la configuración interna del modelo SVM se adapta al conjunto de datos, permitiendo detectar patrones ca-

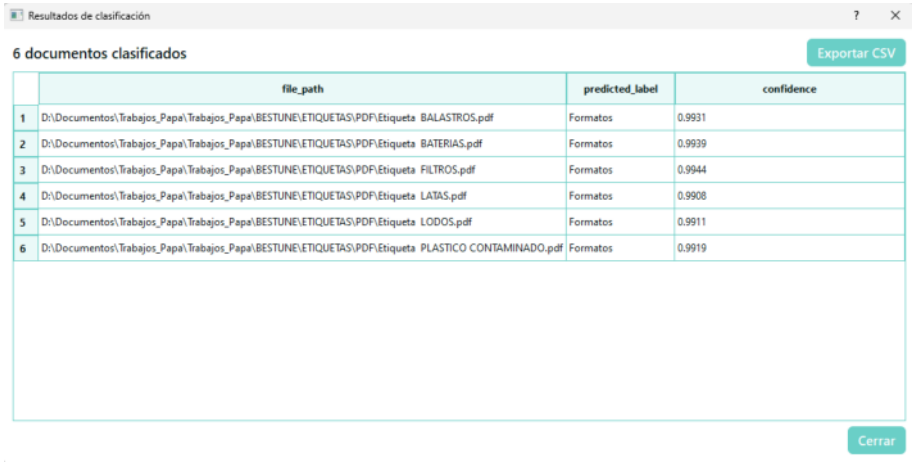

Figura 23. Prueba del clasificador en la GUI. Se observan los resultados obtenidos al probar el clasificador con nuevos ejemplos

racterísticos de cada clase.

Para concluir este capítulo, se puede resumir en que el diseño visual e interno de la GUI permitió realizar tareas computacionalmente demandantes, garantizando que la experiencia de usuario sea fluida y sin complicaciones. Además, se logró integrar el flujo completo de procesamiento necesario para organizar documentos digitales, el cual abarca desde la carga de archivos, su limpieza, vectorización, entrenamiento de modelos, la clasificación de nuevos documentos y el agrupamiento de documentos sin categoría. Los resultados obtenidos permiten validar la funcionalidad de la interfaz y la viabilidad de implementar modelos como BERTopic y SVM en entornos locales, privados y a baja escala.

Conclusiones y trabajo futuro

6.5 Conclusiones

Con el presente proyecto se demostró que la clasificación y el agrupamiento automático de una pequeña cantidad de documentos es completamente posible, permitiendo su integración en un entorno local y privado, sin depender de servicios en la nube ni de infraestructura externa.

Tras ejecutar los experimentos, se obtuvieron los resultados correspondientes a cada combinación planteada, seleccionándose las configuraciones con mejores desempeños según las métricas empleadas. En el apartado de agrupamiento, los algoritmos seleccionados fueron el modelo BERTopic junto con el agrupador HDBSCAN, los cuales permitieron generar agrupamientos temáticos. Para la clasificación supervisada, se eligió la combinación de TF-IDF con el modelo SVM. Esta combinación ofreció un desempeño destacable de acuerdo a los valores de sus métricas.

Un resultado relevante de este trabajo, es que una representación textual tradicional, como TF-IDF combinada con una adecuada configuración de parámetros, demostró tener un mejor desempeño frente a técnicas avanzadas basadas en transformadores. También se destaca el desempeño de HDBSCAN, que demostró ser capaz de adaptarse a escenarios en los que el número de grupos era desconocido, además de su habilidad para detectar estructuras de datos complejas.

Durante la integración de los algoritmos con la interfaz desarrollada, los resultados obtenidos fueron satisfactorios. La GUI logró clasificar y agrupar documentos con un nivel adecuado de coherencia y relevancia temática, facilitando la organización automática de los documentos digitales cargados. Además, la interfaz cuenta con un diseño intuitivo que permite su uso a cualquier usuario no experto.

Este trabajo confirma que la integración de técnicas de PLN y aprendizaje automático

puede ofrecer soluciones reales para la organización documental en entornos locales. La combinación de modelos clásicos y modernos permitió obtener una perspectiva amplia sobre cuales son las ventajas y desventajas de cada algoritmo al ser implementados en este dominio.

6.6 Trabajo futuro

El trabajo futuro para este proyecto se describe en los siguientes puntos:

- Ampliación del conjunto de datos. Una de las principales limitaciones actuales de la interfaz gráfica de usuario es la disponibilidad de documentos para realizar pruebas. Por ello, se contempla evaluar el rendimiento de la GUI utilizando una mayor cantidad y variedad de documentos. Esto permitirá no solo mejorar los modelos de agrupamiento y clasificación, sino también validar su escalabilidad y adaptabilidad a distintos contextos.
- En cuanto a la experimentación, se planea ampliar la evaluación de los parámetros de los algoritmos mediante un rango más amplio de sus valores.
- Para la GUI, se puede ampliar las funcionalidades del apartado de agrupamiento mediante la creación de dos botones adicionales relacionados con la gestión de los hiperparámetros del algoritmo de agrupamiento:

    1. Visualización de hiperparámetros: permitirá al usuario observar cómo se comporta el algoritmo al modificar hiperparámetros específicos, como el número de grupos. Con esto, se busca que el usuario pueda determinar qué hiperparámetros se adaptan mejor a sus documentos.
    2. Actualizar hiperparámetros: Este botón está vinculado al anterior. Una vez que el usuario haya observado con qué características el algoritmo agrupa mejor, podrá actualizar los hiperparámetros y comenzar a agrupar sus documentos con estos nuevos valores.

- Clasificación manual asistida. Se planea agregar una funcionalidad mediante la cual la interfaz gráfica muestre una vista previa del PDF cargado y sugiera a qué carpeta podría pertenecer. El usuario tendrá la opción de aceptar o rechazar esta sugerencia, lo que permitirá aumentar la precisión en casos complejos o ambiguos, además de facilitar la retroalimentación al modelo.

Referencias

Antonelli, L., & Guarracino, M. R. (2023). Special Issue on Supervised and Unsupervised Classification Algorithms—Foreword from Guest Editors. *Algorithms*, *16*(3). https://doi.org/10.3390/a16030145

Bilski, A. (2011). A Review of Artificial Intelligence Algorithms in Document Classification. *International Journal of Electronics and Telecommunications*, *57*, 263-270. https://doi.org/10.2478/v10177-011-0035-6

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning* (1.ª ed.) [Hardcover published: 17 August 2006; Softcover published: 23 August 2016. Series ISSN: 1613-9011; Series E-ISSN: 2197-4128. eBook Packages: Computer Science, Computer Science (R0). Copyright: Springer-Verlag New York 2006]. Springer New York, NY.

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5-32. https://doi.org/10.1023/A:1010933404324

Burges, C. J. (1998). A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, *2*(2), 121-167. https://doi.org/10.1023/A:1009715923555

Campello, R., Moulavi, D., & Sander, J. (2013). Density-Based Clustering Based on Hierarchical Density Estimates. *Advances in Knowledge Discovery and Data Mining*, 160-172.

Campello, R., Moulavi, D., Zimek, A., & Sander, J. (2015). Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection. *ACM Transactions on Knowledge Discovery from Data*, *10*, 1-51. https://doi.org/10.1145/2733381

Cañete, J., et al. (2020). bert-base-spanish-wwm-cased. Consultado el 26 de marzo de 2026, desde https://huggingface.co/dccuchile/bert-base-spanish-wwm-cased

Chicala, J., Arízaga Gamboa, J., & Alvarado Unamuno, E. (2021). Análisis y desarrollo de interfaz gráfica de usuario (GUI). *Serie Científica de la Universidad de las Ciencias Informáticas*, *14*(8), 73-84. http://publicaciones.uci.cu

Chicco, D., Campagner, A., Spagnolo, A., Ciucci, D., & Jurman, G. (2025). The Silhouette coefficient and the Davies-Bouldin index are more informative than Dunn index,

Calinski-Harabasz index, Shannon entropy, and Gap statistic for unsupervised clustering internal evaluation of two convex clusters. *PeerJ Computer Science*, *11*, e3309. https://doi.org/10.7717/peerj-cs.3309

Chowdhary, K. (2020). *Fundamentals of Artificial Intelligence*. Springer. https://doi.org/https://doi.org/10.1007/978-81-322-3972-7

Company, T. Q. (2023). *Thread — PyQT Documentation v5.15.7*. Consultado el 16 de marzo de 2026, desde https://www.riverbankcomputing.com/static/Docs/PyQt5/api/qtcore/qthread.html#qthread

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms* (4ta). The MIT Press.

Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, *20*, 273-297.

Darji, H., Mitrović, J., & Granitzer, M. (2023). German BERT Model for Legal Named Entity Recognition. *Proceedings of the 15th International Conference on Agents and Artificial Intelligence*, 723-728. https://doi.org/10.5220/0011749400003393

Deleón, J. A. R. (2007). *Manual de autoformación en administración de documentos y gestión de archivos* (1.ª ed.). Ciudad de México. https://retaip.infocdmx.org.mx/documentos/material_apoyo/manuales/ManualArchivo.pdf

Domínguez, B., Fernanda, L., Lavayen, L., Clara, A., Romero, S., & Daniel, J. (2024). Ventajas de la automatización de la gestión por procesos [Instituto Superior Tecnológico Japón]. *Revista académica*.

Fowler, M. (2003). *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional.

Galitz, W. O. (2007). *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques* (3rd). Wiley Publishing, Inc.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. Consultado el 20 de noviembre de 2025, desde http://www.deeplearningbook.org

Groot, M., Aliannejadi, M., & Haas, M. R. (2022). Experiments on Generalizability of BERTopic on Multi-Domain Short Text. https://arxiv.org/abs/2212.08459

Haris, E., Cohn, A. G., & Stell, J. G. (2026). A semantic and context-dependent approach to the interpretation of 'near' in historical English Lake District narratives. *International Journal of Geographical Information Science*, *40*(3), 727-758. https://doi.org/10.1080/13658816.2025.2588359

Haykin, S. S. (2009). *Neural Networks and Learning Machines* (3ra). Prentice Hall.

James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). *An Introduction to Statistical Learning with Applications in Python*. Springer Cham.

Jawahar, G., Sagot, B., & Seddah, D. (2019). What Does BERT Learn about the Structure of Language? En A. Korhonen, D. Traum & L. Màrquez (Eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 3651-3657). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1356

Joachims, T. (1998). Text Categorization with Support Vector Machines. *Proc. European Conf. Machine Learning (ECML'98)*. https://doi.org/10.17877/DE290R-5097

Jurafsky, D., & Martin, J. H. (2026). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models* (3rd) [Online manuscript released January 6, 2026]. https://web.stanford.edu/~jurafsky/slp3/

Kamble, S. (2024, mayo). Support Vector Machine (SVM) VCET TechZette. https://techz.vcet.edu.in/2024/05/04/support-vector-machine-svm/

Lamba, M., & Madhusudhan, M. (2022). Text Pre-Processing. En *Text Mining for Information Professionals: An Uncharted Territory* (pp. 79-103). Springer International Publishing. https://doi.org/10.1007/978-3-030-85085-2_3

Luna, S. L., Garigliotti, D., Plumed, F. M., & Ramírez, C. F. (2025). Automatic PDF Document Classification with Machine Learning. En V. Julian, D. Camacho, H. Yin, J. M. Alberola, V. B. Nogueira, P. Novais & A. Tallón-Ballesteros (Eds.), *Intelligent Data Engineering and Automated Learning IDEAL 2024* (pp. 447-459). Springer Nature Switzerland.

Maarten, G. (2022). BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv preprint arXiv:2203.05794*. https://arxiv.org/abs/2203.05794

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Markoulidakis, J., Rallis, I., Georgoulas, I., Kopsiaftis, G., Doulamis, A., & Doulamis, N. (2021). Multiclass Confusion Matrix Reduction Method and Its Application on Net Promoter Score Classification Problem. *Technologies*, *9*, 81. https://doi.org/10.3390/technologies9040081

Mary. (2026, febrero). *Modelos de Transformación: ¿Qué son? ¿Por qué son importantes en IA?* Consultado el 16 de marzo de 2026, desde https://datascientest.com/es/modelos-de-transformacion-que-son

McInnes, L., Healy, J., & Astels, S. (2016). *How HDBSCAN Works — hdbscan 0.8.1 documentation*. Consultado el 16 de marzo de 2026, desde https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

McInnes, L., Healy, J., & Astels, S. (2017). hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, *2*, 205. https://doi.org/10.21105/joss.00205

Mikolov, T. e. a. (2013). Efficient Estimation of Word Representations in Vector Space. *ICLR Workshop*.

Miller, C., Portlock, T., Nyaga, D. M., & O'Sullivan, J. M. (2024). A review of model evaluation metrics for machine learning in genetics and genomics. *Frontiers in Bioinformatics*, *4*. https://api.semanticscholar.org/CorpusID:272580112

Molina, D. L. L., & Tapia, J. C. (2020). La tecnología y los riesgos sobre la privacidad y protección de datos [Artículo de investigación]. *Polo del Conocimiento*, *5*(11), 565-590. https://doi.org/10.23857/pc.v5i11.2009

Morgado, K. (2022). *Spanish News Classification Dataset*. Consultado el 26 de febrero de 2026, desde https://www.kaggle.com/datasets/kevinmorgado/spanish-news-classification

Russell, S. J., & Norvig, P. (2004). *Inteligencia artificial: un enfoque moderno* (2da). PRENTICE HALL.

Saha, R. (2023). Influence of various text embeddings on clustering performance in NLP. https://arxiv.org/abs/2305.03144

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Comput. Surv.*, *34*(1), 1-47. https://doi.org/10.1145/505282.505283

Subakti, A., Murfi, H., & Hariadi, N. (2022). The performance of BERT as data representation of text clustering. *Journal of Big Data*, *9*(1), 15. https://doi.org/10.1186/s40537-022-00564-9

Ting, K. M. (2024). Confusion Matrix. *Encyclopedia of Machine Learning*. https://api.semanticscholar.org/CorpusID:16307526

Uddin, S., & Lu, H. (2024). Dataset meta-level and statistical features affect machine learning performance. *Scientific Reports*, *14*(1), 1670. https://doi.org/10.1038/s41598-024-51825-x

Vaswani, A. e. a. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems (NeurIPS)*.

Yehoshua, R. (2023, julio). *Random Forests*. Consultado el 16 de marzo de 2026, desde https://medium.com/@roiyeho/random-forests-98892261dc49

Zhang, H., & Shafiq, O. (2024). Survey of transformers and towards ensemble learning using transformers for natural language processing. *Journal of Big Data*, *11*(1), 25. https://doi.org/10.1186/s40537-023-00842-0

Zhang, Y., Zhou, Y., & Yao, J. (2020). Feature Extraction with TF-IDF and Game-Theoretic Shadowed Sets. En M.-J. Lesot, S. Vieira, M. Z. Reformat, J. P. Carvalho, A. Wilbik, B. Bouchon-Meunier & R. R. Yager (Eds.), *Information Processing and Management of*

Uncertainty in Knowledge-Based Systems (pp. 722-733). Springer International Publishing.

Anexos

ANEXO A

Configuración predeterminada de los modelos

En este anexo se presentan los valores de los parámetros predeterminados utilizados en los algoritmos HDBSCAN y BERTopic.

Tabla 10. Configuración predeterminada de los parámetros de HDBSCAN

Parámetro	**Valor predeterminado**
Tamaño mínimo de grupo	5
Sensibilidad al ruido y densidad	Igual al tamaño mínimo de grupo si no se especifica
Medida de similitud entre documentos	Distancia euclidiana
Método de selección de grupos	Exceso de masa
Nivel de conservadurismo del modelo	1.0
Cálculo de probabilidades de pertenencia	No activado

Tabla 11. Configuración predeterminada de parámetros de BERTopic

Método	**Parámetro**	**Valor**
UMAP	Dimensionalidad del espacio reducido	5
	Número de vecinos considerados	15
	Distancia mínima entre puntos	0.0
HDBSCAN	Tamaño mínimo de grupo	10
	Sensibilidad al ruido	No especificado
	Medida de similitud	Distancia euclidiana
	Método de selección de grupos	Exceso de masa
c-TF-IDF	Número de palabras representativas por tópico	10
	Rango de n-gramas	(1, 1)