

GCID - Grado en Ciencia e Ingeniería de datos

**Desarrollo de Sistemas de
Procesamiento de Lenguaje Natural
para la Gestión Avanzada de
Documentos Legislativos en el
Parlamento de Canarias**

Victoria Torres Rodríguez

Tutores:

Cayetano Guerra Artal

María Elena Sánchez Nielsen

Julio 2024

CONTENIDOS

ÍNDICE DE FIGURAS	1
ÍNDICE DE TABLAS	2
1. INTRODUCCIÓN	5
1.1. Presentación del ámbito de trabajo	5
1.2. Objetivos.....	8
1.3. Recursos utilizados de hardware y software	9
1.4. Competencias	12
2. ANÁLISIS	13
2.1. Estado del arte	13
3. DISEÑO	19
3.1. Estructura de los datos originales.....	19
3.2. Diseño de las bases de datos.....	19
3.3. Estructura de la aplicación web de las herramientas desarrolladas	21
4. METODOLOGÍA	23
4.1. Descripción de la metodología	23
4.2. Comprensión del Negocio	23
4.3. Comprensión de los datos	24
4.4. Modelado.....	24
4.5. Evaluación	24
4.6. Despliegue	25
5. DESARROLLO	27
5.1. Obtención de los datos	27
5.2. Asistencia en documentación parlamentaria	28
5.2.1 Introducción	28
5.2.2 Elección del modelo	29
5.2.3 Conocimiento base del modelo sobre el Parlamento de Canarias	30
5.2.4 Generación de datasets necesarios para el ajuste fino.....	32
5.2.5 Preprocesamiento de los datasets necesarios para el ajuste fino.	35
5.2.6 Entrenamiento secuencial.....	35
5.2.7 Resultados del entrenamiento secuencial	41
5.2.8 Entrenamiento combinado	43
5.2.9 Resultados del entrenamiento combinado	44
5.2.10 Entrenamiento combinado y uso de memoria vectorial	45

5.2.11	Corrección con modelos GPT	47
5.2.12	Conclusión.....	51
5.3	Generación de resúmenes de sesiones parlamentarias.....	52
5.3.1	Introducción	52
5.3.2	Preprocesado de los datos	52
5.3.3	Elección de la librería llama-cpp-python.....	53
5.3.4	Elección de modelos a estudiar.....	54
5.3.5	Experimentación y benchmarking.....	55
5.3.6	Análisis de los resultados obtenidos	56
5.3.7	Autocorrección de resúmenes con modelos GPT	61
5.3.8	Resumen del benchmarking a través de un cuadro de mando.....	62
5.3.9	Conclusión	64
5.4	Clasificación de Documentos Parlamentarios	65
5.4.1	Introducción	65
5.4.2	Selección de los datos de entrenamiento etiquetados.....	66
5.4.3	Análisis de los datos	66
5.4.4	Preprocesado de los datos	68
5.4.5	Modelos de BERT elegidos para el entrenamiento entrenamiento de BERT con datos del EuroVoc	70
5.4.6	Entrenamiento de BERT con datos de taxonomía EuroVoc	71
5.4.7	Pruebas de evaluación de los modelos	73
5.4.8	Literatura relacionada	76
5.4.9	Conclusión	77
6.	APLICACIÓN DE GESTIÓN DOCUMENTAL.....	79
6.1.	Introducción.....	79
6.2.	Diseño	79
6.2.1.	Backend	79
6.2.2.	Frontend	81
6.2.3.	Bases de datos.....	82
6.2.4.	Orquestación	82
6.2.5.	Casos de uso.....	83
6.2.6	Puntos principales de la implementación	84
7.	APORTACIONES.....	87
8.	CONCLUSIÓN	89
9.	TRABAJOS FUTUROS.....	91

10.	BIBLIOGRAFÍA.....	93
------------	--------------------------	-----------

INDICE DE FIGURAS

Figura 1: Vista de Parcan	13
Figura 2: Arquitectura global de los modelos Transformers	14
Figura 3: Arquitectura del modelo GPT	15
Figura 4: Enmascaramiento en BERT	16
Figura 5: Modelo de base de datos de MongoDB	20
Figura 6: Modelo de base de datos de ChromaDB	21
Figura 7: Infraestructura general del trabajo	21
Figura 8: Comparación del rendimiento de los modelos de mistral y llama	29
Figura 9: Flujo de entrenamiento autorregresivo de un LLM.....	36
Figura 10: Seguimiento de la pérdida del entrenamiento autorregresivo secuencial	40
Figura 11: Seguimiento de la pérdida del entrenamiento supervisado secuencial	41
Figura 12: Seguimiento de la pérdida del entrenamiento combinado	44
Figura 13: Frecuencia de notas de respuestas generadas por modelos Mistral	50
Figura 14: Frecuencia de notas de resúmenes generados por modelos Mistral y Llama2	62
Figura 15: Vista 1 del cuadro de mandos del benchmarking de resúmenes.....	64
Figura 16: Vista 2 del cuadro de mandos del benchmarking de resúmenes.....	64
Figura 17: Distribución de etiquetas del primer nivel de la taxonomía EuroVoc del dataset	67
Figura 18: Distribución del número de palabras y caracteres en el dataset del EuroVoc	68
Figura 19: Gráfica de pérdida de los 4 mejores entrenamientos de cada modelo BERT	73
Figura 21: Arquitectura del backend de la aplicación.....	80
Figura 22: Arquitectura del frontend de la aplicación.....	81
Figura 23: Flujo general de arranque de la aplicación	82
Figura 24: Vista del menú de inicio de la aplicación de gestión documental	84
Figura 25: Vista del clasificador de la aplicación de gestión documental.....	85
Figura 26: Vista del resumidor de la aplicación de gestión documental	85
Figura 27: Vista del chatbot de la aplicación de gestión documental	86

INDICE DE TABLAS

Tabla 1: Estadísticas sobre la evaluación de modelos de Mistral con memoria vectorial	49
Tabla 2: Métricas de Rouge para modelos cuantizados	58
Tabla 3: Tiempos de ejecución de generación de resúmenes	58
Tabla 4: Costes asociados a la generación de resúmenes con la API de OpenAI	59
Tabla 5: Número de tokens en resúmenes generado por modelos	60
Tabla 7: Estadísticas sobre la evaluación de resúmenes de Mistral y Llama2	61
Tabla 8: Métricas de clasificación de los 4 mejores modelos	74
Tabla 9: Porcentajes de clasificación de las 21 etiquetas del EuroVoc de los 4 mejores modelos	75

1. INTRODUCCIÓN

1.1. Presentación del ámbito de trabajo

El procesamiento del lenguaje natural (PLN) se ha consolidado como un campo crucial dentro de la inteligencia artificial, transformando la forma en que los ordenadores manejan el lenguaje humano. Hoy en día, estos desarrollos parecen "mágicos" para quienes no están familiarizados con ellos, jugando un papel vital en el desarrollo de la inteligencia artificial. Estos sistemas han sido fundamentales para lo que ahora se conoce como la generación de lenguaje por parte de los ordenadores.

Con el avance de la tecnología, se dio paso a los sistemas bayesianos, que introdujeron enfoques más probabilísticos en el PLN. Ejemplos de estos sistemas incluyen los modelos de lenguaje n-gramas y los clasificadores bayesianos ingenuos, que utilizaban estadísticas para tareas como el etiquetado de partes del discurso y la clasificación de texto [1].

Más adelante, los modelos de Bolsa de Palabras (*Bag-of-Words*) [2] simplificaron la representación del texto como una colección desordenada de palabras, facilitando tareas como la clasificación de texto y la recuperación de información. La investigación continuó avanzando y los Modelos de Vectores de Palabras (*Word Embeddings*) asignaron palabras a vectores densos en un espacio de características continuo, mejorando significativamente la calidad de los resultados en diversas tareas de PLN al capturar relaciones semánticas y sintácticas entre las palabras [3].

La introducción de las Redes Neuronales Recurrentes (RNNs, *Recurrent Neural Networks*) [4] permitió el procesamiento de secuencias de longitud variable, facilitando tareas como el etiquetado de secuencias, la generación de texto y la traducción automática de manera más precisa y eficiente. Además, las Redes Neuronales Convolucionales (CNNs, *Convolutional Neural Networks*) [5], populares en visión por computador, se aplicaron con éxito a tareas de PLN como la clasificación de texto y la extracción de características, mejorando la capacidad de análisis y comprensión del texto.

Recientemente, los modelos basados en atención, como los *Transformers*, han revolucionado el PLN al permitir la captura de relaciones de largo alcance en el texto. La implementación de estos modelos ha impulsado avances significativos en aplicaciones como la traducción automática y el resumen de texto. Un hito crucial fue el artículo "*Attention is All You Need*" de Vaswani et al. [6], que introdujo la arquitectura *Transformer*, ahora estándar en el procesamiento del lenguaje natural. Estos sistemas, aunque eficaces en ciertos aspectos, estaban limitados en su capacidad para adaptarse a la complejidad del lenguaje humano y a las variadas intenciones en tiempo real. El surgimiento de los chatbots [7] representó un salto cualitativo en esta evolución. Con el desarrollo de modelos de lenguaje cada vez más avanzados, como GPT (*Generative Pre-trained Transformer*) [8] desarrollado por OpenAI [9], y otros sistemas de inteligencia artificial, los chatbots han alcanzado un nivel de sofisticación que les permite comprender y generar lenguaje humano con una precisión sorprendentes.

Este auge ha transformado la interacción con la tecnología en diversos campos, desde el servicio al cliente hasta la educación y la atención médica, alcanzando incluso los ámbitos legislativos nacionales. Este avance tecnológico, caracterizado por su capacidad para automatizar tareas rutinarias, ofrecer información instantánea y proporcionar experiencias de usuario personalizadas, ha tenido un impacto profundo en numerosos sectores.

En este contexto de avance tecnológico, los sectores legislativo, económico, educativo, entre otros, enfrentan desafíos únicos en la gestión de documentos. La cantidad de documentación generada diariamente es inmensa, incluyendo propuestas de ley, apuntes, informes y exámenes. La gestión

eficiente de esta información es crucial para garantizar la transparencia, la accesibilidad y la toma de decisiones informadas. Los sistemas tradicionales de gestión documental a menudo resultan insuficientes para manejar este volumen de datos de manera eficiente. Estos campos y sus equipos necesitan herramientas avanzadas que les permitan clasificar, buscar y resumir documentos rápidamente, así como interactuar con sistemas automatizados que puedan proporcionar respuestas precisas y contextuales sobre los contenidos legislativos.

Una forma de abordar estos desafíos es aprovechar el potencial de modelos avanzados como GPT de OpenAI. Con una amplia variedad de modelos disponibles, desde los más asequibles como GPT-3.5 [10] hasta los más complejos y actualizados como GPT-4 Omni [11], OpenAI ofrece una solución versátil y poderosa. Entre los puntos fuertes de OpenAI se encuentran su facilidad de implementación, la relativa simplicidad de uso sin necesidad de conocimientos profundos en inteligencia artificial y la rapidez de la API para integrarla en sistemas existentes.

En este entorno de continua actualización, el uso de la inteligencia artificial y chatbots ha ganado gran relevancia. Empresas como Meta, que cada mes introduce nuevas actualizaciones de su gran modelo de lenguaje Llama, con su última versión lanzada el 18 de abril de 2024, Llama 3 [12], han sido rápidas en adoptar estos nuevos modelos.

Sin embargo, la situación en el sector público es muy diferente. La implementación de políticas y servicios de terceros, como las APIs de Mistral y OpenAI, que manejan datos, puede entrar en conflicto con las regulaciones y principios de protección de datos establecidos a nivel nacional y europeo. Un análisis superficial de las leyes nacionales de protección de datos revela varios aspectos cruciales:

- **Artículos 40-43 de la LOPDGDD (Ley Orgánica de Protección de Datos Personales y garantía de los derechos digitales):**
 - **Artículo 40:** Las transferencias internacionales de datos están sujetas al Reglamento (UE) 2016/679 (Reglamento General de Protección de Datos), la LOPDGDD y las normas de desarrollo aprobadas por el Gobierno [13].
 - **Artículo 41:** Las autoridades de protección de datos, como la Agencia Española de Protección de Datos y las autoridades autonómicas, tienen competencias para adoptar medidas como la aprobación de cláusulas contractuales tipo [14].
 - **Artículo 42:** Establece la necesidad de autorización previa para ciertas transferencias internacionales [15].
 - **Artículo 43:** Obliga a informar a las autoridades competentes sobre las transferencias internacionales realizadas en base a intereses legítimos imperiosos [16].

El uso de servicios externos como OpenAI por parte de la administración pública española se complica debido a estas regulaciones. OpenAI, al ser una empresa con sede en Estados Unidos, implica la transferencia de datos fuera de la Unión Europea. Esto requiere que la administración pública española cumpla con las disposiciones legales y obtenga la autorización correspondiente de las autoridades de protección de datos. Además, cualquier transferencia internacional de datos basada en intereses legítimos imperiosos debe ser informada a las autoridades competentes.

- **Artículo 48 del RGPD (Reglamento General de Protección de Datos):** Este artículo aborda las transferencias o comunicaciones no autorizadas por el Derecho de la Unión. Establece que cualquier sentencia de un órgano jurisdiccional o decisión de una autoridad administrativa de un tercer país que exija la transferencia o comunicación de datos personales solo será

reconocida o ejecutable si se basa en un acuerdo internacional vigente entre el país tercero requirente y la Unión Europea o un Estado miembro [17].

Aunque existen mecanismos legales para legitimar estas transferencias, su interpretación y aplicación pueden generar incertidumbre y preocupaciones sobre la legalidad y protección de los datos personales transferidos. Esto requiere un análisis detallado y una evaluación de los riesgos asociados con la transferencia de datos a terceros países como Estados Unidos.

Existen numerosas cuestiones legales que hacen que opciones tan asequibles como los chatbots o modelos basados en APIs sean una solución cuestionable para el ámbito público. Debido a esto, surge una alternativa prometedora nacida del “boom” de los modelos generativos: los modelos del lenguaje locales. En este contexto, uno de los repositorios más destacados es Hugging Face [18], una plataforma que proporciona una amplia gama de modelos de lenguaje que pueden ser implementados localmente. Hugging Face ofrece herramientas y bibliotecas que permiten a los desarrolladores entrenar y desplegar modelos de inteligencia artificial en sus propios servidores, sin necesidad de transferir datos sensibles a terceros países. Esta plataforma ha destacado por proporcionar un tratamiento técnico y cercano de los modelos, abarcando desde análisis de imagen y video hasta complejos modelos de lenguaje. Además, cuenta con una amplia comunidad de respaldo que contribuye a su evolución y mejora, asegurando que los usuarios tengan acceso a las últimas innovaciones y desarrollos.

Esta opción resulta particularmente atractiva en el ámbito público por varias razones. En primer lugar, los modelos locales cubren aspectos legales que las APIs externas no pueden garantizar. Al mantener los datos dentro de la infraestructura controlada por la entidad pública, se asegura el cumplimiento de las normativas de protección de datos, como el RGPD y la LOPDGDD, evitando las complejidades y riesgos asociados con las transferencias internacionales de datos. Además, los modelos locales permiten un mayor control y adaptación. En manos de expertos, estos modelos pueden ser finamente ajustados y optimizados para satisfacer necesidades específicas, mejorando su rendimiento y precisión en tareas concretas. Esto incluye la capacidad de adaptar el modelo a los datos y contextos específicos de la administración pública, incrementando así la relevancia y efectividad de las soluciones implementadas.

A pesar de las ventajas significativas que ofrecen estas plataformas, el desarrollo, implementación y mantenimiento de modelos locales presentan desafíos notables. Estos modelos requieren un equipo y personal altamente especializado en la materia. Con el auge del campo del análisis de datos, contratar profesionales con las habilidades necesarias se ha vuelto particularmente complicado, dado que estos perfiles son altamente demandados y existe una intensa competencia en el mercado laboral. Además, se requiere una inversión considerable en infraestructura física. Para garantizar una disponibilidad rápida y un rendimiento aceptable, es esencial contar con hardware de alta gama, incluyendo GPUs (*Graphic Processing Unit*) con gran cantidad de VRAM (*Video Random Access Memory*), entre otros componentes avanzados, que son fundamentales para manejar las complejas tareas de procesamiento y entrenamiento de modelos. Estas inversiones en infraestructura pueden ser prohibitivamente costosas.

En este complejo panorama del sector público, se ha desarrollado este trabajo de fin de grado, centralizado en aportar un primer enfoque a los sistemas de gestión de documentación. Se ha realizado un estudio del manejo de documentación parlamentaria, utilizando como caso de estudio los diarios de sesión del Parlamento de Canarias. Estos diarios provienen de una fuente primaria en formato *verbatim*, que se caracteriza por ser una transcripción textual exacta de los discursos y debates

parlamentarios. Este formato garantiza la transparencia política y permite a usuarios acceder a la información de manera íntegra y sin modificaciones. Actualmente, estos documentos se pueden consultar a través de la página oficial del Gobierno de Canarias [19].

La rápida búsqueda de fragmentos, leyes, debates y otros elementos es crucial para sectores como el periodismo, donde la rapidez en la publicación es fundamental. La extensa longitud de estos ficheros subraya la necesidad de un sistema de gestión eficiente. Ante esta situación, se proponen diferentes vías que podrían proporcionar a los usuarios una gestión y consulta sencilla de estos documentos.

Una de las soluciones destacadas son los resumidores, modelos de lenguaje encargados de tomar un texto extenso y reducir su contenido a una versión más manejable, manteniendo la importancia y relevancia de la información. Con esta funcionalidad, ficheros de hasta 20 o incluso 30 páginas podrían ser resumidos en unas pocas páginas o, incluso, párrafos.

Paralelamente, está la clasificación. En un debate parlamentario se pueden abordar múltiples temáticas definidas por el EuroVoc, un tesoro multilingüe y multidisciplinario de la Unión Europea que cubre 21 niveles de clasificación temática, los cuales a su vez tienen diferentes subniveles [20]. La capacidad de buscar o, al menos, de asignar temáticas de forma sencilla a fragmentos de texto es una funcionalidad altamente necesaria para los equipos que manejan esta documentación.

Finalmente, la mayor facilidad que se podría aportar a un sistema de gestión documental es un asistente, al cual se le podría hacer cualquier pregunta sobre los ficheros correspondientes. Fechas, políticos, leyes y debates concretos podrían ser ubicados en cuestión de minutos, incluso segundos.

Tomando estos tres frentes, se acomete una tarea de investigación sobre la aproximación de estas funcionalidades en el ámbito público, evaluando tanto los aspectos legales como los económicos. Este estudio pretende ofrecer soluciones viables y eficientes dentro de un marco de recursos reducidos, para mejorar la gestión documental en el Parlamento de Canarias y potencialmente en otras situaciones parlamentarias. En un entorno de bajos recursos, la investigación se centra en adaptar herramientas y tecnologías accesibles, maximizando el uso de recursos disponibles sin comprometer la calidad ni la eficiencia. Se busca demostrar cómo, a través de la optimización de modelos de lenguaje, es posible implementar sistemas avanzados de gestión documental que sean asequibles y efectivos, proporcionando así un valor significativo en entornos con limitaciones presupuestarias.

1.2. Objetivos

En el contexto de una investigación orientada a mejorar la gestión documental en entornos jurídicos, el objetivo principal del proyecto es explorar y aplicar herramientas avanzadas de procesamiento del lenguaje natural para optimizar el manejo de grandes volúmenes de datos textuales. Este proyecto tiene como meta desarrollar soluciones innovadoras que aborden los desafíos específicos del procesamiento de documentos legislativos en el Parlamento de Canarias, al mismo tiempo que contribuyan al avance general en el campo del PLN.

Uno de los objetivos centrales es diseñar y poner en práctica una arquitectura de software eficiente para la extracción, preprocesamiento y análisis de documentos legislativos. Este enfoque implica la creación de un *pipeline* automatizado que permita la actualización constante de una base de datos documental, asegurando la calidad y coherencia de la información. La solución propuesta busca

optimizar la recuperación y el análisis de datos, facilitando el acceso a la información de manera más eficiente y efectiva en un contexto normativo en constante evolución.

El proyecto también tiene como objetivo adquirir y aplicar conocimientos sobre las tecnologías más recientes en el mercado laboral, enfocándose en el *fine-tuning* de modelos de lenguaje grande. Se prestará especial atención al uso del modelo Mistral para ajustar y perfeccionar el rendimiento en diversas tareas, evaluando distintas metodologías de entrenamiento para mejorar la capacidad de respuesta del modelo en escenarios específicos.

Además, el estudio y análisis de diferentes técnicas de PLN será un aspecto crucial del proyecto, con el fin de llevar a cabo una evaluación detallada del comportamiento de diversos modelos, probar diferentes métodos de entrenamiento y evaluar su eficacia. Este análisis no solo busca resolver problemas concretos del Parlamento de Canarias, sino también contribuir al conocimiento científico en el campo del PLN, proporcionando perspectivas valiosas sobre las mejores prácticas y metodologías para la gestión documental.

Finalmente, uno de los objetivos clave es demostrar que es posible diseñar aplicaciones eficientes para la gestión y análisis de documentos legislativos a pesar de las limitaciones de recursos. El proyecto se enfoca en desarrollar soluciones viables y prácticas para la gestión documental en contextos con recursos limitados, reflejando un enfoque de investigación que busca maximizar el impacto de las herramientas desarrolladas mediante un uso eficaz de los recursos disponibles.

En conjunto, estos objetivos no solo abordan de manera integral las necesidades del ámbito jurídico, sino que también establecen un marco de investigación para el desarrollo de soluciones innovadoras y prácticas en la gestión de información textual. Este enfoque demuestra que es posible avanzar en el campo del PLN y resolver problemas específicos de la gestión documental incluso en contextos de bajos recursos.

1.3. Recursos utilizados de hardware y software

En el entorno físico del proyecto, se dispone de un ordenador de escritorio con un procesador 13th Gen Intel(R) Core(TM) i7-13700K, operando a una velocidad de 3.40 GHz. Este sistema cuenta con una memoria RAM de 64 GB, de los cuales 24,7 GB son utilizables, proporcionando una capacidad significativa para manejar cargas de trabajo intensivas en datos y procesamiento. El sistema operativo en uso es Windows 11 Pro en su versión 23H2. Además, se incluye una tarjeta gráfica NVIDIA GeForce RTX 4090, que acelera operaciones computacionales intensivas, como el entrenamiento de modelos de aprendizaje automático y la manipulación de gráficos complejos.

La obtención y procesamiento de datos, así como su almacenamiento en una base de datos no relacional, se basa en Java como lenguaje de programación. En particular, se emplea Java 17 para aprovechar sus características avanzadas y su estabilidad en el desarrollo de aplicaciones robustas. El entorno de desarrollo integrado (IDE) elegido es IntelliJ IDEA [21], que facilita la gestión del código y la integración de librerías necesarias para el proyecto. Además, se utiliza Maven [22] para la administración de dependencias y el proceso de construcción del proyecto. Maven, ampliamente utilizado en el desarrollo de software en Java, gestiona las dependencias del proyecto, construye el código, ejecuta pruebas y genera documentación. Su capacidad para ser extensible mediante *plugins*

permite adaptar el proceso de construcción a las necesidades específicas del proyecto, añadiendo funcionalidades como la compilación de código y la ejecución de pruebas.

El desarrollo de técnicas de procesamiento del lenguaje natural (PLN), el entrenamiento de modelos de lenguaje y la creación de una primera aproximación a una aplicación web, se basa en Python como lenguaje de programación. Python es reconocido por su versatilidad en la implementación de técnicas de PLN y la creación de prototipos. En este contexto, se usa Python 3.10 [23] y el entorno de desarrollo integrado (IDE) utilizado es PyCharm de JetBrains [24], que proporciona herramientas avanzadas para la gestión del código y el desarrollo de aplicaciones. Entre las principales librerías software usadas en este proyecto se incluyen:

- **openai** [25]: Integración de modelos de lenguaje avanzado GPT (*Generative Pre-trained Transformer*), facilitando el tratamiento y generación de texto de alta calidad.
- **pymongo** [26]: Interacción con bases de datos MongoDB [27], permitiendo almacenar y gestionar documentos con metadata relacionada de manera eficiente. MongoDB es una base de datos NoSQL orientada a documentos, que almacena datos en formato JSON (*JavaScript Object Notation*) flexible.
- **transformers** [28]: Proporciona acceso a una amplia variedad de modelos de lenguaje preentrenados, esenciales para tareas de PLN como clasificación, generación de texto y análisis de sentimientos. También proporciona un amplio marco de herramientas de *fine-tuning* para adaptar estos modelos a tareas específicas, teniendo un control más cercano de dichos procesos.
- **llama-cpp-python** [29]: Herramienta para trabajar con modelos cuantizados en formato GGUF (*GPT-Generated Unified Format*), optimizando el rendimiento y la eficiencia de los modelos del lenguaje en entornos con recursos limitados.
- **flask** [30]: Microframework para Python que permite el desarrollo de aplicaciones web ligeras, utilizado en este proyecto para crear APIs y servicios web.
- **requests** [31]: Biblioteca de Python para realizar solicitudes HTTP (*Hypertext Transfer Protocol*) de manera simple y eficiente, facilitando la comunicación con servicios web externos y APIs.
- **streamlit** [32]: Herramienta para el desarrollo de aplicaciones web interactivas y visualizaciones de datos, permitiendo la creación de interfaces gráficas intuitivas para los usuarios finales.
- **torch** [33]: Proporciona una variedad de herramientas para el desarrollo y entrenamiento de modelos de IA, esencial para tareas de aprendizaje profundo en el proyecto.
- **trl** (*Transformer Reinforcement Learning*) [34]: Utilizada para aplicar técnicas de aprendizaje por refuerzo a modelos de lenguaje basados en Transformers, mejorando su rendimiento y adaptabilidad.
- **peft** (*Parameter-Efficient Fine-Tuning*) [35]: Biblioteca que facilita el ajuste fino de modelos de lenguaje con una cantidad mínima de parámetros adicionales, optimizando tanto la eficiencia como la efectividad del entrenamiento de modelos grandes.
- **Flask-SocketIO** [36]: Extensión para el microframework Flask de Python que facilita la integración de *WebSockets* en la aplicación web. Permite la comunicación en tiempo real entre el servidor y los clientes, habilitando funcionalidades como chat en vivo, notificaciones instantáneas y actualizaciones en tiempo real.

- **Streamlit:** Desarrollo de interfaces de usuario amigable. Streamlit es conocida por su facilidad de uso y su documentación extensiva, lo que permite desarrollar aplicaciones web rápidas y efectivas para proyectos de ciencia de datos y aprendizaje automático.

En la gestión de la memoria del asistente chatbot, se utiliza ChromaDB [37], implementada en Python como base de datos vectorial. Esta herramienta almacena y recupera información de manera eficiente mediante la representación de datos en vectores. En las consultas, ChromaDB busca los vectores más similares a la consulta realizada e incorpora la información relevante en la ventana de contexto presentada al modelo del lenguaje. Este enfoque mejora significativamente la relevancia y precisión de las respuestas del asistente, ya que el modelo accede a contextos y datos pertinentes, esenciales para generar respuestas coherentes y útiles.

Garantizar una infraestructura robusta y eficiente requiere el uso de Docker [38] a lo largo de todo el desarrollo. Docker permite mantener diferentes componentes del sistema, como bases de datos, tareas de procesamiento y aplicaciones web, separados en contenedores individuales. La separación de dependencias asegura que cada servicio funcione de manera aislada, eliminando posibles conflictos entre librerías y versiones de software. Esto mejora la estabilidad del sistema y facilita la identificación y resolución de problemas específicos en cada componente sin afectar al resto del entorno. Además, Docker simplifica el proceso de despliegue, permitiendo replicar el entorno de desarrollo en producción de manera precisa y eficiente. Esta portabilidad y consistencia son cruciales para garantizar un rendimiento óptimo y una rápida implementación de actualizaciones o cambios en la infraestructura del proyecto.

Además, para el entrenamiento de modelos generativos, se ha utilizado Jupyter Notebook [39]. Esta herramienta es particularmente adecuada para tareas de entrenamiento debido a su capacidad para proporcionar un entorno interactivo y flexible. Jupyter Notebook permite visualizar los resultados y ajustar parámetros en tiempo real, facilitando una experimentación ágil y eficiente en comparación con otros entornos de desarrollo. Su facilidad para integrar código, visualizaciones y notas en un solo documento también mejora significativamente la claridad y el seguimiento del proceso de entrenamiento.

El uso de la librería *Weights & Biases* (wandb) [40] como herramienta principal permite el seguimiento detallado de los entrenamientos. WandB ofrece una plataforma potente para monitorear métricas de rendimiento, comparar experimentos y visualizar el progreso en tiempo real. Esta capacidad resulta invaluable para optimizar y mejorar iterativamente los modelos generativos. El seguimiento detallado que ofrece WandB asegura que cualquier ajuste en los parámetros o cambios en el modelo se documente adecuadamente, facilitando la reproducibilidad y la colaboración en proyectos complejos de inteligencia artificial. Este nivel de control y visibilidad es esencial para garantizar desarrollos precisos y eficientes, permitiendo a los investigadores y desarrolladores trabajar de manera más efectiva y coordinada.

Finalmente, la presentación de datos y los resultados del benchmarking en la generación de resúmenes se lleva a cabo mediante Power BI [41]. Esta herramienta de visualización de datos de Microsoft es esencial para crear informes interactivos y paneles de control dinámicos, permitiendo una exposición clara y efectiva de los resultados del proyecto.

1.4. Competencias

A continuación, se detallan las competencias desarrolladas y aplicadas en el proyecto, enfocadas en habilidades técnicas y conceptuales necesarias para el diseño, desarrollo e implementación de sistemas informáticos orientados a la inteligencia artificial y el procesamiento de datos.

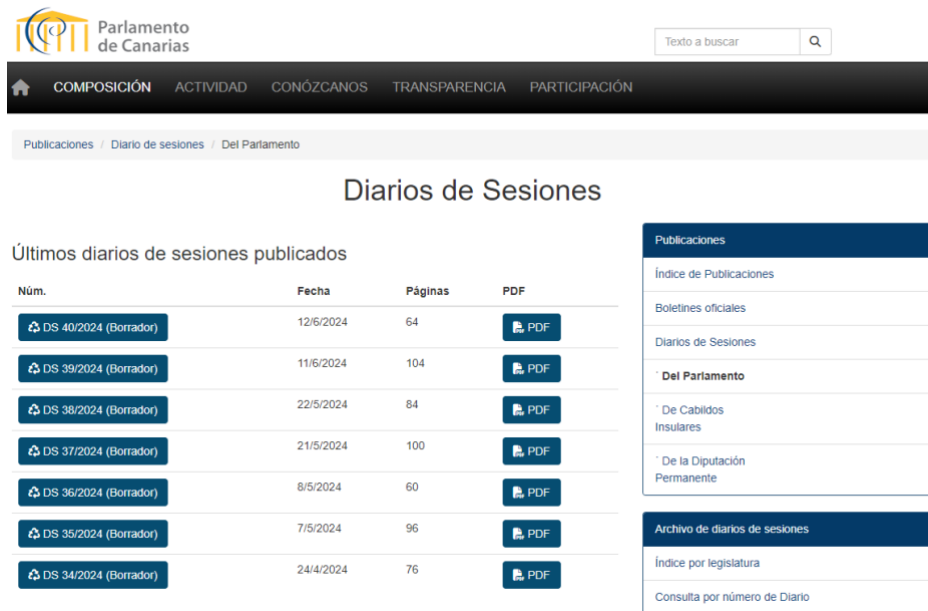
1. **Competencia ED1:** La competencia ED1 se evidencia en este trabajo mediante el diseño y construcción de un sistema de procesamiento de lenguaje natural para documentos legislativos. El proyecto demuestra un profundo conocimiento de técnicas de procesamiento de lenguaje natural (NLP) y la aplicación de algoritmos avanzados para automatizar el análisis y generación de contenido. Se han desarrollado aplicaciones específicas, como un generador de resúmenes de documentos legislativos y un chatbot especializado en el ámbito parlamentario, que muestran la capacidad para diseñar sistemas inteligentes que resuelven problemas complejos en el procesamiento de datos legislativos.
2. **Competencia ED2:** La Competencia ED2 en este proyecto se centra en la capacidad para adquirir, formalizar y representar el conocimiento humano de manera computable para resolver problemas mediante sistemas informáticos. En el contexto del etiquetado de documentos y la generación de resúmenes, esta competencia es crucial para convertir el conocimiento presente en textos legislativos en formatos procesables por máquinas, facilitando el análisis y la recuperación de información en entornos inteligentes como el Parlamento de Canarias.
3. **Competencia ED3:** La Competencia ED3 implica conocer y desarrollar técnicas de aprendizaje automático para diseñar aplicaciones que extraigan información y conocimiento de grandes volúmenes de datos. En este proyecto, se refleja en la aplicación de estas técnicas para la clasificación de documentos y la creación de asistentes especializados, facilitando la extracción de información, identificación de patrones y optimización de procesos en la gestión de datos legislativos.
4. **Competencia ED4:** La Competencia ED4 se manifiesta en el diseño, desarrollo e implementación de soluciones de software para preprocesamiento de datos y generación de resúmenes en el ámbito de la Inteligencia Artificial. Este proceso incluye la identificación de problemas, la creación de algoritmos de clasificación, y la verificación y documentación de soluciones, asegurando su calidad y robustez para futuros desarrollos en el análisis de documentos legislativos.
5. **Competencia ED6:** La Competencia ED6 se manifiesta en el conocimiento profundo de los principios y modelos de Ciencia de Datos para el análisis y predicción de grandes volúmenes de datos. Este conocimiento permite utilizar técnicas avanzadas para procesar información legislativa, anticipar tendencias y adaptar soluciones a las necesidades del entorno jurídico, facilitando una toma de decisiones informada y estratégica en el ámbito parlamentario.

2. ANÁLISIS

2.1. Estado del arte

Los sistemas de gestión documental utilizados en la administración pública están diseñados para facilitar el acceso y la recuperación de documentos oficiales de manera eficiente y organizada. Un ejemplo prominente de estos sistemas es Parcan [42], que ofrece servicios en línea para acceder a la información legislativa y gubernamental de la región de Canarias.

La página de Parcan presenta una interfaz amigable para usuarios del sector o interesados en la temática, con opciones de búsqueda por índice de legislatura y número de diario. Además, proporciona acceso rápido al último diario publicado y a diarios anteriores, lo que permite una consulta ágil tanto de la información más reciente como del archivo histórico, como se podrá comprobar en la Figura 1. Esta funcionalidad es crucial para garantizar que los ciudadanos y profesionales puedan acceder fácilmente a la documentación relevante y mantenerse informados sobre las novedades legislativas y administrativas de la región.



Parlamento de Canarias
 Texto a buscar

COMPOSICIÓN ACTIVIDAD CONÓZCANOS TRANSPARENCIA PARTICIPACIÓN

Publicaciones / Diario de sesiones / Del Parlamento

Diarios de Sesiones

Últimos diarios de sesiones publicados

Núm.	Fecha	Páginas	PDF
DS 40/2024 (Borrador)	12/6/2024	64	
DS 39/2024 (Borrador)	11/6/2024	104	
DS 38/2024 (Borrador)	22/5/2024	84	
DS 37/2024 (Borrador)	21/5/2024	100	
DS 36/2024 (Borrador)	8/5/2024	60	
DS 35/2024 (Borrador)	7/5/2024	96	
DS 34/2024 (Borrador)	24/4/2024	76	

Publicaciones
 Índice de Publicaciones
 Boletines oficiales
 Diarios de Sesiones
Del Parlamento
 De Cabildos Insulares
 De la Diputación Permanente

Archivo de diarios de sesiones
 Índice por legislatura
 Consulta por número de Diario

Figura 1: Vista de Parcan

A pesar de las funcionalidades avanzadas, los sistemas de gestión documental actuales, como el mostrado anteriormente, presentan ciertas limitaciones en sus capacidades de búsqueda. Aunque el buscador de los diarios de sesiones permite realizar búsquedas sencillas por índice de legislatura y número de diario, carece de la capacidad de realizar búsquedas granulares, como la obtención de fragmentos concretos de una ley o debates. Esta carencia puede dificultar la recuperación de información precisa, ya que los resultados de las búsquedas son documentos en formato PDF que requieren una revisión manual para encontrar la información relevante.

Estas limitaciones han impulsado el desarrollo y uso de técnicas de procesamiento del lenguaje natural, como la clasificación textual, la capacidad de resumir texto y la generación de nuevos corpus. Sin embargo, para comprender las técnicas actuales, es necesario revisar los fundamentos históricos que han llevado a estos avances.

Los modelos *transformers* representan una clase avanzada de arquitecturas de aprendizaje profundo que han revolucionado el campo del PLN y otras áreas relacionadas. Presentados en el artículo "*Attention Is All You Need*" [6] por Vaswani et al. en 2017, los *transformers* se distinguen por su capacidad para manejar secuencias de datos, como texto, de manera extremadamente eficaz y eficiente. La clave de su éxito reside en su principal fundamento, los mecanismos de atención, especialmente la atención de múltiples cabezas. En términos simples, la arquitectura de los Transformers consta de un conjunto de capas que trabajan en conjunto, como se muestra en la Figura 2. El elemento principal es el mecanismo de atención, que permite evaluar la importancia de diferentes partes de la entrada. Esto significa que estos modelos pueden enfocar más o menos atención en ciertas palabras al procesar un texto [43].

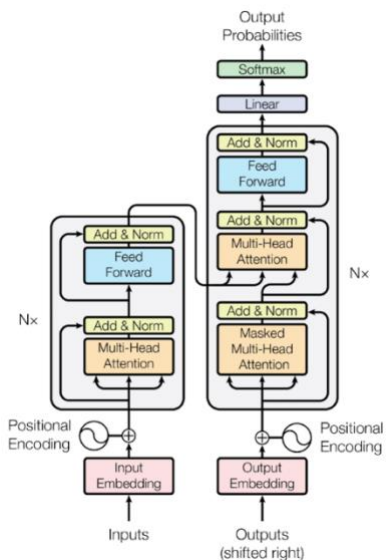


Figura 2: Arquitectura global de los modelos Transformers

El mecanismo de atención se ilustra en la Figura 2, específicamente en las capas de atención multicabezada (*Multi-Head Attention*). Estas capas permiten prestar especial atención a diferentes partes de la secuencia de entrada, mejorando la capacidad del modelo para aprender relaciones complejas y captar diferentes matices de un texto.

Otro aspecto destacado de la arquitectura *transformer* es su estructura en bloques de codificación y decodificación. Por un lado, el codificador (bloque izquierdo de la Figura 2) procesa la entrada y la convierte en una representación interna rica en contexto. Este componente está compuesto por dos subcapas principales: una capa de atención (atención multicabezada) y una red neuronal *feed-forward*. Por otro lado, se encuentra el decodificador (bloque derecho de la Figura 2), que está compuesto por una serie de capas idénticas a las del codificador, pero con ciertas diferencias clave para adaptarse a la tarea de generar una secuencia de salida.

La principal diferencia en el decodificador radica en las subcapas de atención, que se dividen en dos tipos diferenciados:

1. **Capa de atención con enmascaramiento:** Esta subcapa de atención se aplica únicamente a la secuencia de salida generada hasta el momento. Asegura que el modelo solo tenga en cuenta las palabras generadas anteriormente y no las futuras, lo cual es esencial para tareas como la generación de texto, donde cada palabra se genera secuencialmente.
2. **Capa de atención a la entrada codificada:** Esta subcapa se enfoca en la secuencia de salida en relación con la representación interna de la secuencia de entrada producida por el codificador. Permite que cada token de salida "preste especial atención" a todos los tokens de la entrada, integrando la información contextualmente relevante.

Estos modelos Transformers han revolucionado el campo de la tecnología del procesamiento del lenguaje natural, y su comprensión es esencial para situarse adecuadamente en el contexto de modelos avanzados.

Uno de los modelos que más se ha beneficiado de los avances en la arquitectura *transformer* es GPT (*Generative Pre-trained Transformer*). Este modelo se ha convertido en uno de los más utilizados y, gracias al gran soporte proporcionado por OpenAI, se ha transformado en una herramienta indispensable para numerosas empresas. OpenAI ofrece una documentación extensiva, tutoriales y una gran comunidad de desarrolladores, facilitando el acceso a GPT a través de su API. Esto permite a

miles de interesados integrar fácilmente las capacidades del modelo en sus propios productos y servicios. Además, OpenAI sigue innovando y mejorando GPT, lanzando versiones más avanzadas que presentan mejoras significativas en términos de rendimiento y capacidad. Esta evolución continua asegura que GPT se mantenga a la vanguardia de la tecnología de procesamiento del lenguaje natural, ofreciendo siempre soluciones de última generación a sus usuarios. El modelo es capaz de manejar volúmenes masivos de solicitudes y ofrecer tiempos de respuesta rápidos, lo cual es crucial para aplicaciones comerciales.

La arquitectura de GPT se basa en el uso de un decodificador *transformer*, pero sin la parte de atención del codificador. El modelo se entrena mediante la predicción de la siguiente palabra en una secuencia, un método de entrenamiento no supervisado donde se predice la palabra siguiente en una oración dada. Las secuencias de entrada de estos modelos varían dependiendo de la versión.

GPT no puede "comprender" palabras de manera directa. El primer paso consiste en mantener un vocabulario de todas las palabras, con una asignación de valores. Si una palabra no está presente en el diccionario del modelo, se utiliza un método de tokenización de subpalabras denominado *byte pair encoding* (BPE) [44]. Este algoritmo de subpalabras crea codificaciones posicionales que convierten las subpalabras en vectores. Este primer paso se ilustra en la Figura 3, donde la entrada tokenizada se introduce en el bloque decodificador.

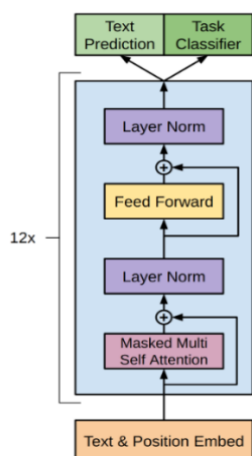


Figura 3: Arquitectura del modelo GPT

Una vez realizada la codificación, se calculan los vectores clave (*key*), consulta (*query*) y valor (*value*) para cada token de la secuencia, utilizando la multiplicación de matrices para producir una matriz de puntuaciones. Esta matriz de puntuaciones determina la importancia que se debe atribuir a una palabra en relación con otras palabras de la secuencia. Cuanto mayor sea la puntuación, mayor será la atención asignada. Este proceso ocurre en la capa de atención multicabezada con enmascaramiento (capa rosa en la Figura 3).

A continuación, las puntuaciones se reducen dividiéndolas por la raíz cuadrada de la dimensión de las consultas y las claves, lo que permite obtener gradientes más estables. Finalmente, se aplica una capa *SoftMax* a la puntuación escalada para obtener los pesos de atención, que proporcionan valores de probabilidad entre 0 y 1, permitiendo que el modelo sea más preciso sobre a qué palabras debe prestar atención [43].

Otro punto clave son las conexiones residuales, que permiten un flujo de gradientes más efectivo y facilitan el entrenamiento de redes profundas. Posteriormente, se utiliza la normalización por capas (*Layer Normalization*) para estabilizar el entrenamiento y mejorar la convergencia. La normalización por capas se aplica después de la adición residual, asegurando que las salidas tengan una media y varianza constantes.

En el estado actual de la tecnología, una de las herramientas más demandadas son los asistentes personalizados o chatbots, que en su mayoría se fundamentan en el uso de modelos como GPT. Existen otras técnicas que, combinadas con el uso de estos asistentes, están marcando una clara línea de desarrollo. Una de estas técnicas es el uso de memorias vectoriales, que permiten a los chatbots recordar información relevante sobre una conversación anterior o de una base de conocimiento, mejorando así su capacidad para mantener conversaciones coherentes y proporcionar respuestas más precisas [45].

Una de las librerías pioneras en esta técnica es LangChain [46], que utiliza bases de datos vectoriales para implementar la memoria en diferentes modelos de lenguaje. Este enfoque permite a los modelos recordar y utilizar información histórica de manera efectiva, mejorando significativamente la interacción con los usuarios.

Cabe recalcar que los modelos de lenguaje basados en Transformers no solo se emplean para generar texto, sino también para tareas de clasificación textual. Uno de los modelos más destacados en este contexto es BERT (*Bidirectional Encoder Representations from Transformers*), que ha demostrado un rendimiento sobresaliente en una variedad de tareas.

La arquitectura de BERT se basa en el codificador de los Transformers, a diferencia de GPT, que utiliza el decodificador. BERT emplea una estructura de atención bidireccional, lo que permite al modelo capturar el contexto de las palabras en ambas direcciones dentro de una oración. Su arquitectura consta de múltiples capas de codificadores, cada una de las cuales procesa la entrada de texto de forma independiente. Cada capa de codificador incluye atención multicabezada (tanto con enmascaramiento como autoatención) y una capa *feed-forward*, lo que permite al modelo capturar relaciones complejas entre las palabras del texto [47].

Durante el entrenamiento, BERT utiliza una técnica conocida como *masking* [47], donde ciertas palabras en la entrada se ocultan aleatoriamente, y el modelo debe predecirlas en función del contexto proporcionado por las palabras circundantes. Esto le permite capturar una representación más profunda y contextualizada del texto, como se muestra en la Figura 4.

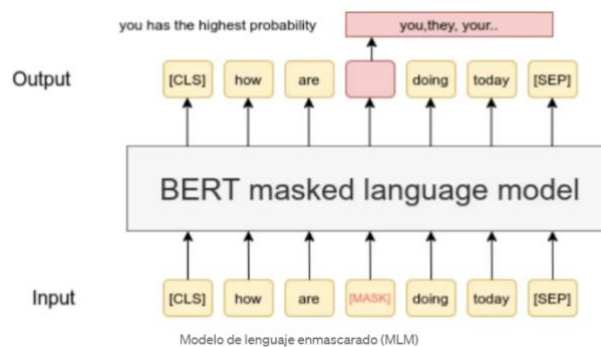


Figura 4: Enmascaramiento en BERT

Observando el gran potencial que ofrecen estas herramientas, se ha generado una creciente competencia para OpenAI. Varios modelos de lenguaje avanzado están emergiendo, presentando innovaciones significativas en rendimiento y funcionalidad. Esta competencia impulsa la evolución de la tecnología, ofreciendo a los usuarios una gama cada vez mayor de opciones para abordar tareas de procesamiento del lenguaje natural de manera más eficaz y eficiente.

Mistral [48] es otro modelo de lenguaje avanzado diseñado específicamente para optimizar el rendimiento y la eficiencia. Este modelo incorpora innovaciones como la atención agrupada de consultas (*grouped-query attention* o GQA), que permite una inferencia más rápida, y la atención de ventana deslizante (*sliding window attention* o SWA), que maneja secuencias largas con un menor coste computacional [49]. Además, Mistral emplea una caché de búfer continuo para limitar el uso de memoria en la atención, y utiliza técnicas de prellenado y fragmentación que optimizan la generación de secuencias. Mistral se presenta en varias versiones, incluyendo 7B, 8x7B y 8x22B, y su modelo más

reciente, Codestral 22B [50], está enfocado en tareas de generación de código y abarca más de 80 lenguajes de programación.

Por otro lado, Llama, un modelo impulsado por Meta [51], es un modelo multilingüe con capacidad de aprendizaje continuo. Llama ha sido entrenado con un conjunto de datos masivo de 15 billones de tokens, incluyendo una mayor cantidad de código y lenguas extranjeras en comparación con su predecesor, Llama 2. Llama se beneficia de un ajuste fino basado en instrucciones, lo que mejora significativamente su rendimiento en tareas específicas [52]. Para desbloquear todo el potencial de Llama 3 en aplicaciones de chat y diálogo, Meta ha introducido un nuevo enfoque alineado con el concepto de ajuste fino por instrucciones. Este método combina ajuste fino supervisado (*supervised fine-tuning* o SFT), muestreo de rechazo, optimización de políticas proximales (*proximal policy optimization* o PPO) y optimización de preferencias directas (*direct preference optimization* o DPO).

Llama 3 también adopta un enfoque de seguridad multinivel para mitigar sesgos y garantizar un uso responsable de la inteligencia artificial. Esta última versión refuerza estas características con una expansión de capacidades multilingües y una optimización del rendimiento, consolidándose como una herramienta avanzada para el procesamiento del lenguaje natural en diversos idiomas y contextos [52].

En el podio de los modelos de lenguaje más conocidos, otro competidor destacado es Géminis, anteriormente conocido como Bard. Este modelo está diseñado para facilitar diálogos fluidos y coherentes, y su arquitectura se basa en decodificadores *transformer* que permiten una gran escalabilidad y una inferencia optimizada. Con un contexto de longitud de 32k tokens y mecanismos de atención eficientes, Géminis ofrece un rendimiento sobresaliente en tareas de diálogo [53]. Además, este modelo es multimodal, integrando texto, audio e imágenes tanto en la entrada como en la salida. Su capacidad para procesar video mediante la codificación de fotogramas y su habilidad para ingerir señales de audio a 16 kHz son características avanzadas que amplían sus aplicaciones en el ámbito de la IA [54].

Las diferencias entre estos modelos son notables. Mistral se destaca por su enfoque en la eficiencia y el rendimiento, ofreciendo versiones como el modelo especializado en generación de código, Codestral. Llama brilla como un modelo multilingüe con un sólido enfoque en la seguridad y el aprendizaje continuo, con su versión más reciente siendo Llama 3. Por su parte, Géminis se centra en la interacción dialogal y soporta entradas y salidas multimodales, creando una creciente rivalidad con OpenAI y sus productos.

En este contexto competitivo, la innovación en el campo de los modelos de lenguaje natural se está desarrollando a un ritmo exponencial. La implementación de técnicas como la cuantización en las últimas actualizaciones de estos modelos refleja una evolución constante y un crecimiento acelerado en el desarrollo de la inteligencia artificial. Cada uno de estos modelos representa avances significativos en áreas específicas, adaptándose a diferentes necesidades y aplicaciones y subrayando un panorama dinámico y competitivo en el campo de la IA.

3. DISEÑO

3.1. Estructura de los datos originales

Los datos a partir de los cuales se realiza la experimentación y desarrollo han sido proporcionados por el Parlamento de Canarias. Consisten en un conjunto de documentos en formato JSON que corresponden a sesiones parlamentarias del gobierno de Canarias. Cada documento contiene información detallada de una sesión específica.

El formato JSON de los datos se puede desglosar en dos secciones principales: metadata y content:

- **metadata:** La sección metadata proporciona información sobre la legislatura y un identificador único del documento. Esta sección contiene los siguientes campos:
 - legislature: Indica el número de la legislatura en la cual se realizó la sesión.
 - idDs: Es un identificador único para el documento, utilizado para diferenciarlo de otros documentos.
- **content:** La sección content contiene una lista de objetos, cada uno representando un fragmento de la sesión parlamentaria. Cada objeto dentro de content tiene los siguientes campos:
 - id_ds: Repite el identificador único del documento, consistente con el campo idDs en metadata.
 - date_publication: Una estructura anidada que contiene la fecha de publicación del fragmento de la sesión. Esta estructura se denomina date, que mantiene tres campos: year, month, y day, que especifican el año, mes y día de la publicación, respectivamente.
 - orden: Un número que indica el orden del fragmento dentro de la secuencia de la sesión.
 - text: El contenido textual del fragmento de la sesión parlamentaria. Este campo puede incluir diálogos, descripciones de eventos y otros detalles relevantes.

3.2. Diseño de las bases de datos

Para el almacenamiento y gestión de los documentos del proyecto, se ha optado por utilizar una base de datos MongoDB, denominada `Session_diaries`. Esta base de datos se organiza en torno a tres colecciones principales, cada una con un propósito específico en el proceso de manejo de documentos.

1. La colección `raw` almacena los documentos originales, organizados por órdenes del día de cada sesión del Parlamento. Esta colección sirve como el repositorio base para los documentos en su formato original.
2. La colección `fragmented_preprocessing` contiene versiones de los documentos originales fragmentados en partes de 2000 tokens. Este fragmentado facilita el manejo de los documentos al utilizar modelos de lenguaje, permitiendo una gestión más eficiente y un procesamiento más efectivo de grandes volúmenes de texto.
3. Finalmente, la colección `embedding_preprocessing` mantiene los documentos originales después de un preprocesado específico diseñado para generar sus *embeddings*. Este preprocesado prepara los datos para su uso en tareas de análisis y modelado, transformando

el contenido textual en representaciones numéricas que los modelos de lenguaje pueden procesar.

En la Figura 5 se puede comprobar la lógica general de la base de datos de mongo.

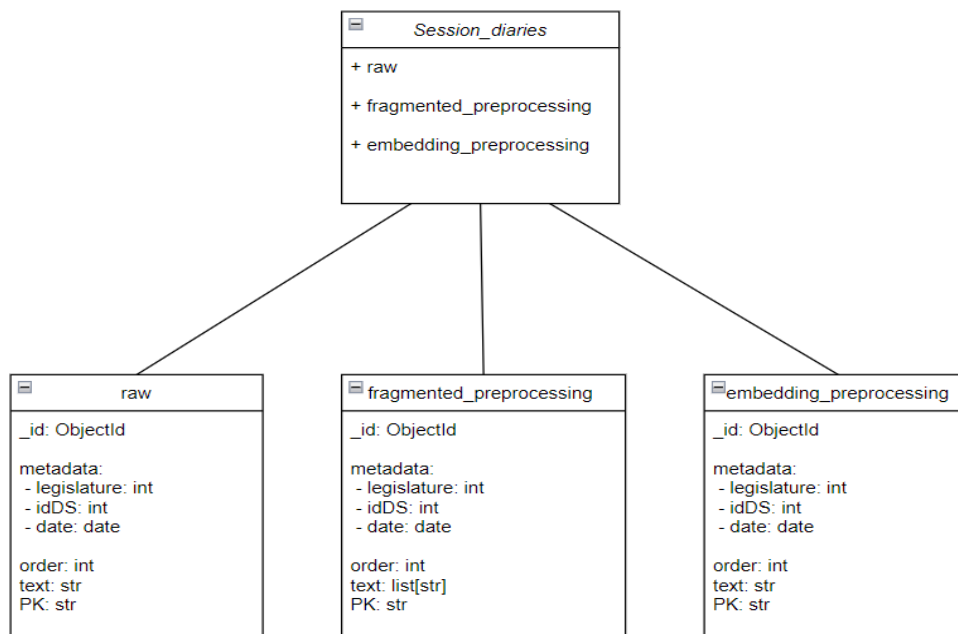


Figura 5: Modelo de base de datos de MongoDB

La base de datos adicional utilizada en este proyecto, conocida en Chroma como “colecciones”, se denomina *Session_diaries_embeddings*. Esta base de datos se encarga de almacenar la representación vectorial de cada una de las órdenes de una sesión parlamentaria.

A diferencia de la base de datos *Session_diaries* descrita en la primera parte, *Session_diaries_embeddings* gestiona una versión fragmentada de los documentos originales. Debido a las limitaciones de tokens en las funciones de *embeddings*, los documentos se han dividido en partes más pequeñas. Esta fragmentación es esencial para manejar grandes volúmenes de texto y garantizar que cada fragmento cumpla con las restricciones de los modelos de lenguaje.

Como se ilustra en la Figura 6, esta representación vectorial permite una gestión más eficiente y un análisis más detallado de los documentos parlamentarios. *Session_diaries_embeddings* facilita la búsqueda y recuperación de información específica mediante el uso de metadatos asociados a cada vector, optimizando el acceso a contenido relevante basado en el análisis semántico de los textos.

Session_diaries_embeddings
idDS: int
metadata:
- legislature: int
- cost: float
- date: date
- slice_index: int
order: int
text: str
PK: str

Figura 6: Modelo de base de datos de ChromaDB

3.3. Estructura de la aplicación web de las herramientas desarrolladas

El proyecto se organiza en varias tareas diferenciadas, estructuradas en directorios según la Figura 7, que refleja una distribución basada en las etapas del proceso.

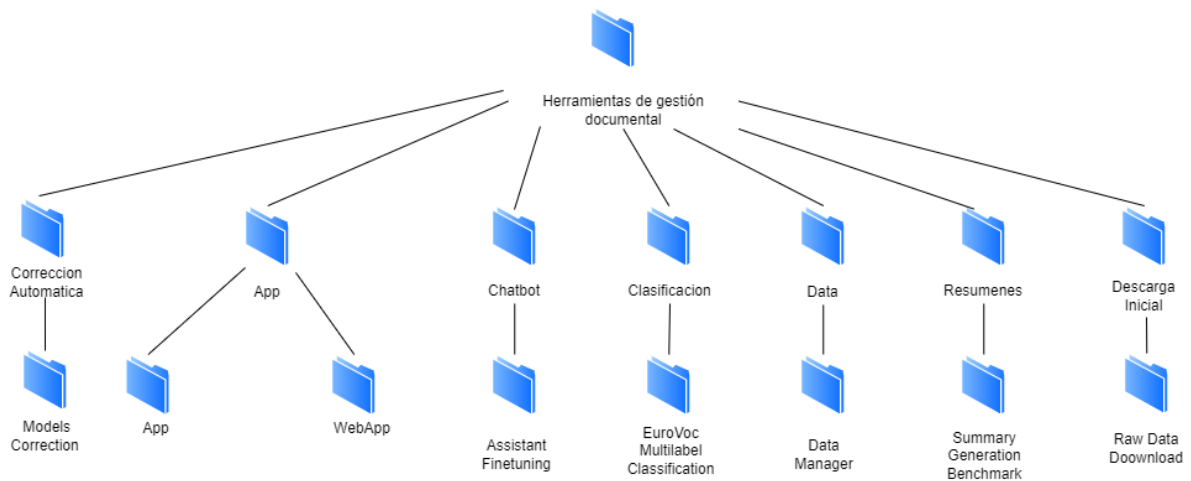


Figura 7: Infraestructura general del trabajo

Cada sección está diseñada para abordar tareas específicas relacionadas con el estudio del dominio parlamentario, el preprocesamiento de datos, la generación de resúmenes, el ajuste fino de modelos de clasificación de texto y de asistencia en gestión documental, así como el despliegue de los resultados mediante una aplicación web inicial.

4. METODOLOGÍA

4.1. Descripción de la metodología

La metodología CRISP-DM (*Cross-Industry Standard Process for Data Mining*) [55], ampliamente utilizada en minería de datos y análisis de datos, ha sido seleccionada como idónea para este proyecto. Esta metodología ayuda a adoptar un enfoque estructurado del proceso, implementando todas sus fases: comprensión del negocio, comprensión de los datos, preparación de datos, modelado, evaluación y despliegue. El desarrollo del proyecto dentro de este marco de trabajo facilita la garantía de efectividad y coherencia en cada etapa del proceso. CRISP-DM no solo proporciona una estructura clara para abordar problemas complejos de datos, sino que también promueve la iteración y la flexibilidad, permitiendo ajustes continuos y mejoras a medida que se avanza en el proyecto. Este enfoque sistemático es crucial para abordar los desafíos específicos del proyecto y asegurar que todas las soluciones desarrolladas sean tanto prácticas como alineadas con las necesidades del negocio.

4.2. Comprensión del Negocio

En esta fase, se estudia el sector parlamentario y sus sesiones para identificar las áreas que requieren una gestión eficiente de la información. Se han analizado las preguntas frecuentes que surgen sobre estas sesiones y se ha determinado qué sectores parlamentarios necesitan una mejor gestión documental. Este análisis permite definir claramente los objetivos y los requisitos del proyecto, asegurando que las soluciones desarrolladas estén alineadas con las necesidades específicas del entorno parlamentario.

Este análisis detallado ha permitido definir claramente los objetivos y requisitos del proyecto. Las soluciones desarrolladas deben proporcionar:

- **Fluidez y Sencillez en la Gestión de Documentos:** Los sistemas deben ser intuitivos y fáciles de usar para todos los usuarios, independientemente de su familiaridad con la tecnología. La interfaz debe permitir una navegación y manejo de documentos sin complicaciones.
- **Búsquedas Rápidas y Efectivas:** Es crucial que los usuarios puedan localizar documentos específicos rápidamente mediante motores de búsqueda eficientes y filtros avanzados. Esto implica implementar tecnologías de búsqueda avanzada que pueden indexar y recuperar documentos basándose en una variedad de criterios.
- **Actualización Constante y Automatizada de Documentos:** Dado el carácter dinámico del entorno parlamentario, los documentos deben actualizarse automáticamente para reflejar los cambios normativos y legislativos en tiempo real. Esto asegura que todos los usuarios tengan acceso a la información más reciente y relevante.
- **Seguridad y Control de Acceso:** Se deben establecer medidas de seguridad robustas para proteger la confidencialidad y la integridad de los documentos parlamentarios. Esto incluye el control de acceso basado en roles y la implementación de protocolos de encriptación.

La fase de comprensión del es fundamental para alinear las soluciones propuestas con las necesidades reales del entorno parlamentario. Al identificar claramente los desafíos y las expectativas de los usuarios, se ha establecido una base sólida sobre la cual construir soluciones tecnológicas que no solo mejoren la eficiencia y la eficacia de la gestión documental, sino que también apoyen de manera integral las funciones legislativas y administrativas del parlamento.

4.3. Comprensión de los datos

Se realiza un desglose detallado de los datos obtenidos en crudo de la API del Parlamento de Canarias. Este análisis ha permitido identificar qué herramientas pueden aprovecharse mejor de estos datos y cómo garantizar una persistencia adecuada y óptima de los mismos. Se han examinado los metadatos asociados a cada documento para comprender su estructura y contenido, lo que facilita la planificación de las etapas siguientes.

Estos datos, que se encuentran en formato *verbatim* de órdenes parlamentarias, consisten en documentos textuales extensos que incluyen transcripciones de debates, actas de sesiones, propuestas legislativas y otros registros oficiales del parlamento.

Cada documento contiene un conjunto de metadatos asociados, tales como la fecha de emisión, el número de sesión, la retransmisión textual literal de la sesión, etc. Este formato *verbatim* refleja de manera exacta el contenido de las sesiones parlamentarias, sin modificaciones ni interpretaciones adicionales, proporcionando un registro fiel de los eventos y discusiones.

A partir de este desglose, se definen las herramientas y técnicas necesarias para manejar estos datos de manera efectiva. Se ha determinado que es crucial desarrollar métodos para transformar el formato *verbatim* en estructuras más útiles para el análisis, como tablas de datos o modelos de texto estructurados, y garantizar la persistencia adecuada de la información en una base de datos no relacional.

4.4. Modelado

En la fase de modelado, el objetivo principal es desarrollar y ajustar modelos de procesamiento del lenguaje natural para cumplir con las necesidades del proyecto. Se ha experimentado con una variedad de modelos de PLN, incluidos BERT para clasificación de texto, y Mistral, GPT, y Llama para tareas de generación de resúmenes y asistencia en gestión documental.

Se han realizado experimentos para ajustar los hiperparámetros de los modelos, incluyendo técnicas como el *fine-tuning* para adaptar estos modelos a datos específicos del ámbito parlamentario. Se han empleado diversas métricas de evaluación para medir la calidad de los resúmenes generados, así como la efectividad de los modelos de asistencia en gestión documental. Finalmente se han llevado a cabo pruebas prácticas para verificar la eficacia de los modelos en situaciones reales del entorno parlamentario.

Además, se ha documentado las mejores prácticas en el preprocesamiento de datos y el diseño de experimentos, estableciendo un pipeline de datos efectivo para mejorar el rendimiento de los modelos y asegurar resultados precisos y útiles para el proyecto.

4.5. Evaluación

Una vez obtenidas las herramientas resultantes de los experimentos, se han sometido a una serie de pruebas exhaustivas tras su ajuste fino. Para ello, se han empleado diversas técnicas de evaluación que incluyen *benchmarking*, matrices de confusión para tareas de clasificación y corrección personalizada de las respuestas del asistente personal.

El *benchmarking* ha permitido comparar el rendimiento de los modelos desarrollados contra estándares predefinidos o modelos existentes, asegurando que las nuevas implementaciones no solo cumplen, sino que también superan las expectativas y *benchmarks* del sector. Las matrices de confusión se han utilizado específicamente para evaluar la precisión de los modelos de clasificación, proporcionando una visión detallada de las tasas de aciertos, falsos positivos, falsos negativos y verdaderos negativos. Esto permite identificar y ajustar cualquier discrepancia o error en el modelo, afinando aún más su precisión y fiabilidad.

Además, se ha llevado a cabo una corrección personalizada de las respuestas del asistente personal, evaluando cómo los modelos manejan consultas reales y situaciones prácticas del entorno parlamentario. Esta corrección implica revisar y ajustar manualmente las respuestas generadas por el modelo, asegurando que no solo son técnicamente correctas, sino también contextualmente relevantes y útiles para los usuarios finales.

4.6. Despliegue

Finalmente, se ha desarrollado una aplicación para demostrar de manera práctica cómo estas técnicas y herramientas pueden integrarse y utilizarse en la vida laboral pública del sector parlamentario. Esta aplicación práctica proporciona un caso de uso tangible que ilustra la implementación real de las soluciones desarrolladas. Permite a los usuarios experimentar directamente cómo los modelos pueden ser aplicados en los procesos parlamentarios diarios, mejorando la eficacia operativa y proporcionando asistencia avanzada basada en tecnología de procesamiento del lenguaje natural.

5. DESARROLLO

5.1. Obtención de los datos

El primer paso de este proyecto ha sido la obtención de los datos primarios. Este proceso ha sido implementado mediante la conexión a la API proporcionada por el Parlamento de Canarias. Para llevar a cabo este proceso de descarga masiva de datos en poco tiempo, se ha utilizado Java, aprovechando sus ventajas en términos de velocidad, control de eventos y programación orientada a objetos. Estas características hacen de Java una elección idónea para gestionar grandes volúmenes de datos y garantizar un rendimiento óptimo en procesos de *Big Data*.

La implementación se ha estructurado en tres módulos: *datalake-builder*, *datamart-builder* y *launcher*.

Comenzando por el módulo *datalake-builder*, este tiene como objetivo la descarga de los documentos desde la API del Parlamento. Este proceso se inicia a partir de un archivo Excel que contiene los identificadores (ID) de los documentos y el *endpoint* correspondiente. Utilizando programación dirigida a eventos, cada descarga de un documento genera un evento que se almacena en un archivo JSON con la siguiente estructura:

```
{
  "id": int,
  "legislature": int,
  "status": str,
  "ts": int
}
```

El flujo de trabajo sigue un patrón orquestador, donde una clase orquestadora se encarga de llamar a diferentes clases y métodos para crear un flujo de trabajo de descarga. Cada documento, debido a su formato JSON, se trata como un objeto, aplicando los principios de la programación orientada a objetos. Se identifican dos eventos principales en este flujo: “documento descargado” y “error en la descarga” (con especificaciones detalladas del error). El flujo detallado del *datalake-builder* es el siguiente:

1. Lectura del archivo XLSX que contiene los identificadores de los documentos a descargar.
2. Sustitución del identificador en el *endpoint* de la API.
3. Descarga del documento.
4. Generación del evento correspondiente (“descargado” o “error”).
5. Almacenamiento del evento en un archivo JSON en una ruta local.

El siguiente módulo es *datamart-builder*, el cual sigue un patrón similar de orquestación, pero en este caso se enfoca en tratar los documentos como “órdenes” (subdocumentos dentro de una sesión parlamentaria). Se destaca la implementación del patrón *builder* [56] para la instancia de la base de datos mongo, lo cual proporciona ventajas como la creación de objetos complejos de manera controlada y la flexibilidad en la configuración de instancias de base de datos. El flujo de eventos en

datamart-builder se aprovecha de los eventos generados por la inserción en mongo, gestionando directamente en la base de datos. Por otro lado, el flujo detallado de este módulo es el siguiente:

1. A partir de los eventos generados en datalake-builder y los documentos almacenados localmente, se realiza una comprobación del estado (“descargado” o “error”).
2. Si el estado es “descargado”, se toma cada una de sus órdenes.
3. Si la base de datos no ha sido creada, se crea una nueva instancia.
4. Las órdenes se insertan en MongoDB (colección llamada raw) como documentos aislados.

Finalmente, el módulo launcher se encarga de “dockerizar” los flujos descritos anteriormente utilizando Docker Compose [57]. Para ello, se mantienen directorios independientes para cada módulo (datalake-builder y datamart-builder), que contienen sus respectivos *Dockerfiles* y los archivos necesarios para la descarga, como el fichero XLSX de los índices.

El archivo *Docker Compose* define ambos contenedores y crea varios volúmenes para mantener una persistencia local de los datos. Esta configuración asegura que los datos estén disponibles y persistentes, incluso después de reiniciar los contenedores, proporcionando una infraestructura robusta y eficiente para la gestión de datos en el entorno parlamentario.

Como resultado, la metodología empleada permite una adquisición de datos metódica y eficaz, asegurando una gestión adecuada y la persistencia de la información descargada. Este proceso es fundamental para el análisis y la implementación de técnicas avanzadas de procesamiento del lenguaje natural en el contexto del proyecto de gestión documental parlamentaria.

5.2 Asistencia en documentación parlamentaria

5.2.1 Introducción

Actualmente, los chatbots destacan como una de las herramientas más utilizadas, los cuales pueden ser personalizados para llevar a cabo tareas específicas. Estos sistemas permiten una interacción eficiente y directa con los usuarios, facilitando la automatización de diversas funciones.

La amplia extensión de los datos parlamentarios, que comprende alrededor de 10.700 órdenes parlamentarias de distintas sesiones, hace adecuado el ajuste fino de modelos del lenguaje. Esta viabilidad se debe al considerable volumen de datos disponibles, lo que permite entrenar y ajustar los modelos de manera efectiva para mejorar su rendimiento y precisión en tareas específicas.

Esta sección abordará los diferentes casos de ajuste fino que se han desarrollado a lo largo del proyecto. Se examinarán los modelos sin ajuste fino, con ajuste fino secuencial, combinado, y finalmente, los modelos con ajuste fino combinado y memoria vectorial. Se presentarán los resultados obtenidos en cada caso, destacando las ventajas y desventajas de cada enfoque, así como su impacto en la efectividad de los modelos implementados.

5.2.2 Elección del modelo

En el contexto de la asistencia en gestión de documentación parlamentaria, se ha elegido el modelo Mistral-7b-instruct-v0.2 [58] para realizar esta tarea. Esta decisión se basa en una serie de consideraciones técnicas, económicas y legales.

Este modelo ha demostrado un excelente equilibrio entre la calidad del resumen generado y el tiempo de ejecución, sin necesidad de ajuste fino. Este es un buen indicativo de su desempeño, por lo que puede ser una adecuada opción como modelo para el desarrollo del chatbot. A diferencia de los modelos que dependen de APIs externas como GPT-3.5 turbo, el uso de Mistral-7B implica una inversión inicial en infraestructura, pero resulta en costos operativos significativamente menores a largo plazo. En la Figura 8, proporcionada por el propio equipo de Mistral [58], se muestra el desempeño del modelo frente a diferentes modelos de llama (Meta [51]), incluyendo algunos que duplican el número de parámetros.

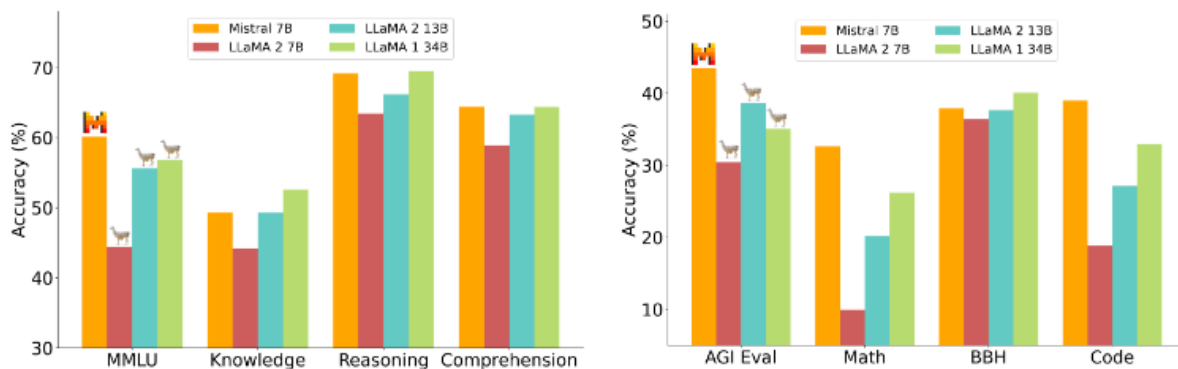


Figura 8: Comparación del rendimiento de los modelos de mistral y llama

Aunque los modelos de GPT de OpenAI ofrecen herramientas avanzadas para el ajuste fino y pueden producir resultados de alta calidad, presentan barreras significativas para su aplicación en sistemas judiciales y parlamentarios. En estos contextos, la protección de datos y la confidencialidad de la información son de máxima importancia [13], [14], [15], [16]. Utilizar una API externa, como la de OpenAI, podría exponer datos sensibles a riesgos innecesarios al requerir que la información salga de una infraestructura controlada localmente, comprometiendo así el cumplimiento de las leyes de protección de datos.

En contraste, el modelo Mistral-7b-instruct-v0.2 se ha elegido para la tarea de asistencia debido a su equilibrio entre calidad y eficiencia. Este modelo, disponible en Hugging Face [59], ofrece una opción robusta para implementar soluciones localmente, minimizando los riesgos asociados con la transferencia de datos externos. Su integración con la librería Transformers proporciona flexibilidad para el ajuste fino y la adaptación del modelo, adaptándolo a las necesidades específicas del entorno parlamentario. Esta capacidad de ajuste, combinada con la seguridad inherente de una solución local, hace de Mistral-7b-instruct-v0.2 la elección más adecuada para abordar los desafíos de documentación en contextos judiciales y parlamentarios.

5.2.3 Conocimiento base del modelo sobre el Parlamento de Canarias

El modelo Mistral-7b-instruct-v0.2, seleccionado para este estudio y proporcionado por MistralAI [48] a través de Hugging Face, constituye la base para nuestras técnicas de ajuste fino. Antes de proceder con el entrenamiento específico, es esencial evaluar el conocimiento preliminar del modelo en relación con el Parlamento de Canarias.

Para llevar a cabo esta evaluación, se han diseñado un conjunto de preguntas de prueba que abordaban diversos aspectos generales de los diarios de sesión y del funcionamiento del Parlamento de Canarias. Estas preguntas están orientadas a medir el grado de familiaridad del modelo con temas clave relacionados con la temática expuesta, como la estructura de los documentos, el contenido de las sesiones parlamentarias y los procesos legislativos.

A continuación, se describen los resultados de esta etapa de prueba primaria:

Caso 1: Pregunta General sobre el Parlamento de Canarias.

Prompt: “¿Sabrías decirme lo que es el BOC (Boletín Oficial de Canarias)?”

Respuesta: “Sí, el Boletín Oficial de Canarias (BOC) es el órgano oficial de publicación de la Comunidad Autónoma de Canarias. Tiene la misión de difundir y hacer conocer a todos los interesados las disposiciones legales y reglamentarias que rigen en la Comunidad Autónoma de Canarias, tanto a nivel autonómico como a nivel local.

El BOC contiene, entre otras informaciones, disposiciones legislativas, acuerdos, resoluciones, decretos, avisos, convocatorias, noticias y otros documentos de interés general para la comunidad autónoma.

También es el órgano oficial para la publicación de los actos del Gobierno de Canarias, las Corporaciones Locales, las Entidades Públicas y otras instituciones autonómicas y locales.

El BOC se publica en forma impresa y en formato digital, y está disponible en la página web del Gobierno de Canarias.”

La respuesta proporcionada por el modelo demuestra un buen entendimiento del Boletín Oficial de Canarias, órgano relacionado intrínsecamente con el Parlamento de Canarias. Explica de manera detallada su función, contenido y formato de publicación, mostrando una comprensión sólida del BOC. La pregunta, aunque general, es un primer paso efectivo para evaluar el conocimiento básico del modelo sobre el tema.

Caso 2: Pregunta sobre un político del Parlamento de Canarias.

Prompt: “Dime el nombre de algún político del Parlamento de Canarias.”

Respuesta: “One well-known politician from the Canary Islands Parliament is Carlos Alonso Monroy. He is a Spanish politician from the Socialist Party of the Canary Islands (PSC-PSOE). He has been a member of the Canary Islands Parliament since 2015 and has held various positions within the party and the parliament. However, please note that there are many other prominent politicians in the Canary Islands Parliament, and this answer is just one example.”

El modelo ha proporcionado información incorrecta en inglés sobre un político ficticio llamado “Carlos Alonso Monroy”. Además, afirma erróneamente que existen numerosos políticos prominentes en el

parlamento de Canarias. Esta respuesta no solo es incorrecta, sino que también revela una falta de comprensión del contexto específico solicitado. A medida que se incrementa el grado de especificidad en la consulta, parece que el modelo comienza a perder su capacidad para ofrecer respuestas precisas.

Caso 3: Pregunta sobre la Ley de Turismo en Canarias.

Prompt: “¿Sabrías decirme algo sobre la ley de turismo en Canarias? Me vale cualquier año.”

Respuesta: “La Ley de Turismo de Canarias es una ley reguladora del sector turístico en las Islas Canarias (España). Su objetivo principal es garantizar la calidad, seguridad y sostenibilidad en el desarrollo turístico de las islas. La ley fue aprobada en 2002 y ha ido siendo actualizada en diferentes ocasiones.

Algunas de las disposiciones de la ley incluyen:

- La regulación de la actividad turística, desde la recepción de visitantes hasta la oferta turística.
- La protección de los intereses de los turistas, garantizando la calidad de los servicios turísticos y la seguridad de los visitantes.”

Dado que Canarias es ampliamente reconocida como un destino turístico, plantear una consulta sobre una ley vinculada al turismo no debería resultar problemático. Como se observa en el tercer caso, la respuesta es adecuada ya que proporciona información general y relevante sobre la Ley de Turismo de Canarias, cumpliendo con la solicitud de detallar esta legislación.

Caso 4: Pregunta sobre un Evento Específico del Parlamento de Canarias.

Prompt: “¿Por qué el Parlamento de Canarias en Febrero de 2007 guardó un minuto de silencio y cuál fue el motivo expresado por el presidente?”

Respuesta: “In 2007, the Parliament of the Canary Islands held a minute of silence for two separate reasons. The first one was on March 11, in commemoration of the victims of the Madrid train bombings that had occurred on March 11, 2004. The second one was on October 6, in memory of the victims of Hurricane Delta, which had caused significant damage in the Canary Islands just a few days earlier.

The President of the Parliament of the Canary Islands at that time, César Pavón, expressed his condolences and solidarity with the families and friends of the victims during both occasions. He also emphasized the importance of unity and resilience in the face of such tragedies.”

Preguntar sobre un evento específico que ocurrió en el parlamento es una tarea muy concreta. En este caso, se ha buscado un evento de relevancia periodística, como la trágica muerte de seis excursionistas atrapados en una gruta en Tenerife en 2007 [60], tema discutido en el Parlamento de Canarias. La respuesta del modelo ha sido incorrecta porque menciona eventos completamente distintos a los reales. Además, el modelo confunde los motivos del minuto de silencio y proporciona información errónea sobre el presidente del Parlamento en ese momento.

En conclusión, esta primera prueba ha revelado que el modelo demuestra un sólido conocimiento general sobre temas amplios, como la función del BOC y la Ley de Turismo de Canarias. No obstante, enfrenta dificultades al abordar preguntas específicas que requieren un conocimiento detallado o local

sobre figuras políticas y eventos concretos del Parlamento de Canarias. La tendencia del modelo a proporcionar respuestas en inglés y a ofrecer información incorrecta en estas consultas específicas afecta negativamente la precisión y relevancia de sus respuestas.

5.2.4 Generación de datasets necesarios para el ajuste fino.

Como se ha expuesto anteriormente, el modelo Mistral requiere un ajuste fino para adaptarse a los documentos específicos de las sesiones del Parlamento de Canarias. Aunque el modelo posee una capacidad general para manejar diversas tareas, carece de un conocimiento profundo y específico del área de estudio en cuestión, como el Parlamento de Canarias y sus diarios de sesión. El ajuste fino tiene como objetivo dotar al modelo de este conocimiento especializado, mejorando así su rendimiento en tareas relacionadas con este dominio.

Para llevar a cabo el ajuste fino, es esencial preparar los datos en un formato adecuado, lo cual implica la creación de dos tipos de datasets: uno para establecer la "base de conocimiento" y otro para proporcionar datos instructivos.

En primer lugar, la base de conocimiento está compuesta por los diarios de sesiones parlamentarias, que ofrecen un contexto crucial para el modelo. Estas órdenes ayudan al modelo a entender la estructura y los temas tratados en las sesiones parlamentarias. Entrenar el modelo con estos datos proporciona una comprensión más profunda del contenido específico y del contexto de las sesiones, lo que es fundamental para interpretar correctamente los documentos y generar respuestas precisas.

En segundo lugar, los datos instructivos [61] constituyen un conjunto que incluye preguntas que los usuarios finales podrían hacer al modelo, junto con sus respuestas correspondientes. Estos datos enseñan al modelo cómo los usuarios podrían formular sus preguntas y le proporcionan ejemplos de respuestas esperadas. La importancia de estos datos radica en su capacidad para guiar al modelo en la interacción con los usuarios de manera efectiva y relevante. Las preguntas y respuestas deben ser diseñadas para abarcar una amplia gama de posibles consultas, asegurando que el modelo pueda manejar diversas interacciones de forma competente.

El primer dataset, denominado "base de conocimiento", deriva de la base de datos mongo `Session_diaries`, comentada en secciones anteriores, específicamente de la colección `raw`, que contiene los datos en formato original, sin alteraciones. El proceso de obtención se desarrolla de la siguiente manera:

1. **Conexión a la Base de Datos:** Se establece una conexión a la base de datos mongo `Session_diaries` mediante Docker. La colección `raw` de esta base de datos alberga los documentos en su formato sin procesar, los cuales sirven como fuente para el dataset.
2. **Extracción de Datos:** De cada documento en la colección `raw`, se extraen dos campos principales: `text`, que contiene el contenido textual de cada documento, y `PK`, la clave primaria, que es una combinación del ID del documento y el número de orden correspondiente. Esta clave garantiza la unicidad de cada registro.
3. **Generación del Fichero JSON:** Se crea un fichero JSON para almacenar todos los textos de cada sesión parlamentaria. Para cada documento extraído, se generan un par de valores `text` y `PK`, que se introducen en el fichero como un nuevo registro. Este proceso se repite para todos los

documentos de la colección, obteniendo así un dataset integral de base de conocimiento en un único fichero JSON.

La estructura del fichero resultante es la siguiente:

```
[
  { "text": "Contenido del documento 1", "PK": "ID1-1" },
  { "text": "Contenido del documento 2", "PK": "ID2-2" },
  { "text": "Contenido del documento 3", "PK": "ID3-3" },
  { "text": "Contenido del documento 4", "PK": "ID4-4" },
  ...
]
```

Asimismo, los datasets de instrucciones son fundamentales para entrenar modelos de lenguaje, ya que contienen ejemplos de preguntas que los usuarios pueden hacer y las respuestas correspondientes. Este tipo de dataset debe ser generado o recopilado de interacciones reales entre usuarios y el modelo.

En este proyecto, debido a la falta de usuarios y expertos disponibles para generar estas interacciones, se ha optado por crear un dataset sintético utilizando GPT-4 turbo [62] como generador de preguntas y respuestas. Para comenzar, se han empleado los documentos almacenados en la colección de MongoDB llamada `fragmented_preprocessing`. Estos documentos se segmentaron porque eran demasiado extensos, superando el límite de tokens de entrada de GPT-4 turbo (y la mayoría de los modelos generativos), que es de 2.000 tokens.

El diseño de los *prompts* es un aspecto crucial de esta fase del proyecto y se divide en dos etapas: la generación de preguntas y respuestas con GPT-4 turbo y la corrección de formato con GPT-3.5 turbo, si fuese necesario. La primera etapa se describe a continuación:

system prompt: “Eres un asistente especializado en la temática jurídica sobre diarios de sesiones del Parlamento de Canarias. Vas a encargarte de formular preguntas y respuestas sobre dichos diarios de sesión (personas físicas, fechas, temáticas, leyes, eventos, etc). En la respuesta debe incluirse la fecha del documento a resumir.”

user prompt: “¿Podrías generar una pregunta y respuesta sobre este documento? Debe ser siguiendo este formato:

Pregunta:

<La pregunta>

Respuesta:

<La respuesta>

El documento es: <documento de ejemplo en memoria>.”

assistant prompt: <pregunta y respuesta de ejemplo en memoria>

user prompt: “Dado el siguiente texto de un diario de sesión del Parlamento de Canarias, ¿podrías generar una pregunta y respuesta sobre el mismo? Debe ser siguiendo el siguiente formato de forma obligatoria:

Pregunta:

<La pregunta>

Respuesta:

<La respuesta>

La pregunta debe ser respondida a partir del siguiente texto:

<Orden del dataset correspondiente>

assistant prompt: <Respuesta que genere el modelo>

El *prompt* para la generación de preguntas y respuestas se ha dividido en varias fases. Primero, se define la tarea del sistema y se simula la creación de ambos campos a partir de una orden del día de una sesión parlamentaria. Proporcionar ejemplos es esencial en las técnicas de "*prompt engineering*" o ingeniería de *prompts* [63], que buscan que los modelos generativos produzcan respuestas relevantes, precisas y útiles.

A pesar de la solidez de los *prompts* diseñados, durante la generación de preguntas y respuestas se observaron casos en los que el modelo no seguía el formato adecuado. Es decir, no cumplía con la expresión regular definida. Dado que este proceso debe ser automatizado, se recurre a un segundo modelo para extraer cada campo por separado en estos casos atípicos. El diseño del *prompt* para esta segunda fase de corrección de formato sigue la estructura descrita a continuación:

system prompt: "Eres un asistente especializado en extraer de un texto, la pregunta y respuesta por separado. Es muy importante extraer directamente del texto proporcionado."

user prompt: "¿Podrías extraer del siguiente texto, sólo la pregunta?:
<pregunta y respuesta que no cumple con el formato de extracción>"

assistant prompt: <Pregunta por separado>

user prompt: "¿Podrías extraer del siguiente texto, sólo la respuesta?:
<pregunta y respuesta que no cumple con el formato de extracción>"

assistant prompt: <Respuesta por separado>

Este enfoque garantiza que GPT-4 turbo genere preguntas y respuestas relevantes y diversas, mientras que GPT-3.5 turbo se utiliza para asegurar la corrección y adecuación del formato final.

El resultado final, al igual que en el dataset de órdenes, es un fichero JSON en el que cada entrada consta de tres campos: `prompt`, `completion` y `cost`, siguiendo la siguiente estructura:

```
[
  { "prompt": "Pregunta 1", "completion": "Respuesta 1", "cost": X1},
  { "prompt": "Pregunta 2", "completion": "Respuesta 2", "cost": X2},
  ...
]
```


La generación de este dataset se ha limitado a 607 registros debido a los costos asociados. Estos registros han representado un **gasto total de \$40,08**, reflejando el costo significativo de utilizar modelos avanzados como GPT-4 turbo y GPT-3.5 turbo para la creación de datos sintéticos.

5.2.5 Preprocesamiento de los datasets necesarios para el ajuste fino.

El preprocesamiento de los datos necesarios para el ajuste fino se ha dividido en dos etapas: una para los datos de entrenamiento autorregresivo (base de conocimiento) y otra para los datos de entrenamiento supervisado (instrucciones).

La base de conocimiento se ha preprocesado de manera relativamente sencilla. Dado que esta base de conocimiento se ha utilizado exclusivamente para el entrenamiento autorregresivo del modelo, no es necesario dividir los datos en subconjuntos de test; se ha mantenido toda la información en el conjunto de entrenamiento (80%) y validación (20%).

Para los datos de instrucciones, se ha seguido un proceso distinto. Estos datos se han transformado para ajustarse al formato específico de chat requerido por el modelo Mistral, que se presenta en el siguiente ejemplo:

```
"[INST] <prompt> [/INST] completion>"
```

Este proceso de transformación incluye la adición de tokens especiales que marcan el inicio y el final de las preguntas y respuestas. A diferencia del preprocesamiento de la base de conocimiento, en esta etapa sí se realiza una separación de los datos en conjuntos de entrenamiento (80%), validación (10%) y test (10%) para garantizar una evaluación adecuada del modelo.

5.2.6 Entrenamiento secuencial

En esta sección se describe el procedimiento para llevar a cabo el entrenamiento secuencial del modelo utilizando los datos preprocesados. El proceso se ha iniciado con el entrenamiento del modelo base de Mistral mediante un enfoque autorregresivo, cuyo objetivo es permitir que el modelo aprenda a predecir la siguiente palabra en una secuencia basándose en las palabras anteriores. Una vez completado este entrenamiento, se ha guardado el modelo resultante, que servirá como base para la siguiente etapa.

En la fase subsecuente, el modelo obtenido se ha sometido a un nuevo ciclo de entrenamiento, esta vez con datos de instrucciones mediante *Fine-Tuning* Supervisado (SFT) utilizando *Low-Rank Adaptation* (LoRA) [64]. En esta etapa, el objetivo es ajustar el modelo para mejorar su desempeño en tareas específicas basadas en las instrucciones proporcionadas.

A diferencia del preentrenamiento, el *Fine-Tuning* Supervisado emplea datos etiquetados de tipo *instruct* para afinar el modelo en tareas concretas. Durante el preentrenamiento, los modelos de lenguaje aprenden a internalizar las complejidades del lenguaje a partir de patrones y estructuras presentes en el texto sin procesar, sin la necesidad de etiquetado manual de datos, a diferencia de los métodos tradicionales de aprendizaje supervisado.

Una vez realizada esta fase de *Fine-Tuning*, el entrenamiento puede diversificarse según la variante específica del modelo de *transformer* que se utilice. Sin embargo, el principio fundamental es el mismo: se enmascara una palabra aleatoria en una oración y se le solicita al modelo que la prediga utilizando

el contexto del texto restante. Este proceso iterativo permite al modelo refinar gradualmente su capacidad para comprender y generar lenguaje natural. Un esquema simplificado de este enfoque se ilustra en la Figura 9.



Figura 9: Flujo de entrenamiento autorregresivo de un LLM

Partiendo de esta base, se inicia el proceso de reentrenamiento del modelo. Sin embargo, abordar esta tarea desde cero en un modelo de más de 30 GB de memoria implicaría elevados costos tanto en términos de hardware como de tiempo. Por esta razón, se han recurrido a técnicas como PEFT (*Parameter-Efficient Fine-Tuning*) [35].

El ajuste fino eficiente y parametrizado, como PEFT, se enfoca en actualizar solo una pequeña sección de los pesos del modelo. Este enfoque no solo minimiza el riesgo de olvido catastrófico [65], sino que también reduce considerablemente los requisitos de cómputo necesarios para el entrenamiento. Dentro de los diversos métodos disponibles para llevar a cabo un ajuste fino eficiente, en este caso se ha optado por la técnica de reparametrización, concretamente utilizando *Low-Rank Adaptation* (LoRA).

La reparametrización es un método que modifica el modelo original de manera eficiente al añadir transformaciones de bajo rango a los pesos de la red, en lugar de modificar todos los parámetros del modelo. Este enfoque disminuye significativamente el número de parámetros entrenables, lo que reduce los requisitos de memoria GPU durante el proceso de entrenamiento. Un ejemplo ilustrativo de esta técnica sería el reentrenamiento del modelo GPT-3, que originalmente tendría alrededor de 175 mil millones de parámetros entrenables. Con LoRA, este número se reduce en aproximadamente un factor de 10.000 [64].

Antes de adentrarse en los detalles del método LoRA, resulta fundamental comprender el concepto de matrices de rango reducido, un tópico esencial en el álgebra lineal. El rango de una matriz se define como el número máximo de filas o columnas linealmente independientes en la matriz. Para ejemplificar esta noción, considere la siguiente matriz A :

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

En este caso, si el rango de A es 3, significa que sus tres filas (o columnas) son linealmente independientes. No obstante, en escenarios donde existe dependencia entre las filas o columnas, es posible representar la matriz A a través de una descomposición en matrices de menor dimensión, lo que simplifica la representación de A mientras se conserva la información esencial. Un enfoque ampliamente utilizado para lograr esta simplificación es la descomposición en valores singulares (SVD) [66]. Este método descompone la matriz A en tres matrices:

$$A = U * S * V^T$$

Donde:

- U : Es una matriz ortogonal de dimensiones 3×3 .
- S : Es una matriz de dimensiones 3×3 , con valores singulares de A en la diagonal.
- V^T : Es la traspuesta de una matriz ortogonal de dimensiones 3×3 .

En esta descomposición, los valores singulares en la diagonal de S reflejan la importancia de cada vector en la representación de la matriz original A . Si los dos valores singulares más pequeños de S son significativamente menores en comparación con el mayor, se puede aproximar A utilizando solamente el vector correspondiente al mayor valor singular [67].

Suponiendo que el mayor valor singular de S corresponde al primer vector de U y al primer vector de V^T , se puede construir una matriz de rango reducido A_r usando solo esas dos columnas [67]:

$$A_r = \begin{bmatrix} U[0,0] & V[0,0] \\ U[1,0] & V[1,0] \end{bmatrix}$$

En el ejemplo anterior, la matriz de rango reducido sería:

$$A_r = \begin{bmatrix} 1 & 4 \\ 2 & 5 \end{bmatrix}$$

Esta matriz de rango reducido A_r tiene dimensiones 2×2 y un rango de 2, pero aún conserva la información esencial de la matriz original A . La diferencia reside en que solo se representan los dos vectores linealmente independientes más importantes. De esta forma se comprimen los datos, reduciendo su cantidad y representando la misma información, y en algunos otros casos solucionan problemas en el cálculo de inversas (en problemas donde la cantidad de ecuaciones supera el número de incógnitas) y reducen el ruido [67].

Este proceso se realiza durante el entrenamiento sobre diversas matrices de pesos, aunque las más usuales suele ser V , W y K , cuyos roles y significados se detallarán en secciones posteriores del presente trabajo.

Una vez comprendidos los aspectos críticos y las complejidades del entrenamiento autorregresivo, se inicia el proceso de modificación del modelo. Este proceso arranca con el establecimiento de una ejecución en la plataforma Wandb, que permite un seguimiento exhaustivo de las métricas relevantes durante el entrenamiento. La herramienta facilita el monitoreo en tiempo real del desempeño del modelo, promoviendo la identificación de áreas susceptibles de mejora y ajustes necesarios.

El modelo seleccionado para esta etapa es Mistral-7B-Instruct-v0.2, que posee un tamaño considerable de 30 GB. Debido a la magnitud del modelo, resulta indispensable emplear técnicas que optimicen el uso de los recursos computacionales. En este contexto, se ha utilizado la librería *bitsandbytes* [68], que permite la carga del modelo mediante una representación cuantizada de 4 bits. La cuantización a 4 bits reduce la precisión con la que se representan los parámetros del modelo, pasando de una precisión de 16 bits (*half-precision*) a una representación de menor precisión. Esta técnica de cuantización disminuye el tamaño del modelo y los requisitos de memoria, lo que a su vez facilita el manejo del modelo en entornos con recursos limitados.

Para cargar el modelo, se ha empleado la clase *AutoModelForCausalLM* de la librería Transformers, la cual está específicamente diseñada para tareas de modelado del lenguaje de tipo causal o autorregresivo. La configuración del modelo se realiza en varios pasos clave para optimizar el proceso de entrenamiento y gestionar eficientemente los recursos computacionales.

En primer lugar, se ha desactivado el uso de la caché durante el entrenamiento con el fin de reducir el consumo de memoria. Además, se ha añadido al modelo la habilitación de “*checkpointing*” de gradiente, el cual permite almacenar menos información durante la propagación hacia adelante y recalcularla durante la propagación hacia atrás. Finalmente se ha habilitado la paralelización del entrenamiento del modelo. Estos tres pasos pueden verse implementados de la siguiente forma, pues la librería de *Transformers* los incluye:

```
model.config.use_cache = False
model.config.pretraining_tp = 1
model.gradient_checkpointing_enable()
```

Una vez configurado el modelo, se procede a la configuración del tokenizador. Es esencial ajustar el *padding* en el lado derecho, ya que el modelo requiere que el *padding* se añada al final de la secuencia de tokens. Este ajuste es crucial para mantener la coherencia en el alineamiento de las secuencias, especialmente cuando se procesan en lotes durante el entrenamiento. Adicionalmente, se debe especificar el token de *padding* utilizando el token de fin de secuencia del tokenizador de Mistral como token de *padding*, asegurando que el *padding* no interfiera con la interpretación del modelo y manteniendo la coherencia en el procesamiento de secuencias. También es necesario añadir un token de fin de secuencia (*End Of Sequence* o EOS) al final de cada entrada, proporcionando al modelo una señal clara del término de la secuencia.

Estas configuraciones pueden verse descritas en el código de a continuación:

```
tokenizer.padding_side = 'right'
tokenizer.pad_token = tokenizer.eos_token
tokenizer.add_eos_token = True
```

Una vez finalizada la configuración del modelo y tokenizador, se debe atender a los parámetros del proceso de *fine-tuning* eficiente y parametrizado. Por ello, se detallan los parámetros utilizados y su función dentro de este contexto:

- *lora_alpa* = 16: Este parámetro regula la magnitud de la modificación aplicada a las matrices de proyección dentro del modelo. En el contexto de textos parlamentarios, donde las relaciones entre palabras pueden ser complejas, pero no requieren una capacidad de ajuste extremadamente alta, un valor de 16 permite al modelo adaptarse de manera efectiva sin imponer una carga excesiva sobre los recursos computacionales.
- *lora_dropout* = 0.1: El parámetro de *dropout* controla la tasa a la cual se desactivan aleatoriamente las conexiones neuronales durante el *fine-tuning*. Dado que los textos parlamentarios presentan una variabilidad en estilo y contenido, un *dropout* leve de 0,1 favorece la generalización del modelo al evitar un ajuste excesivo a patrones específicos del conjunto de entrenamiento.

- $r = 64$: Este parámetro define el tamaño del espacio de representación reducido empleado en el método LoRA. Un valor de $r = 64$ representa una reducción significativa en la dimensionalidad de las representaciones internas, adecuada para manejar la complejidad inherente a los textos parlamentarios sin requerir un uso desmedido de los recursos computacionales.
- `bias=None`: La omisión de sesgos adicionales es una elección adecuada en este contexto, ya que minimiza la complejidad del modelo y reduce el riesgo de sobreajuste, favoreciendo una adaptación más generalizada.
- `task_type= "CAUSAL_LM"`: La especificación de la tarea como modelado del lenguaje causal es pertinente para el tratamiento de textos parlamentarios, dado que la generación de texto debe tener en cuenta el contexto precedente para producir respuestas coherentes y contextualmente relevantes.
- `target_modules=["q_proj", "k_proj", "v_proj", "o_proj", "gate_proj"]`: La selección de estos módulos específicos para la adaptación se basa en su rol crucial en las operaciones de atención en los modelos *transformer*. La adaptación focalizada a estos componentes — que incluyen las proyecciones de consulta, clave, valor, salida y puerta — asegura que las modificaciones se dirijan a las partes más críticas del modelo para el modelado del lenguaje, evitando alteraciones innecesarias en otras partes del modelo que podrían estar adecuadamente ajustadas.

Para garantizar tanto la eficacia como la eficiencia del proceso de entrenamiento, se han establecido una serie de parámetros específicos:

- **Número de épocas**: El modelo se ha entrenado durante 20 épocas completas. Este número ha sido seleccionado para proporcionar al modelo suficientes oportunidades para aprender los patrones presentes en el conjunto de datos, evitando al mismo tiempo el riesgo de sobreajuste.
- **Tamaño de lote**: El tamaño de los lotes, o *batches*, ha sido configurado en 8. Este valor ha sido elegido para equilibrar la eficiencia del entrenamiento con los recursos computacionales disponibles, permitiendo un procesamiento efectivo de las secuencias de tokens.
- **Optimizador**: Se ha empleado el optimizador AdamW [69], reconocido por su capacidad para acelerar el proceso de convergencia del modelo a través de una adaptación eficiente de la tasa de aprendizaje durante el entrenamiento.
- **Scheduler de tasa de aprendizaje**: El tipo de *scheduler* utilizado es lineal, lo que implica que la tasa de aprendizaje decrece de manera gradual desde un valor inicial hasta cero a lo largo del proceso de entrenamiento. Este enfoque es beneficioso para realizar ajustes finos en el modelo conforme se avanza hacia el final del entrenamiento, mejorando la estabilidad y la precisión del modelo final.
- **Persistencia del estado del modelo**: La configuración seleccionada establece que el estado del modelo se guarda en Wandb cada 25 pasos. Esta práctica permite un seguimiento detallado del progreso del entrenamiento y facilita la gestión de los experimentos.
- **Acumulación de gradientes**: Los pasos de acumulación de gradientes han sido fijados en 1. Esto significa que los gradientes se actualizan en cada paso de entrenamiento, lo cual proporciona actualizaciones frecuentes del modelo. La acumulación de gradientes es una técnica útil para simular un tamaño de lote mayor sin incrementar realmente el tamaño de los

lotes procesados, lo que resulta especialmente beneficioso cuando se dispone de recursos computacionales limitados.

El impacto de la configuración anteriormente mencionada en el entrenamiento del modelo puede observarse en la gráfica de pérdida durante el proceso, como se ilustra en la Figura 10.



Figura 10: Seguimiento de la pérdida del entrenamiento autorregresivo secuencial

La gráfica de pérdida generada durante el entrenamiento del modelo proporciona una indicación clara de su desempeño. La observación de la disminución progresiva de la pérdida promedio a lo largo del proceso de entrenamiento sugiere que el modelo está aprendiendo de manera efectiva a partir de los datos de entrenamiento. Sin embargo, se nota que el modelo no completó las 20 épocas inicialmente programadas, deteniéndose automáticamente debido al *early stopping* configurado en el proceso.

Al concluir el entrenamiento autorregresivo, se procede a la fase de *fine-tuning* supervisado (*instruct*), en la que el modelo se ajusta para cumplir con tareas específicas utilizando datos etiquetados. La configuración del modelo en esta etapa se ha mantenido idéntica a la del entrenamiento autorregresivo, con la única excepción del entrenador utilizado. En el ajuste fino supervisado (*Supervised Fine-Tuning, SFT*) [70], se ha adaptado un modelo de lenguaje preentrenado a una tarea concreta mediante el uso de datos etiquetados, a diferencia del entrenamiento inicial, que suele ser no supervisado.

Para llevar a cabo esta etapa, se ha empleado la clase *SFTTrainer* [70] de la biblioteca Hugging Face, utilizando el conjunto de datos *instruct* (preguntas y respuestas) previamente generado. Esta fase del entrenamiento se centra en ajustar el modelo a las especificaciones de la tarea, mejorando su capacidad para generar respuestas relevantes y precisas basadas en ejemplos validados. Los resultados obtenidos de esta etapa de *fine-tuning* se ilustran en la Figura 11.

Gráfica de pérdida del entrenamiento secuencial (parte supervisada)

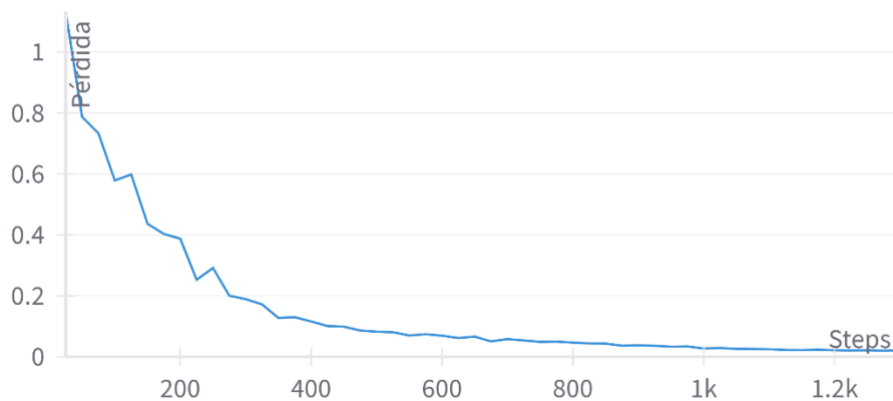


Figura 11: Seguimiento de la pérdida del entrenamiento supervisado secuencial

De manera similar al caso de entrenamiento autorregresivo, durante el entrenamiento supervisado la pérdida desciende de manera progresiva, un buen indicativo de que el modelo “aprende” de forma adecuada.

5.2.7 Resultados del entrenamiento secuencial

Una vez completada la fase de entrenamiento secuencial del modelo, es necesario evaluar su desempeño mediante una serie de pruebas. Para esta evaluación inicial, se seleccionaron aleatoriamente 10 preguntas con el objetivo de obtener una visión general de la capacidad del modelo. La configuración del modelo durante esta etapa incluyó una temperatura de 0,1 para asegurar una alta objetividad en las respuestas, así como un límite máximo de generación de hasta 500 tokens por respuesta.

A continuación, se presentan algunos de los resultados obtenidos en esta evaluación. Las respuestas del modelo se han clasificado según su calidad y precisión de la siguiente manera:

- Verde: Sentencias que demuestran una respuesta precisa y adecuada por parte del modelo, cumpliendo con los criterios de exactitud y relevancia establecidos.
- Amarillo: Respuestas que presentan deficiencias, tales como errores contextuales u ortográficos, que afectan la calidad de la respuesta, aunque aún muestran una comprensión básica del contenido.
- Rojo: Respuestas que no tienen correspondencia con la información contenida en los documentos originales, lo que indica la presencia de alucinaciones o afirmaciones incorrectas.

Esta categorización ha permitido una evaluación detallada del rendimiento del modelo y facilita la identificación de áreas para futuras mejoras. A continuación, se muestra una muestra representativa de los resultados obtenidos durante esta fase de prueba:

Pregunta 1: “¿Cuál fue la principal crítica de la señora diputada María Eulalia Guerra de Paz sobre el manejo del III Plan Canario de Igualdad de Oportunidades entre Mujeres y Hombres durante la sesión parlamentaria del 13 de marzo de 2008?”

Respuesta Esperada: “La principal crítica de la señora diputada María Eulalia Guerra de Paz fue que, a pesar de que el III Plan Canario de Igualdad de Oportunidades entre Mujeres y Hombres (2003-2006) ya había finalizado, aún se encontraba en un periodo de reflexión y evaluación, lo cual no consideraba adecuado. Argumentó que la evaluación del plan es fundamental para determinar la efectividad de las políticas activas de igualdad y debería haberse presentado para analizar y continuar haciendo esfuerzos en las áreas necesarias. Además, señaló la falta de inclusión de políticas activas de empleo para mujeres en la Estrategia de Empleo actual y criticó el uso de lenguaje masculino en la documentación.”

Respuesta Generada: “Durante la sesión parlamentaria del 13 de marzo de 2008, la señora diputada María Eulalia Guerra de Paz criticó duramente el manejo del III Plan Canario de Igualdad de Oportunidades entre Mujeres y Hombres. **Sus principales críticas incluyeron que el plan era tardío y que no se había cumplido con el artículo 21 del Derecho Canario a la Igualdad, que exige la igualdad en la esfera pública y privada. Además, señaló que el plan era copiado de un informe del Parlamento y que no se había realizado un análisis de la situación de igualdad en Canarias,** lo que hacía **suiposiciones** sobre la realidad de la situación social y laboral en la región. También expresó su preocupación por que el plan no abarcaba todas las áreas de la vida, como la educación y la sanidad, y que no estaba dirigido a la eliminación de la violencia de género, lo que consideraba un **significativo omisión**. Finalmente, señaló que no había sido aprobado en el Parlamento con la mayoría necesaria, lo que sugirió que no era adecuado para la realidad canaria.”

En este caso, se han identificado dos principales deficiencias. La primera está relacionada con la mezcla de contextos. El modelo muestra una tendencia a combinar información de diferentes eventos y leyes, generando respuestas que, aunque pueden ser coherentes en términos generales, no se ajustan con precisión al contexto específico de la pregunta. La segunda deficiencia está relacionada con errores ortográficos y alucinaciones. Se han detectado errores ortográficos y de contenido en las respuestas generadas, lo que afecta la calidad y precisión de las respuestas del modelo.

Pregunta 2: “¿Qué decisión tomó la señora diputada doña Gloria del Pilar Gutiérrez Arteaga respecto a su pregunta sobre los trámites exigidos por la Ley de Costas?”

Respuesta Esperada: “La señora diputada doña Gloria del Pilar Gutiérrez Arteaga decidió retirar su pregunta sobre los trámites exigidos por la Ley de Costas en relación con el caserío de Almáciga, Roque de las Bodegas y Tachero en la mañana del día de la sesión parlamentaria, el 28/03/2007.

Respuesta Generada: La señora diputada doña Gloria del Pilar Gutiérrez Arteaga **decidió retirar su pregunta sobre los trámites exigidos por la Ley de Costas en relación con el caserío de Almáciga, Roque de las Bodegas y Tachero en la mañana del 28/03/2007, antes de que fuera tratada en la sesión del parlamento. Este fue informado durante la sesión mismas por el señor presidente. No se proporciona un motivo específico para la retirada de la pregunta en el texto del diario.”**

Sin embargo, la segunda respuesta generada por el modelo coincide en un 100% con la respuesta esperada, lo que sugiere que el modelo es capaz de discernir eventos específicos y puntuales, como la retirada de una pregunta en una fecha concreta, a partir de su base de conocimiento adquirida. Este resultado indica que, en ciertos casos, el modelo es capaz de proporcionar respuestas exactas y detalladas basadas en información precisa y bien contextualizada.

Pregunta 3: “¿Cuál fue el resultado de la votación sobre la proposición de ley de Modificación de la Disposición Derogatoria de la Ley 11/2006, de la Hacienda Pública Canaria, llevada a cabo el 25 de enero de 2007?”

Respuesta Esperada: “La votación sobre la proposición de ley de Modificación de la Disposición Derogatoria de la Ley 11/2006, de la Hacienda Pública Canaria, resultó en 50 votos a favor, ninguno en contra y ninguna abstención. Esto indica que la proposición fue tomada en consideración de manera unánime por el Parlamento de Canarias en la sesión del 25 de enero de 2007.”

Respuesta Generada: “La **proposición de ley fue** aprobada con **44 votos a favor**, **ninguno en contra** y **3 abstenciones** durante la sesión del parlamento realizada el 25 de enero de 2007. Esta aprobación es necesaria para que se realicen las mismas medidas que se buscan legalizar en la proposición de ley, ya que las disposiciones derogatorias actuales impiden que sean aplicadas. **La firma del presidente del parlamento firmará la proposición de ley para que se envíe al Gobierno para su tramitación final.**”

A pesar de que el modelo demostró una alta precisión en la respuesta a la pregunta anterior, en esta instancia se observa una discrepancia significativa entre la respuesta generada y la respuesta esperada. La respuesta correcta para la pregunta 3 era que la proposición de ley había sido aprobada por unanimidad, con 50 votos a favor y sin abstenciones ni votos en contra. Sin embargo, el modelo ha generado una respuesta incorrecta, indicando que hubo 44 votos a favor, 3 abstenciones y que el presidente del Parlamento aún debía firmar la proposición para su envío al Gobierno.

La trayectoria general de este entrenamiento sigue la línea de estos tres casos. Es cierto que existen casos donde el modelo se acerca mucho, o incluso iguala a la calidad de las respuestas esperadas. Sin embargo, la gran mayoría de casos el modelo mantiene faltas de ortografía y datos falsos sobre las preguntas generadas, dando como resultado un asistente muy por debajo de las características necesarias.

5.2.8 Entrenamiento combinado

El entrenamiento combinado representa un enfoque distinto al del entrenamiento secuencial en la mejora de modelos de lenguaje. A diferencia del método secuencial, en el que el modelo es inicialmente entrenado en una tarea autorregresiva y luego sometido a un proceso de ajuste tipo *instruct*, el entrenamiento combinado integra ambas fases de manera iterativa y continua.

Este proceso iterativo se desarrolla mediante un ciclo repetitivo de entrenamiento autorregresivo seguido de una fase de ajuste supervisado de tipo *instruct*, lo cual permite al modelo ajustar y refinar sus parámetros de forma dinámica y simultánea a lo largo de todo el proceso de entrenamiento.

Este método presenta varias ventajas significativas sobre el enfoque secuencial tradicional. Al alternar entre las fases autorregresiva y supervisada, el modelo no solo se beneficia de la exposición continua a las tareas de modelado del lenguaje, sino que también se ajusta de manera efectiva a tareas específicas mediante datos etiquetados. Este ciclo iterativo fomenta un aprendizaje más profundo y robusto, ya que el modelo tiene la oportunidad de generalizar a partir de datos generados en una fase mientras se ajusta a las necesidades específicas de una tarea en la otra.

Siguiendo las mismas configuraciones que en los entrenamientos anteriores, el resultado del seguimiento de la pérdida en este caso es el mostrado en la Figura 12.

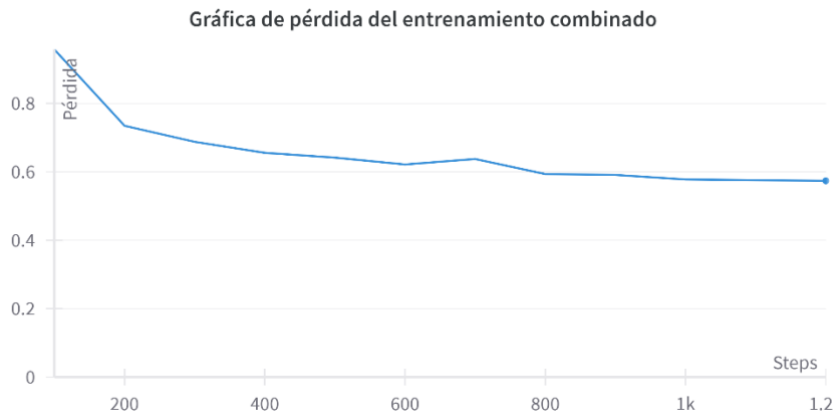


Figura 12: Seguimiento de la pérdida del entrenamiento combinado

A primera vista, se puede apreciar que la pérdida también desciende de manera adecuada durante el entrenamiento. Sin embargo, parece ser que disminuye sensiblemente menos que en el caso del entrenamiento secuencial. Si bien a priori podría parecer intuitivo que un menor valor de pérdida indica un mejor ajuste del modelo a los datos, en la práctica existen diversos factores que pueden llevar a que un modelo con una pérdida final más alta generalice mejor.

Por ello, para comprobar su efectividad, se desarrolla el mismo proceso anteriormente explicado.

5.2.9 Resultados del entrenamiento combinado

Como se menciona anteriormente, las preguntas no se repetirán. Por consiguiente, se ha seguido el mismo proceso de evaluación descrito previamente.

Respuesta Generada a la pregunta 1: “Durante la sesión parlamentaria del 13 de marzo de 2008, la señora diputada María Eulalia Guerra de Paz criticó el hecho de que el III Plan Canario de Igualdad de Oportunidades entre Mujeres y Hombres estaba a punto de caducar y aún no se le había emitido el acuerdo para su prórroga. Esta situación se consideraba alarmante porque implicaba que todas las políticas activas de empleo específicas para mujeres que estaban incluidas en el plan podrían cesar, afectando a las mujeres beneficiarias de estas medidas. Además, se criticó la falta de repercusión y ejecución efectiva del plan en los tres años de vigencia, lo que hizo que no se cumplieran los objetivos establecidos.”

Como primer acercamiento a la efectividad del modelo, se observa un avance significativo. El modelo comienza a ofrecer respuestas más concisas y precisas ante preguntas específicas, lo que indica una mejora en su capacidad para comprender y procesar la información relevante del contexto parlamentario. Esta evolución sugiere que el modelo es capaz de generar respuestas coherentes y adecuadas, reflejando una mayor precisión y una reducción en la tendencia a confundir o mezclar información.

Respuesta generada a la pregunta 2: “La señora diputada doña Gloria del Pilar Gutiérrez Arteaga menciona que la pregunta número 10, que estaba dirigida al **señor** consejero de Obras Públicas y Transportes y tenía relación con los trámites exigidos por la Ley de Costas, **fue retirada. No se proporciona información sobre la razón específica de la retirada.**”

Al igual que en la pregunta número 1, el modelo muestra una capacidad adecuada para responder de manera precisa y concisa a las preguntas, aunque aún presenta algunas faltas de ortografía. Esta tendencia indica que el modelo es capaz de comprender y generar respuestas coherentes y específicas, aunque requiere mejoras adicionales en la corrección ortográfica y gramatical.

Respuesta generada a la pregunta 3: “La proposición de ley de Modificación de la Disposición Derogatoria de la Ley 11/2006, de la Hacienda Pública Canaria, **fue rechazada durante la sesión del Parlamento de Canarias el 25 de enero de 2007. El resultado de la votación fue de 34 votos contra y 32 a favor, con tres abstenciones. Así, la proposición no obtuvo la mayoría necesaria para ser aprobada.**”

Finalmente, se ha revelado la principal debilidad de este enfoque de entrenamiento. El resultado obtenido en la pregunta número 3 ilustra la tendencia general del desempeño del modelo. Aunque las pruebas realizadas demuestran una mejora significativa en muchos aspectos, el modelo muestra dificultades recurrentes con consultas que implican fechas, eventos específicos o resultados de votaciones. Estas temáticas tienden a confundirse con el conocimiento adquirido durante el entrenamiento, lo que en muchos casos lleva a la generación de respuestas incorrectas o falacias.

Las tres pruebas realizadas siguen la línea general de la evaluación del entrenamiento secuencial. Se observa una notable mejoría en la calidad de redacción y contenido. Sin embargo, persisten lagunas que impiden un desempeño óptimo de la herramienta, especialmente en la precisión de respuestas sobre fechas, eventos específicos y resultados de votaciones.

5.2.10 Entrenamiento combinado y uso de memoria vectorial

Reconociendo que una de las principales deficiencias del modelo, incluso con la aplicación del método de entrenamiento combinado, es su capacidad limitada para manejar fechas o eventos específicos con precisión, se ha implementado de una memoria vectorial para abordar estas limitaciones y mejorar su desempeño en estas áreas.

Para ello, se ha utilizado ChromaDB para desarrollar una base de datos denominada *Session_diaries_embeddings*, que compila toda la base de conocimiento relacionada con las órdenes parlamentarias en formato de *embeddings*. Estos *embeddings* se generan mediante el modelo de OpenAI *text-embedding-ada-002* [71].

El propósito fundamental de esta implementación es enriquecer el contexto del *prompt* introducido al modelo, con el fin de mejorar la precisión en la recuperación de información relevante. Cuando se presenta un *prompt*, el sistema busca ampliar el contexto calculando la similitud entre este y los documentos de la base de datos utilizando *embeddings*. La similitud se determina a través de la métrica L2 (distancia euclidiana), en lugar de otras métricas comunes como la similitud del coseno o el producto

interno, con el objetivo de identificar los documentos más relevantes o relacionados con el contenido del *prompt*.

La distancia L2 se define como:

$$L2 = \sum_{i=1}^n (a_i - b_i)^2$$

- a_i y b_i : Son las componentes individuales de los vectores de características a y b , respectivamente. En el contexto de los *embeddings*, estos vectores representan la representación numérica de los documentos y el *prompt* en un espacio de características multidimensional.
- i : Indica el índice de la dimensión en el espacio de características. El valor de i varía desde 1 hasta n , donde n es el número total de dimensiones del vector.
- $(a_i - b_i)^2$: Calcula la diferencia al cuadrado entre las componentes correspondientes de los vectores a y b . Esta diferencia al cuadrado mide la discrepancia en cada dimensión individual entre los dos vectores.

Esta métrica es preferible en este caso por varias razones, como su interpretabilidad, eficiencia computacional, sensibilidad a la magnitud vectorial y por no requerir normalización.

Siguiendo la misma línea de experimentación, se han llevado a cabo una nueva serie de pruebas con el modelo mejorado mediante el uso de memoria vectorial. A continuación, se detallan los resultados obtenidos para la primera pregunta, así como una evaluación exhaustiva de estos resultados en comparación con el desempeño previo del modelo.

Respuesta Generada a la pregunta 1: “La principal crítica de la señora diputada María Eulalia Guerra de Paz fue que la evaluación del III Plan Canario de Igualdad de Oportunidades entre Mujeres y Hombres no estaba presente durante la sesión parlamentaria del 13 de marzo de 2008, aunque ella creía que era necesaria para efectuar una adecuada reflexión y valoración del plan. Ella destacó la importancia de analizar la evaluación para conocer el punto en que estaba el fallo y poder continuar con las acciones necesarias para mejorar la situación de igualdad entre mujeres y hombres en Canarias.”

En esta nueva fase de pruebas, la mejora en la calidad de las respuestas generadas es notable. La capacidad del modelo para abordar la pregunta con precisión y relevancia se ha incrementado considerablemente en comparación con los resultados previos del entrenamiento secuencial. En esta ocasión, el modelo no solo responde de manera más acertada, sino que también muestra una capacidad mejorada para utilizar el lenguaje de manera técnica y formal, adecuado al contexto parlamentario.

Respuesta Generada a la pregunta 2: “La señora diputada doña Gloria del Pilar Gutiérrez Arteaga retiró su pregunta durante la sesión del parlamento el 28 de marzo de 2007. No se tiene constancia de la razón específica por la que se retiró la pregunta.”

En esta segunda prueba, el modelo demuestra una notable mejoría en su capacidad para generar respuestas concisas y relevantes. La precisión en el contenido de la respuesta refleja un avance significativo respecto a las versiones anteriores del modelo. A diferencia de los resultados previos, en los que el modelo a menudo ofrecía respuestas vagas o erróneas, esta vez la respuesta es clara y ajustada a la pregunta formulada.

Una observación destacable es que, a pesar de haber establecido una longitud máxima de 500 tokens para las respuestas, el modelo logra ser más eficiente en la generación de respuestas concisas y directas, ajustándose adecuadamente a la necesidad de brevedad sin sacrificar la claridad. Este desempeño indica una mejora en la capacidad del modelo para filtrar y presentar información de manera efectiva, lo cual es un avance importante en términos de calidad de respuesta.

Respuesta Generada a la pregunta 3: “La votación sobre la proposición de ley de Modificación de la Disposición Derogatoria de la Ley 11/2006, de la Hacienda Pública Canaria, llevada a cabo el 25 de enero de 2007, tuvo un resultado de 50 votos a favor, ninguno en contra y ninguna abstención. Por lo tanto, la proposición de ley fue aprobada con estos votos.”

En el caso de la tercera pregunta, el modelo alcanza un nivel de especificidad que corresponde al detalle requerido por la consulta. La respuesta es precisa y correcta, reflejando adecuadamente el resultado de la votación sin errores. Este logro demuestra que el modelo es capaz de manejar preguntas con detalles concretos y específicos, como fechas y resultados de eventos, con una alta precisión.

Sin embargo, aunque la mejora en las respuestas generadas es evidente, el análisis de la totalidad del experimento sugiere que estas mejoras no son uniformes en todos los casos. La capacidad del modelo para generar respuestas correctas y bien fundamentadas se observa en aproximadamente un 60% de los casos, evidenciando una mejora general en comparación con el rendimiento anterior. No obstante, el 20% de las respuestas presentan errores ortográficos o discrepancias contextuales, mientras que otro 20% de las respuestas contienen alucinaciones o afirmaciones incorrectas, lo que indica que aún persisten debilidades significativas en el modelo.

Este patrón sugiere que, a pesar de los avances conseguidos mediante la implementación de memoria vectorial, persisten limitaciones inherentes al enfoque de entrenamiento secuencial. En particular, se plantea la hipótesis de que el modelo experimenta el fenómeno conocido como olvido catastrófico [65], en el cual el conocimiento adquirido durante el entrenamiento es parcialmente perdido o degradado durante el proceso de ajuste de nuevos datos. Esto se debe a la forma en que el entrenamiento secuencial puede llevar a la sobrescritura del conocimiento previo al añadir nueva información, un problema que se ha mitigado en parte con la técnica de entrenamiento combinado, pero que sigue siendo una consideración importante en el desarrollo de modelos de lenguaje.

5.2.11 Corrección con modelos GPT

En el proceso de ajuste y mejora de modelos de lenguaje, la implementación de memoria vectorial ha demostrado ser una técnica altamente prometedora. La capacidad de almacenar y recuperar información contextual relevante mediante *embeddings* ha facilitado notables avances en la calidad de las respuestas generadas por estos modelos. Sin embargo, surge una cuestión fundamental: ¿es la combinación del modelo reentrenado con memoria vectorial la que produce mejoras significativas, o

podría la memoria vectorial por sí sola ofrecer un rendimiento comparable utilizando el modelo base sin reentrenar?

Para explorar esta cuestión, se ha llevado a cabo un experimento con el objetivo de evaluar si el modelo base de Mistral, en combinación con la misma memoria vectorial utilizada anteriormente, puede alcanzar niveles de desempeño similares a los observados con el modelo reentrenado. Este análisis busca determinar si las mejoras observadas en el rendimiento se deben en su totalidad al proceso de reentrenamiento del modelo o si la implementación de memoria vectorial es una técnica suficiente para lograr un rendimiento óptimo por sí misma.

En este contexto, se seleccionan 60 preguntas del conjunto de prueba del dataset *instruct* previamente generado. Se ha desarrollado un *pipeline* que procesa automáticamente cada pregunta, generando un dataset en formato JSON que contiene los siguientes campos: respuesta original, respuesta generada, tiempo de ejecución y la pregunta misma. Este dataset es posteriormente analizado mediante los modelos GPT de la API de OpenAI, específicamente utilizando el modelo GPT-4 o [72].

El propósito principal de esta metodología es doble. En primer lugar, se busca evaluar si el modelo base de Mistral, cuando se apoya únicamente en la memoria vectorial, puede igualar el desempeño del modelo reentrenado.

Para llevar a cabo esta evaluación, se ha empleado el siguiente *prompt* para el análisis de las respuestas generadas en ambos escenarios:

system prompt: "Eres un evaluador que compara respuestas generadas con una respuesta modelo. Tu tarea es evaluar la precisión, relevancia y completitud de la respuesta generada en comparación con la respuesta modelo, y penalizar las respuestas cuyo idioma no coincida con el de la respuesta modelo."

user prompt: "Por favor, evalúa la respuesta generada comparándola con la respuesta modelo utilizando los siguientes criterios:

Evaluación de Contenido (1 a 10):

- 10: La respuesta generada es completamente precisa, relevante y completa, coincidiendo en gran medida con la respuesta modelo.
- 7-9: La respuesta generada es mayoritariamente precisa y relevante, con ligeras omisiones o errores menores.
- 4-6: La respuesta generada es parcialmente precisa y relevante, pero contiene errores notables o falta información clave.
- 1-3: La respuesta generada es mayoritariamente incorrecta, irrelevante o incompleta, o el idioma no coincide con el de la respuesta modelo.

El formato debe ser el siguiente:

Nota:

<La nota del 0 al 10>

Justificación:

<Pequeña justificación>

La respuesta generada a evaluar es la siguiente:

<Ejemplo de respuesta modelo y respuesta generada cargada en memoria>”
assistant prompt: <Ejemplo en memoria de nota y justificación>
user prompt: “Por favor, evalúa la respuesta generada comparándola con la respuesta modelo utilizando los siguientes criterios:
Evaluación de Contenido (1 a 10):
- 10: La respuesta generada es completamente precisa, relevante y completa, coincidiendo en gran medida con la respuesta modelo.
- 7-9: La respuesta generada es mayoritariamente precisa y relevante, con ligeras omisiones o errores menores.
- 4-6: La respuesta generada es parcialmente precisa y relevante, pero contiene errores notables o falta información clave.
- 1-3: La respuesta generada es mayoritariamente incorrecta, irrelevante o incompleta, o el idioma no coincide con el de la respuesta modelo.
El formato debe ser el siguiente:
Nota:
<La nota del 0 al 10>
Justificación:
<Pequeña justificación>
La respuesta generada a evaluar es la siguiente:
<Respuesta generada a evaluar del conjunto de datos de prueba>”
assistant prompt: <Nota y justificación dadas por GPT-4 o>

De manera similar a lo descrito en la subsección 4.1.4, si es necesario, se realiza la evaluación en dos pasos para garantizar una precisión exhaustiva. El primer paso se lleva a cabo utilizando el modelo experto (GPT-4 o), seguido de una revisión adicional por parte de un modelo auxiliar (GPT-3.5 turbo) para poder en el formato adecuado la evaluación (uso de expresiones regulares).

Los resultados obtenidos de la evaluación de las respuestas generadas por el modelo base de Mistral en comparación con el modelo reentrenado, utilizando memoria vectorial, reflejan diferencias significativas en términos de precisión y calidad de las respuestas. A continuación, se presentan las estadísticas descriptivas y una interpretación detallada de los datos, en la Tabla 1:

Modelo	Media	Mínimo	Máximo	Desviación Estándar
Mistral (base)	3,95	1	10	3,27
Mistral (reentrenado)	5,57	1	10	2,97

Tabla 1: Estadísticas sobre la evaluación de modelos de Mistral con memoria vectorial

La tabla anterior muestra que el modelo base de Mistral presenta una media de 3,95 con una desviación estándar de 3,27, mientras que el modelo reentrenado con memoria vectorial tiene una

media de 5,57 y una desviación estándar de 2,97. Estos datos sugieren una mejora notable en la calidad de las respuestas cuando se emplea el modelo reentrenado junto con la memoria vectorial.

A su vez, el análisis de la frecuencia de las notas revela que el modelo reentrenado con memoria vectorial obtiene una mayor concentración de calificaciones en los rangos más altos (notas de 7 a 10), en comparación con el modelo base, que presenta una mayor frecuencia de calificaciones bajas (notas de 1 a 3). Este patrón sugiere una tendencia general hacia una evaluación más positiva del modelo reentrenado, lo cual se puede apreciar en la Figura 13.

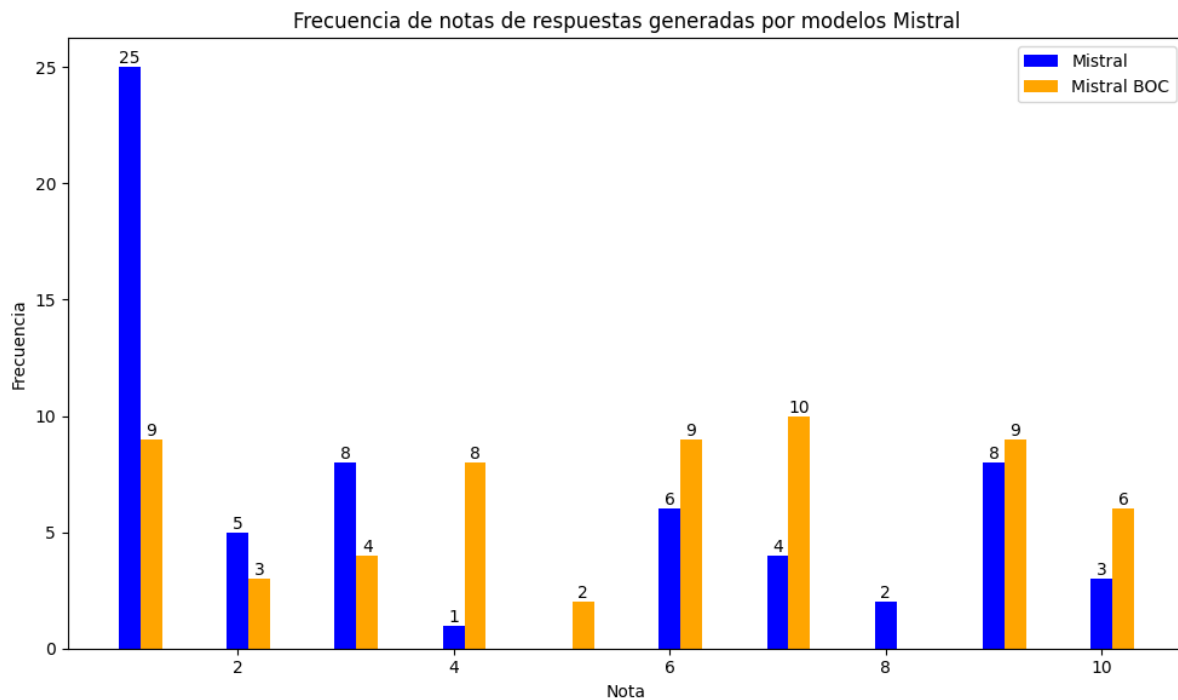


Figura 13: Frecuencia de notas de respuestas generadas por modelos Mistral

Los datos obtenidos confirman que el proceso de reentrenamiento combinado con el uso de memoria vectorial resulta en una mejora significativa en la capacidad del modelo para generar respuestas precisas y relevantes. La diferencia en las medias de las calificaciones y el patrón de frecuencias en las notas indican que el modelo reentrenado con memoria vectorial supera al modelo base en términos de desempeño general.

La implementación de la memoria vectorial no solo ha servido para mejorar la calidad de las respuestas, sino también ha demostrado ser una herramienta clave para contextualizar las respuestas generadas por el modelo. Esta técnica, en combinación con un modelo reentrenado, ofrece un enfoque robusto para abordar tareas complejas de generación de texto y recuperación de información.

En contraste, el modelo base de Mistral, a pesar de sus capacidades iniciales, muestra limitaciones evidentes en cuanto a precisión y relevancia de las respuestas cuando se le compara con la versión mejorada. La evidencia sugiere que el reentrenamiento del modelo, junto con el uso de memoria vectorial, es crucial para alcanzar un nivel óptimo de desempeño en tareas de evaluación de contenido y recuperación de información.

Por lo tanto, estos resultados subrayan la efectividad del enfoque de reentrenamiento combinado con memoria vectorial como una estrategia superior en la mejora de modelos de lenguaje, estableciendo

un precedente para futuras investigaciones y aplicaciones en el campo de la inteligencia artificial y el procesamiento del lenguaje natural.

Después de analizar el desempeño de los modelos de lenguaje, es importante considerar los costos asociados. La corrección automática cuesta aproximadamente 1,51 dólares, un gasto mínimo en comparación con los 40 dólares necesarios para crear el dataset etiquetado utilizando el costoso modelo GPT-4, dado que GPT-4 o aún no estaba disponible. Además, los *prompts* de corrección tienen un límite de hasta 1000 tokens, frente a las más de 600 órdenes de longitud variable del dataset.

5.2.12 Conclusión

El uso de modelos locales se destaca como una solución eficiente para la creación de asistentes personales, evitando la dependencia de APIs externas y las restricciones legales asociadas. En el desarrollo de este proyecto, se ha llevado a cabo todo el proceso de creación de datos necesarios para el asistente, incluyendo la persistencia final de los modelos. Al ser todo el despliegue local, se elimina la necesidad de depender de servicios externos, lo que proporciona mayor control y seguridad sobre los datos y su procesamiento.

Como se ha observado, el entrenamiento combinado, en el que en cada época se realiza una iteración con entrenamiento autorregresivo seguido de entrenamiento supervisado, evita el proceso de olvido catastrófico en modelos del lenguaje. A diferencia del enfoque secuencial, en el que primero se entrena todo el modelo de forma autorregresiva y luego se vuelve a entrenar de forma supervisada, el método combinado tiende a ofrecer resultados más precisos y concisos sobre los datos de entrenamiento.

Este tipo de entrenamiento, junto con el uso de memoria vectorial, supera significativamente a los modelos del lenguaje base de Mistral (en concreto el de 7 billones de parámetros). Sin embargo, aunque los resultados son aceptables, no alcanzan un nivel notable. La evaluación realizada sobre el modelo reentrenado no es suficiente para confiar plenamente en este modelo del lenguaje. Por ello, se propone como proyecto futuro utilizar aprendizaje por refuerzo durante el nuevo reentrenamiento.

El uso de aprendizaje por refuerzo podría ser beneficioso, ya que este enfoque permite que el modelo aprenda a tomar decisiones optimizadas a través de la interacción continua con el entorno de datos. A través de la retroalimentación obtenida de estas interacciones, el modelo puede ajustar sus parámetros de manera más precisa, mejorando así su rendimiento y capacidad para generar respuestas más relevantes y exactas.

5.3 Generación de resúmenes de sesiones parlamentarias

5.3.1 Introducción

Las sesiones parlamentarias suelen ser extensas, lo que implica que el tiempo necesario para su análisis o revisión puede ser considerablemente alto, representando un desafío significativo para aquellos que necesitan revisar estos documentos diariamente. Una solución efectiva a esta problemática es la generación automatizada de resúmenes, que permite condensar la información relevante en un formato más manejable y accesible.

Aunque existen varios modelos diseñados para la generación de resúmenes, en la actualidad esta tarea se ha delegado principalmente a grandes modelos del lenguaje como GPT. Estos modelos han demostrado una notable capacidad para comprender y sintetizar textos extensos, ofreciendo resúmenes precisos y coherentes.

Este experimento no solo permite evaluar la precisión y eficiencia de los modelos en condiciones ideales, sino también su desempeño en entornos con recursos limitados, proporcionando una visión más completa y práctica de las posibles aplicaciones de estos modelos en la generación de resúmenes parlamentarios.

El objetivo de este estudio es determinar si es posible generar resúmenes de calidad a partir de modelos libres, aprovechando técnicas de generación de texto sin modificaciones adicionales del modelo. De esta manera, se pretende evaluar si un modelo base, operando en un entorno con recursos limitados, puede producir resúmenes efectivos de sesiones parlamentarias, lo que representaría una solución práctica y eficiente para el análisis de documentos legislativos.

5.3.2 Preprocesado de los datos

Para llevar a cabo el preprocesado de los datos, se han seguido dos procesos principales que son fundamentales para garantizar la calidad de los resúmenes generados por el modelo. Dado que los modelos generativos, como los de lenguaje natural, operan bajo restricciones de tamaño en sus *prompts* (secuencias de entrada), es esencial gestionar adecuadamente estos límites para evitar errores en la generación de texto. Es importante tener en cuenta que, en muchos casos, el tamaño del *prompt* incluye tanto el contenido de entrada como el de salida, lo que puede influir en el rendimiento del modelo.

Dado que los textos correspondientes a las sesiones parlamentarias varían en longitud y muchos superan con creces el límite de tokens permitido por el modelo, se ha tomado la decisión de segmentar el texto en unidades más manejables. El primer paso en este proceso consistió en fragmentar los documentos almacenados en la base de datos MongoDB previamente creada (colección "raw"). Cada entrada del campo "text" de esta base de datos se ha dividido en una lista de subcadenas, donde cada subcadena contiene un máximo de 2.000 tokens.

Para la tokenización de estos textos, se ha utilizado *tiktoken* [73] de OpenAI, una librería seleccionada por su robustez y eficacia en el procesamiento de lenguaje natural. Tiktoken es ampliamente aplicable en diferentes contextos de tokenización y resulta adecuado para manejar grandes volúmenes de datos textuales. Adicionalmente, se ha implementado un solapamiento de 100 tokens entre los fragmentos de texto. Este solapamiento implica que cada nuevo fragmento comienza con los últimos 100 tokens

del fragmento anterior, lo que ayuda a preservar el contexto y mejorar la coherencia entre los fragmentos consecutivos.

Para ilustrar el proceso de segmentación con solapamiento, se presenta el siguiente ejemplo. Supongamos que tenemos la frase:

" El proyecto se desarrolla en un entorno colaborativo."

El objetivo es dividir esta frase en subcadenas de 5 palabras cada una, aplicando un solapamiento de 2 palabras entre los fragmentos. El proceso se realiza de la siguiente manera:

1. **Primera subcadena:** "El proyecto se desarrolla en"
2. **Segunda subcadena:** "desarrolla en un entorno"
3. **Tercera subcadena:** "un entorno colaborativo."

En este ejemplo, cada subcadena se construye a partir de la frase original con un solapamiento de 2 palabras entre cada fragmento consecutivo, lo cual garantiza la preservación del contexto entre los fragmentos y mejora la coherencia del texto al procesarlo en unidades independientes.

Una vez realizada la segmentación, se ha realizado una limpieza de datos, que es un paso crucial para garantizar que los datos sean interpretados correctamente por los modelos generativos. Este proceso de limpieza incluye la eliminación de espacios adicionales, tabulaciones y saltos de línea innecesarios, lo cual es esencial para evitar errores y mejorar la calidad de los resultados generados.

Además, se añade una frase al inicio de cada primer fragmento del documento para proporcionar contexto adicional. Esta frase especifica la fecha de la orden y la fecha de realización de la sesión parlamentaria en el siguiente formato:

"Esta sesión del parlamento se realizó el {date}. {text}"

Este formato asegura que cada fragmento de texto incluya información contextual relevante desde el comienzo, facilitando una interpretación más precisa del contenido por parte del modelo.

El preprocesado descrito se aplica a cada uno de los documentos de la colección original, y los resultados se almacenan en una nueva colección en mongo denominada `fragmented_preprocessing`. Esta nueva colección contiene los documentos procesados de manera individual, listos para ser utilizados en etapas posteriores del análisis.

5.3.3 Elección de la librería llama-cpp-python

Para llevar a cabo la experimentación con modelos cuantizados en el contexto de la generación de resúmenes de sesiones parlamentarias, se ha seleccionado la librería llama-cpp-python [29]. Esta elección responde a varias consideraciones técnicas y prácticas relacionadas con las necesidades específicas del proyecto.

En primer lugar, la elección de modelos cuantizados se basa en la premisa de que, para tareas de resumen de textos, los modelos no requieren una especialización extrema en comparación con otras aplicaciones más complejas. Los modelos cuantizados ofrecen una serie de ventajas que los hacen particularmente adecuados para este tipo de tareas.

La cuantización es una técnica de optimización que convierte los pesos del modelo de representaciones de alta precisión, como las de 32 bits, a representaciones de menor precisión, como las de 8 bits. Este proceso de reducción de precisión tiene dos efectos principales: disminuye significativamente los requisitos de almacenamiento del modelo y mejora la velocidad de procesamiento sin comprometer de manera significativa su capacidad para generar respuestas precisas. Esta característica es particularmente relevante en aplicaciones que demandan rapidez y eficiencia, como es el caso de la generación de resúmenes a partir de grandes volúmenes de texto.

Una ventaja adicional de los modelos cuantizados es su capacidad para operar en entornos con recursos limitados. En el contexto de esta experimentación, donde se busca evaluar la generación de resúmenes de calidad sin recurrir al *fine-tuning* del modelo, la opción de utilizar modelos de hasta más de 50 billones de parámetros en un entorno reducido resulta ser especialmente adecuada. La librería llama-cpp-python facilita el uso de estos modelos grandes en un entorno eficiente y manejable, proporcionando acceso a modelos avanzados sin necesidad de recursos computacionales extensivos.

En pocas palabras, a través de la librería llama-cpp-python, se pueden implementar y ejecutar modelos de lenguaje cuantizados en el formato GGUF (*GPT-Generated Unified Format*), lo cual permite aprovechar las capacidades de modelos extensos y bien entrenados mientras se mantiene un entorno de ejecución eficiente y accesible. Esta metodología no solo optimiza el rendimiento del modelo para la generación de resúmenes, sino que también asegura que los resultados sean de alta calidad, lo que es crucial para el éxito de la tarea.

5.3.4 Elección de modelos a estudiar

En el proceso de evaluación del rendimiento para la generación de resúmenes de sesiones parlamentarias, se han seleccionado cuatro modelos en su versión cuantizada a 5 bits, basándose en la documentación oficial que indica que estos modelos representan un equilibrio óptimo entre calidad de salida y eficiencia en el uso de recursos. Los modelos seleccionados son:

- **Llama2-7B-chat** [74]: Este modelo es una variante de Llama 2 diseñado para mantener conversaciones de tipo diálogo, con un tamaño de 4.78 GB en su versión cuantizada a 5 bits.
- **Llama2-13B-chat** [75]: Otra versión del modelo Llama 2, con una capacidad de 13 mil millones de parámetros, optimizada para tareas de conversación y con un tamaño de 9.23 GB en formato cuantizado.
- **Mistral-7B-v0.2-instruct** [76]: Un modelo de Mistral, también cuantizado a 5 bits, entrenado específicamente para seguir instrucciones y realizar tareas basadas en comandos detallados, con un tamaño de 5.12 GB.
- **Mistral-8x7B-v0.1-instruct** [77] Este modelo representa una de las versiones más grandes de Mistral con un total de 56 mil millones de parámetros, diseñado para tareas instruccionales, con un tamaño de 32.2 GB en su versión cuantizada.

Es importante señalar que, aunque han sido lanzados nuevos modelos como Llama 3 [12] desde la fecha de realización de este estudio, se optó por estos modelos cuantizados debido a que en el momento de la experimentación sus implementaciones no estaban disponibles. La selección de estos modelos se basa en su balance documentado entre calidad y eficiencia, y en su idoneidad para las tareas de generación de resúmenes de acuerdo con los recursos disponibles en ese período.

En cuanto a las diferencias entre los modelos, se observa una clara variación en el número de parámetros. Por ejemplo, el modelo Llama2-13B-chat cuenta con 13 mil millones de parámetros, mientras que el Mistral 8x7B-v0.1-instruct dispone de 56 mil millones de parámetros. Esta diferencia en la escala de los modelos se refleja en sus capacidades y requisitos de recursos, siendo los modelos más grandes generalmente más potentes, pero también más exigentes en términos de memoria y capacidad computacional.

5.3.5 Experimentación y benchmarking

Para evaluar el rendimiento en la generación de resúmenes de sesiones parlamentarias, se han seleccionado aleatoriamente 25 documentos de la base de datos `fragmented_preprocessing`. Esta selección aleatoria se realizó por cuestiones prácticas de tiempo, con el objetivo de abarcar una muestra representativa del conjunto de datos.

Cada documento ha sido sometido a dos pruebas de generación de resúmenes: una utilizando el 50% del contenido original y otra utilizando el 25% del contenido original. Estos dos niveles de resumen permiten evaluar la capacidad de los modelos para condensar la información en diferentes proporciones.

Es importante destacar que el tiempo de ejecución de cada modelo aumenta significativamente en función de sus pesos, incluso cuando están cuantizados. Esta consideración es crucial para el análisis posterior de los resultados, ya que influye en el tiempo total dedicado al experimento y en la comparación entre los modelos evaluados.

Para facilitar un benchmarking fluido y automatizado, se desarrolló un proyecto que implementa un flujo de trabajo optimizado. Este proceso de benchmarking registra automáticamente los resultados de cada modelo en un archivo CSV estructurado, incluyendo diversas métricas y detalles sobre la generación de resúmenes. El archivo CSV contiene información esencial como el tiempo de ejecución, el porcentaje de resumen, el modelo utilizado, el costo asociado (especialmente relevante para modelos como GPT), metadatos del documento, el contenido resumido y el número de palabras tanto del texto original como del resumen.

Este enfoque sistemático no solo permite una evaluación rigurosa del rendimiento de los modelos en la tarea de generación de resúmenes, sino que también facilita la comparación objetiva entre diferentes técnicas y configuraciones. Así, se obtiene una comprensión clara y cuantitativa de las capacidades y limitaciones de los modelos evaluados en condiciones de prueba controladas. A través de este proceso, se busca no solo medir la eficacia de los modelos en generar resúmenes precisos y coherentes, sino también proporcionar *insights* valiosos para su aplicación en la práctica de análisis de sesiones parlamentarias.

El flujo de experimentación diseñado para la generación automática de resúmenes de sesiones parlamentarias y su posterior evaluación mediante técnicas de benchmarking sigue una serie de pasos meticulosamente estructurados, descritos a continuación:

1. **Definición de Parámetros Iniciales:** En esta etapa, se introducen los parámetros fundamentales para el experimento, incluyendo el modelo de lenguaje a utilizar, la versión específica del modelo y el porcentaje de contenido que se desea resumir (50% o 25%). Estos

parámetros determinan las configuraciones iniciales para el proceso de generación de resúmenes.

2. **Extracción de Documentos:** Se accede a la base de datos `fragmented_preprocessing` para leer los documentos seleccionados aleatoriamente. A partir de esta base de datos, se extrae el campo `text`, que contiene una lista de fragmentos del texto original, previamente segmentados con un solapamiento para mantener la coherencia del contexto entre fragmentos.
3. **Generación de resúmenes por fragmento:** Cada fragmento extraído se procesa utilizando el modelo de lenguaje seleccionado. El modelo genera un resumen para cada fragmento basado en el porcentaje de entrada especificado. Este paso implica pasar cada segmento del texto a través del modelo para obtener un resumen que condense la información relevante del fragmento.
4. **Combinación de fragmentos en un resumen completo:** Los resúmenes generados para cada fragmento son combinados para formar un único resumen que represente el documento original en su totalidad. Este paso asegura que la información consolidada refleja el contenido completo del documento, manteniendo la coherencia entre los fragmentos individuales.
5. **Recopilación de métricas:** Se recopilan una serie de métricas clave para evaluar el desempeño de los modelos. Entre estas métricas se incluyen el tiempo de ejecución del modelo, el costo asociado (particularmente para modelos que requieren pago por uso como GPT), el número de palabras tanto en el texto original como en el resumen, y otros datos relevantes. Estas métricas permiten una evaluación cuantitativa del rendimiento del modelo en función de la eficiencia y el costo.
6. **Almacenamiento y registro de resultados:** Los datos obtenidos durante el proceso de generación de resúmenes se registran en un archivo CSV. Este archivo CSV está estructurado con cabeceras que reflejan las métricas y la información del resumen, facilitando así un análisis posterior de los resultados. Este formato asegura que los datos sean accesibles y organizados para su evaluación y comparación.

Para evitar la repetición de procesos y garantizar una mayor eficiencia, se ha desarrollado un flujo de trabajo automatizado. Se ha adoptado un enfoque de desarrollo basado en la flexibilidad, implementando patrones de diseño orientados a objetos, como el *Factory Method* [78]. Este patrón de diseño ofrece varias ventajas en el contexto de la experimentación, incluyendo modularidad, escalabilidad y flexibilidad.

5.3.6 Análisis de los resultados obtenidos

En este análisis, evaluamos los resúmenes generados por diferentes modelos de lenguaje para 25 documentos de prueba seleccionados aleatoriamente. El conjunto de datos resultante incluye los siguientes campos para cada documento: texto original, texto resumido, número de palabras del texto original, número de palabras del texto resumido, modelo utilizado, coste asociado, porcentaje de resumen aplicado y metadatos correspondientes del documento.

El primer paso consiste en dividir este conjunto de datos en dos categorías: resúmenes de referencia, generados por GPT-4o, y resúmenes de evaluación, producidos por otros modelos. Para evaluar la calidad de los resúmenes, se utiliza la métrica ROUGE, que incluye las medidas ROUGE-1, ROUGE-2,

ROUGE-L y ROUGE-Lsum. Estas métricas se calculan para cada documento con el fin de comparar la similitud entre los resúmenes de referencia y los resúmenes de evaluación. A continuación, se describe la contribución de cada métrica a este estudio:

- **ROUGE-1:** Mide la similitud entre los resúmenes generados y los resúmenes de referencia basándose en la frecuencia de ocurrencia de palabras individuales (unigramas). Calcula el porcentaje de palabras unigramas en común entre el resumen generado y el resumen de referencia, dividiendo esto por el número total de unigramas en el resumen de referencia.
- **ROUGE-2:** Considera la similitud basada en secuencias de dos palabras consecutivas (bigramas). Calcula el porcentaje de bigramas en común entre el resumen generado y el resumen de referencia, dividido por el número total de bigramas en el resumen de referencia.
- **ROUGE-L:** Mide la similitud considerando la secuencia más larga común entre el resumen generado y el resumen de referencia. Se calcula como la longitud de la subsecuencia común más larga dividida por la longitud del resumen de referencia.
- **ROUGE-Lsum:** Es una métrica que considera la media del ROUGE-L sobre todos los n-gramas en lugar de solo el n-grama más largo. A diferencia de ROUGE-L, que considera solo la subsecuencia más larga, ROUGE-Lsum promedia los valores ROUGE-L para todos los n-gramas en el resumen de referencia.

Una vez seleccionadas las métricas, se procede a realizar un análisis del desempeño de cada modelo, cuyos resultados se presentan en la Tabla 2.

Resultados Llama2-13b-chat (cuantizado a 5 bits)						
Métrica	25% Promedio	25% Máximo	25% Mínimo	50% Promedio	50% Máximo	50% Mínimo
ROUGE-1	0,55	0,66	0,24	0,59	0,68	0,35
ROUGE-2	0,29	0,38	0,10	0,32	0,48	0,12
ROUGE-L	0,32	0,43	0,24	0,34	0,58	0,24
ROUGE-Lsum	0,39	0,52	0,24	0,44	0,58	0,24

Resultados Llama2-7b-chat (cuantizado a 5 bits)						
Métrica	25% Promedio	25% Máximo	25% Mínimo	50% Promedio	50% Máximo	50% Mínimo
ROUGE-1	0,48	0,65	0,04	0,48	0,67	0,04
ROUGE-2	0,24	0,35	0,02	0,25	0,37	0,02
ROUGE-L	0,27	0,45	0,04	0,28	0,51	0,04
ROUGE-Lsum	0,34	0,54	0,04	0,37	0,55	0,03

Resultados Mistral-7B-instruct (cuantizado a 5 bits)						
Métrica	25% Promedio	25% Máximo	25% Mínimo	50% Promedio	50% Máximo	50% Mínimo
ROUGE-1	0,55	0,68	0,15	0,59	0,68	0,47
ROUGE-2	0,30	0,45	0,05	0,32	0,43	0,20
ROUGE-L	0,35	0,52	0,15	0,36	0,53	0,24
ROUGE-Lsum	0,39	0,52	0,15	0,45	0,58	0,31

Resultados Mistral-8x7B-instruct (cuantizado a 5 bits)						
Métrica	25% Promedio	25% Máximo	25% Mínimo	50% Promedio	50% Máximo	50% Mínimo
ROUGE-1	0,57	0,68	0,21	0,58	0,70	0,26
ROUGE-2	0,32	0,48	0,09	0,33	0,50	0,11
ROUGE-L	0,36	0,53	0,18	0,37	0,54	0,21

ROUGE-Lsum	0,40	0,54	0,19	0,42	0,55	0,23
------------	------	------	------	------	------	------

Tabla 2: Métricas de Rouge para modelos cuantizados

Es notable que el mejor modelo es el que tiene más pesos, ya que Mistral-8x7b muestra un rendimiento superior en la generación de resúmenes en comparación con Mistral-7B. Sin embargo, es importante considerar tanto el tamaño como la cantidad de parámetros de cada modelo al evaluar la mejora en el rendimiento. La mejora no es proporcional al incremento de los parámetros. Mistral-7B ofrece una relación costo-beneficio muy favorable, al proporcionar un rendimiento competitivo con una fracción de los parámetros de Mistral-8x7b. Por lo tanto, al elegir entre estos modelos, es crucial considerar tanto el rendimiento absoluto como la eficiencia en el uso de recursos.

Por otro lado, los modelos Llama2 han tenido un rendimiento inferior incluso utilizando el mismo número de parámetros (7 mil millones). Esto indica que no son adecuados para tareas de producción de resúmenes según las métricas ROUGE.

Otro aspecto crucial a la hora de ofrecer un servicio es el tiempo de ejecución. Para evaluar el rendimiento de los modelos en términos de tiempo de ejecución, se empleó una GPU NVIDIA GeForce 4090 con aproximadamente 24 GB de VRAM. A continuación, se presentan los tiempos de ejecución promedio, mínimo y máximo para cada modelo al generar resúmenes correspondientes al 25% y al 50% del texto original.

Dado que la generación de resúmenes generalmente implica el procesamiento de textos largos, es de esperar que el tiempo de ejecución sea elevado en los modelos locales debido a los recursos limitados de la infraestructura experimental. En la Tabla 3, se muestra la distribución del tiempo de ejecución (en segundos) para cada modelo estudiado.

Modelo	Porcentaje	Media (s)	Mínimo (s)	Máximo (s)
GPT-3.5turbo	0,25	15,59	0,75	40,87
	0,50	16,40	0,934	43,16
Llama2-13b-chat	0,25	247,99	4,23	699,97
	0,50	241,90	4,312	686,11
Llama2-7b-chat	0,25	187,61	5,66	725,80
	0,50	224,11	5,20	747,01
Mistral-7b-instruct-v0.2	0,25	164,81	1,690	489,13
	0,50	159,93	1,33	517,92
Mistral-8x7b-instruct-v0.1	0,25	647,07	36,01	1857,26
	0,50	627,97	35,15	1624,03

Tabla 3: Tiempos de ejecución de generación de resúmenes

Como era de esperar, el tiempo promedio de ejecución de la API de GPT es considerablemente bajo en comparación con los otros modelos, tanto para generar un resumen del 25% como del 50% del texto original. Además, la variabilidad en los tiempos es mínima, con tanto los tiempos mínimos como máximos siendo bajos, lo que sugiere una ejecución rápida y consistente. Es importante tener en

cuenta que GPT-3.5 turbo es un servicio de API, lo cual implica que no utiliza recursos locales, pero conlleva un costo por su uso.

En contraste, los modelos Llama2 destacan por ser significativamente más lentos, lo que los convierte en una opción menos favorable cuando la eficiencia en tiempo es crucial. Estos modelos muestran una variabilidad considerable, aunque no muy dispar entre sí, especialmente notable dado que los tiempos mínimos y máximos no difieren significativamente entre los modelos de 13 y 7 billones de parámetros.

Por otro lado, el modelo Mistral-7B se muestra como una opción altamente competitiva, ofreciendo un rendimiento robusto con tiempos de ejecución razonables. Esto lo convierte en una alternativa atractiva para escenarios en los que tanto los recursos como el tiempo son factores críticos. Aunque no puede igualar la rapidez de GPT-3.5 turbo en su configuración actual, Mistral-7B tiene el potencial de ofrecer un rendimiento comparable en un entorno con recursos equivalentes. Este equilibrio entre capacidad y eficiencia lo convierte en una opción adecuada para quienes buscan un modelo potente sin depender de una API externa, especialmente en contextos con limitaciones de recursos.

Finalmente, la versión Mistral-8x7B, debido a su gran tamaño, ha demostrado ser la menos eficiente en términos de tiempo de ejecución, con tiempos que pueden alcanzar hasta los 10 minutos. Esto la convierte en una opción menos favorable para situaciones en las que el tiempo de procesamiento es una preocupación importante.

La dependencia de una API, como la de OpenAI, ofrece ventajas claras en términos de rapidez y calidad en la generación de resúmenes. Sin embargo, el costo es una consideración crucial, especialmente para aplicaciones en sectores de alto uso como la administración pública y el ámbito parlamentario. Este tipo de servicios se emplea ampliamente para resumir documentos, lo que conlleva un uso intensivo y recurrente. En este experimento, los costos asociados con el uso de la API de GPT-3.5 turbo se detallan en la Tabla 4, desglosados por documento:

Modelo	Costo promedio (\$)	Costo Mínimo (\$)	Costo Máximo (\$)	Coste total acumulado (\$)
GPT-3.5 turbo	0,004456	0,000082	0,012136	0,222819

Tabla 4: Costes asociados a la generación de resúmenes con la API de OpenAI

El costo puede escalar rápidamente con el uso intensivo. Supongamos que una agencia gubernamental procesa 10.000 documentos mensuales, con un tamaño variable de entre 500 y 2000 tokens:

- **Costo mínimo mensual:** 10.000 documentos * \$0,000082 = \$0,82
- **Costo máximo mensual:** 10.000 documentos * \$0,012136 = \$121,36
- **Costo promedio mensual:** 10.000 documentos * \$0,004456 = \$44,56

Aunque la API de GPT-3.5 de OpenAI proporciona una solución rápida y de alta calidad para la generación de resúmenes, su uso intensivo puede resultar en costos significativos, especialmente en aplicaciones gubernamentales y parlamentarias que manejan grandes volúmenes de documentos.

Finalmente, se ha optado por desarrollar una métrica personalizada para evaluar la capacidad de los modelos de generación de resúmenes en cuanto al cumplimiento del porcentaje de resumen solicitado

por el usuario. Es esencial que un resumen respete el límite de longitud establecido: un documento resumido no puede superar el porcentaje de tokens definido respecto al texto original. A partir de esta premisa, se ha evaluado si los modelos cumplen con este criterio, verificando en cada porcentaje de resumen solicitado, de los 25 documentos de prueba, cuántos se ajustan a la longitud deseada.

Para desarrollar esta métrica, se ha calculado la longitud original en tokens del documento, se ha aplicado el porcentaje de resumen correspondiente y se ha establecido el número ideal de tokens para el resumen. Además, se han definido rangos de error de 100 tokens para medir el cumplimiento de la longitud. Los resultados se detallan en la Tabla 5.

Modelo	Porcentaje	Documentos que cumplen la métrica
GPT-3.5 turbo	0,25	11/25
	0,50	6/25
Llama2-13b-chat	0,25	8/25
	0,50	6/25
Llama2-7b-chat	0,25	7/25
	0,50	5/25
Mistral-7b-instruct-v0.2	0,25	13/25
	0,50	6/25
Mistral-8x7b-instruct-v0.1	0,25	5/25
	0,50	6/25

Tabla 5: Número de tokens en resúmenes generado por modelos

GPT-3.5 muestra un rendimiento razonable en la métrica del 25%, pero su desempeño disminuye notablemente al solicitar un resumen del 50%. Esto sugiere que el modelo tiende a generar resúmenes que a menudo exceden la longitud permitida, especialmente cuando se requiere un resumen más extenso.

De manera similar, Llama2-13b-chat presenta un mejor desempeño en la métrica del 25% en comparación con el 50%, aunque su rendimiento general es inferior al de GPT-3.5, indicando una menor precisión en la generación de resúmenes ajustados al tamaño requerido. Por otro lado, Llama2-7b muestra un rendimiento comparable al de Llama2-13b-chat, con una tendencia similar hacia un mejor cumplimiento en resúmenes más cortos, pero con una precisión en términos de longitud que aún necesita mejoras.

En cuanto a los modelos Mistral, se observa una tendencia similar. Mistral-7B destaca en la métrica del 25%, cumpliendo con el porcentaje de longitud en más de la mitad de los documentos. Sin embargo, su desempeño en la métrica del 50% también disminuye, aunque sigue siendo comparable al de otros modelos. Por el contrario, a pesar de su mayor tamaño, Mistral-8x7B tiene un rendimiento inferior en el cumplimiento de la métrica del 25% y su desempeño en la métrica del 50% es comparable al de los otros modelos, sin llegar a sobresalir.

En conclusión, considerando todas las métricas evaluadas, Mistral-7B se perfila como una opción prometedora al equilibrar la precisión en la longitud del resumen con la calidad del contenido generado. Seguido de Mistral-8x7B, cuyo rendimiento parece ser ideal para tareas de generación de resúmenes, a excepción del tiempo de ejecución, lo cual es un problema solucionable.

5.3.7 Autocorrección de resúmenes con modelos GPT

Además de calcular las métricas previamente descritas, se ha llevado a cabo una evaluación de los resúmenes generados por los cuatro modelos estudiados, utilizando el resumen generado por GPT-3.5 turbo como referencia para la autocorrección. Esta experimentación sigue el mismo enfoque que el apartado 4.1.11, utilizando el módulo ModelCorrection del proyecto para comparar y ajustar la calidad de los resúmenes generados por cada modelo.

A continuación, se presentan los resultados obtenidos de la evaluación, que se resumen en la Tabla 6:

Modelo	Media	Mínimo	Máximo	Desviación Estándar
Llama2-13B	4,54	1	10	2,67
Llama2-7B	4,34	2	10	2,51
Mistral-7B	6,08	2	10	2,53
Mistral-8x7B	7,24	4	10	1,72

Tabla 6: Estadísticas sobre la evaluación de resúmenes de Mistral y Llama2

En términos generales, Mistral-8x7B se destaca como el modelo con el mejor desempeño, con una media de 7.24 y una desviación estándar baja de 1.72, indicando que ofrece resúmenes de alta calidad con una buena consistencia. Por otro lado, Mistral-7B también muestra un buen desempeño con una media de 6.08 y una desviación estándar de 2.51, evidenciando que produce resúmenes de alta calidad, aunque con algo más de variabilidad en las calificaciones. Los modelos Llama2-13B y Llama2-7B muestran un rendimiento inferior, con medias de 4.54 y 4.34 respectivamente, y desviaciones estándar de 2.67 y 2.51, lo que refleja una menor calidad en sus resúmenes y una mayor variabilidad en las evaluaciones.

Por otro lado, se realiza un estudio de la distribución de las calificaciones en torno a los 4 modelos. Este resultado queda reflejado en la Figura 14.

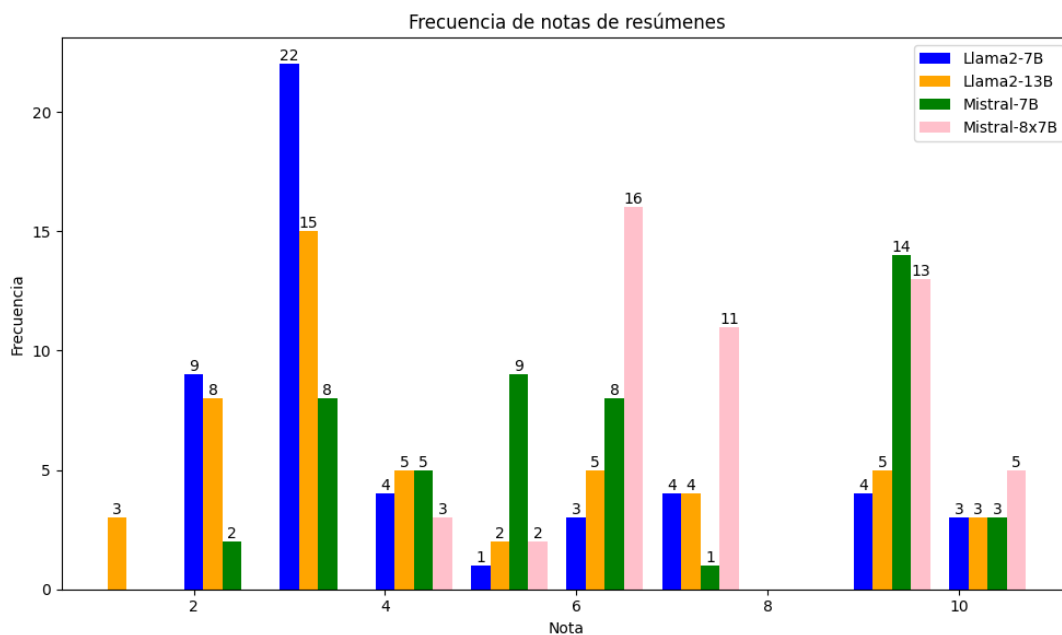


Figura 14: Frecuencia de notas de resúmenes generados por modelos Mistral y Llama2

Al analizar la distribución de calificaciones, se observa que Mistral-8x7B obtiene la mayor cantidad de calificaciones en el rango superior, destacando con una alta frecuencia de calificaciones de 9 y una notable cantidad de calificaciones de 10. Este modelo tiende a generar resúmenes de alta calidad, aunque con cierta variabilidad. Mistral-7B también muestra una alta concentración de calificaciones en el rango superior, con una significativa cantidad de calificaciones de 9, aunque sufre de algunas inconsistencias, evidenciadas por las calificaciones más bajas. En contraste, Llama2-13B y Llama2-7B tienen una mayor frecuencia de calificaciones bajas, reflejando una calidad menos consistente en sus resúmenes. Llama2-7B, en particular, presenta una mayor concentración en calificaciones bajas y menos representación en los rangos superiores, indicando un desempeño inferior en comparación con los modelos de Mistral.

Después de analizar el desempeño de los modelos de lenguaje, es importante considerar los costos asociados. La corrección automatizada de resúmenes tiene un coste aproximado de 11.27 dólares. Este gasto es mínimo en comparación con el coste del dataset etiquetado de preguntas y respuestas del chatbot, dado que se utiliza GPT-4 o en lugar de GPT-4.

5.3.8 Resumen del benchmarking a través de un cuadro de mando

Finalmente, para ofrecer una visión general del benchmarking realizado, se ha creado un cuadro de mando en Power BI [41]. Este cuadro de mando está organizado en dos vistas principales, diseñadas para facilitar la comparación y análisis de los datos de los modelos de inteligencia artificial evaluados.

La primera vista del cuadro de mando se centra en la comparación general entre los modelos Mistral-7B, Mixtral-8x7B, Llama2-7B, Llama2-13B y GPT-3.5. Se evalúan aspectos clave como el tiempo de respuesta, el costo de los experimentos y la capacidad de los modelos para cumplir los objetivos planteados. También se incluyen otros parámetros relevantes para evaluar la precisión y coherencia de los resúmenes generados, permitiendo una comparación directa del desempeño general de los modelos.

La segunda vista está dedicada a la comparación de los valores ROUGE, una métrica estándar que mide la calidad de los resúmenes comparando la superposición de n-gramas entre el resumen generado y un conjunto de resúmenes de referencia. En esta vista, se comparan los modelos Mistral-7B, Mistral-8x7B, Llama2-7B y Llama2-13B con los resúmenes generados por GPT-4, permitiendo así evaluar la similitud de los resúmenes generados respecto a los textos originales.

Para facilitar el análisis de los resultados obtenidos en el benchmarking, el cuadro de mando en Power BI se organiza en diferentes grupos que permiten una evaluación detallada y comparativa de los modelos.

El grupo de acciones incluye herramientas interactivas que permiten al usuario personalizar la visualización y comparar los datos de manera efectiva.

- **Filtros:** Los filtros permiten seleccionar uno o más modelos y ajustar el porcentaje de resumen (50%, 25% o ambos). Estos filtros proporcionan flexibilidad en el análisis, facilitando la comparación de aspectos específicos según las necesidades del usuario.
- **Botón de Navegación:** Un botón permite cambiar entre las diferentes vistas del cuadro de mando, facilitando la exploración de los datos y la comparación entre los modelos.

A su vez, el grupo de gráficos presenta los datos de rendimiento de los modelos a través de representaciones visuales que facilitan el análisis comparativo.

- **Diagramas de Barras:** Los datos se muestran mediante diagramas de barras. Este formato ayuda a identificar rápidamente diferencias y similitudes entre los modelos.
- **Organización de Gráficos:** En ambas vistas, los gráficos están organizados en un formato que prioriza la información más relevante, con el gráfico más importante en la parte superior izquierda y el menos relevante en la parte inferior derecha.

Por último, el grupo de datos totales muestra las métricas agregadas que resumen el rendimiento global de los modelos en cada vista.

- **Primera Vista:** Se presentan datos globales sobre variables como el tiempo y el costo de los experimentos, junto con métricas clave desarrolladas para transformar los datos en formatos más legibles y calcular el promedio de palabras resumidas.
- **Segunda Vista:** Se incluyen métricas específicas para identificar el modelo con el mayor valor ROUGE, facilitando la evaluación de cuál modelo genera resúmenes más precisos y similares a los textos originales.

Esta descripción puede verse plasmada en la Figura 15, y Figura 16 ilustrando las diferentes vistas del cuadro de mando. Estas imágenes destacan la estructura y organización de los datos, proporcionando una visión clara y detallada del rendimiento de los modelos evaluados.



Figura 15: Vista 1 del cuadro de mandos del benchmarking de resúmenes



Figura 16: Vista 2 del cuadro de mandos del benchmarking de resúmenes

5.3.9 Conclusión

El *benchmarking* llevado a cabo sobre diversos modelos de generación de resúmenes ha proporcionado una visión integral de sus capacidades y limitaciones en cuanto a la calidad del resumen, el tiempo de ejecución y el cumplimiento de la longitud especificada.

El análisis evidencia que el modelo Mistral-8x7B se presenta como una opción altamente adecuada debido a su capacidad para generar resúmenes de alta calidad y su cumplimiento con las longitudes de

salida especificadas. A pesar de que GPT-3.5 turbo supera a Mistral-8x7B en términos de rapidez gracias a su infraestructura dedicada, este último ofrece un rendimiento robusto que resulta más accesible para aplicaciones locales y puede ser una alternativa más costo-efectiva en escenarios de uso intensivo.

Asimismo, se destaca que Mistral-7B demuestra un desempeño notable en comparación con su sucesor, Mistral-8x7B. A pesar de contar con casi 50 mil millones de parámetros menos, este modelo logra un rendimiento más que adecuado en su versión cuantizada, evidenciando que modelos con menor complejidad pueden alcanzar niveles de calidad competitivos en tareas de generación de resúmenes.

Los resultados obtenidos en este estudio demuestran que para muchas tareas de gestión documental no es necesario recurrir a modelos de gran escala o alta especialización. Modelos cuantizados, incluso en versiones de 5 bits, pueden lograr un desempeño notable en la generación de resúmenes, lo que sugiere que es posible obtener resultados eficaces con recursos limitados. Además, los problemas asociados con el tiempo de ejecución pueden ser mitigados mediante mejoras en la infraestructura de hardware, lo cual refuerza la idea de que es factible alcanzar resultados significativos con una inversión moderada en recursos.

De manera concluyente, el estudio demuestra que es posible equilibrar calidad, eficiencia y costo en la generación de resúmenes, ofreciendo opciones viables tanto para aplicaciones locales como para escenarios de alta demanda.

5.4 Clasificación de Documentos Parlamentarios

5.4.1 Introducción

Este apartado presenta la etapa final del proyecto, enfocada en la clasificación de documentos parlamentarios utilizando las etiquetas de EuroVoc [20]. El objetivo es optimizar el uso de hardware mediante técnicas de procesamiento del lenguaje natural (PLN) más accesibles y eficientes.

La clasificación de documentos parlamentarios es esencial para organizar sesiones legislativas a partir de un esquema estandarizado. Se ha elegido trabajar con el primer nivel de etiquetas de EuroVoc, proporcionando una categorización general que facilita una implementación eficiente adaptada a las necesidades del proyecto.

Para esta tarea se ha empleado BERT (*Bidirectional Encoder Representations from Transformers*) [47], un modelo más ligero y accesible que permite realizar la clasificación con aproximadamente 16 GB de VRAM, un recurso común en entornos de desarrollo actuales. BERT, con su capacidad para entender el contexto bidireccional del texto, ofrece una solución efectiva para la clasificación de documentos parlamentarios en un entorno con limitaciones de recursos.

En conclusión, esta sección demuestra cómo BERT, con requisitos de hardware modestos, puede ser una herramienta viable para la clasificación de documentos parlamentarios, avanzando hacia técnicas de PLN más accesibles y eficientes.

5.4.2 Selección de los datos de entrenamiento etiquetados

A pesar de disponer de un dataset compuesto por sesiones parlamentarias, es importante señalar que estos datos carecen de etiquetas, lo que impide su uso directo en modelos de clasificación para el entrenamiento. Para abordar esta limitación, se llevó a cabo una búsqueda exhaustiva con el objetivo de encontrar un dataset que ofreciera características legislativas y políticas similares a las de las sesiones parlamentarias, y que además contara con una estructura de etiquetado adecuada para las necesidades del estudio.

Durante este proceso de búsqueda, se evaluaron diversas fuentes de datos disponibles en el ámbito de la documentación legislativa. Finalmente, se optó por el dataset “MultiEURLEX” [79] disponible en la plataforma Hugging Face, reconocido por su amplia variedad de datos en el contexto legislativo. Esta selección se basa en la robustez del dataset y en su capacidad para proporcionar datos etiquetados que son relevantes para la tarea de clasificación.

El dataset “MultiEURLEX” en su versión española comprende aproximadamente 65.000 documentos, que incluyen leyes, debates, cartas y otros textos relacionados con la Unión Europea. Este dataset ha sido anotado con conceptos (etiquetas) provenientes del sistema EuroVoc [20], el cual es un sistema de clasificación de documentación parlamentaria europea que organiza los conceptos en ocho niveles de profundidad. Cada documento en el dataset está etiquetado con uno o más conceptos de EuroVoc, siguiendo una jerarquía que asigna etiquetas a documentos sin extenderse a ancestros o descendientes del concepto principal.

Para los fines de este estudio, se ha decidido utilizar únicamente las etiquetas del primer nivel de EuroVoc. Este nivel proporciona una categorización general y manejable, adecuada para el tipo de clasificación requerida en esta investigación. Las etiquetas de niveles superiores (del 3 al 8) no se han considerado relevantes para el contexto específico de la clasificación de documentos parlamentarios en esta etapa del proyecto.

5.4.3 Análisis de los datos

Para comprender de manera más profunda el contenido y las características del dataset utilizado en este estudio, es esencial desglosar la estructura de cada uno de sus elementos. A continuación, se presenta la estructura típica de los datos contenidos en el dataset:

```
{
    "celex_id": str,
    "text": str,
    "labels": list[int],
}
```

Cada elemento del dataset está compuesto por los siguientes campos:

- **celex_id**: Un identificador único proporcionado por EuroVoc para cada documento, que actúa como una referencia distintiva en el sistema de clasificación.
- **text**: El contenido textual del documento jurídico, el cual puede abarcar diferentes tipos de textos legislativos como leyes, debates, cartas, entre otros.

- `labels`: Una lista de números enteros que representan las etiquetas asignadas al documento, indicando los conceptos relevantes según el sistema de clasificación EuroVoc.

El dataset está dividido en tres conjuntos de datos: 52.785 documentos para entrenamiento, 5.000 documentos para validación y 5.000 documentos para prueba. Esta partición permite evaluar el modelo de manera efectiva y asegurar que el rendimiento sea medido de forma justa en datos no vistos previamente.

Antes de proceder con el entrenamiento del modelo, es necesario realizar un análisis exhaustivo de los datos para asegurar una preparación adecuada. Uno de los aspectos más relevantes a considerar es la distribución de las etiquetas. El primer nivel de la taxonomía EuroVoc está constituido por un total de 21 etiquetas. Como se muestra en la Figura 17, se observa un desbalanceo significativo en la frecuencia de aparición de las etiquetas. Algunas etiquetas aparecen en hasta 50.000 documentos, mientras que otras están presentes en tan solo 1.000 documentos.

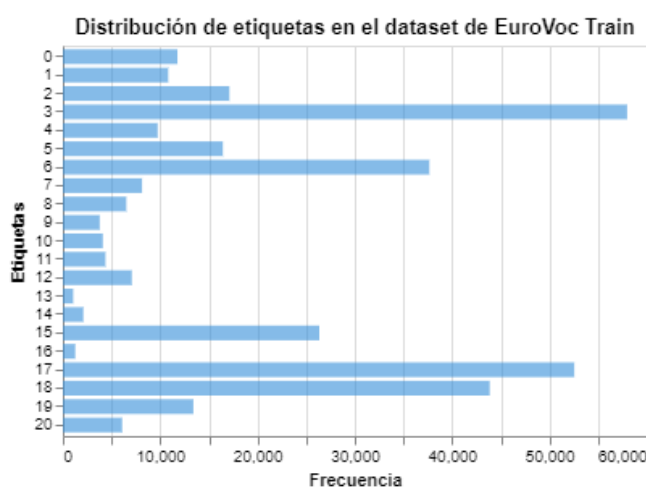


Figura 17: Distribución de etiquetas del primer nivel de la taxonomía EuroVoc del dataset

Este desbalanceo en la distribución de las etiquetas debe ser tenido en cuenta durante el entrenamiento de los modelos, ya que puede afectar la capacidad del modelo para aprender y generalizar de manera efectiva. En particular, un desbalance en la frecuencia de las etiquetas puede llevar a un sesgo en la predicción del modelo, favoreciendo las etiquetas más comunes en detrimento de las menos frecuentes. Por lo tanto, se han considerado técnicas de ajuste de clase y estrategias de manejo del desbalanceo para abordar estas desigualdades y mejorar la capacidad del modelo para tratar con todas las etiquetas de manera equitativa.

Este desbalanceo en la distribución de las etiquetas puede presentar un desafío importante durante la fase de inferencia final. La disparidad en la frecuencia de las etiquetas puede llevar a que el modelo se incline hacia las clases más comunes, lo que puede resultar en un rendimiento deficiente para las etiquetas menos frecuentes. En particular, este desbalanceo puede causar que el modelo sea menos preciso en la clasificación de documentos pertenecientes a las categorías menos representadas, afectando negativamente la calidad global de las predicciones.

Por otro lado, es crucial analizar la distribución de palabras y caracteres por documento. Como se muestra en la Figura 18, la mayoría de los documentos tienen entre 1.000 y 50.000 palabras. En cuanto a los caracteres, la mayoría oscila entre 50.000 y 350.000. Este análisis es fundamental para

comprender la longitud y complejidad de los documentos, aspectos que pueden influir en la configuración de hiperparámetros y en el diseño del modelo.

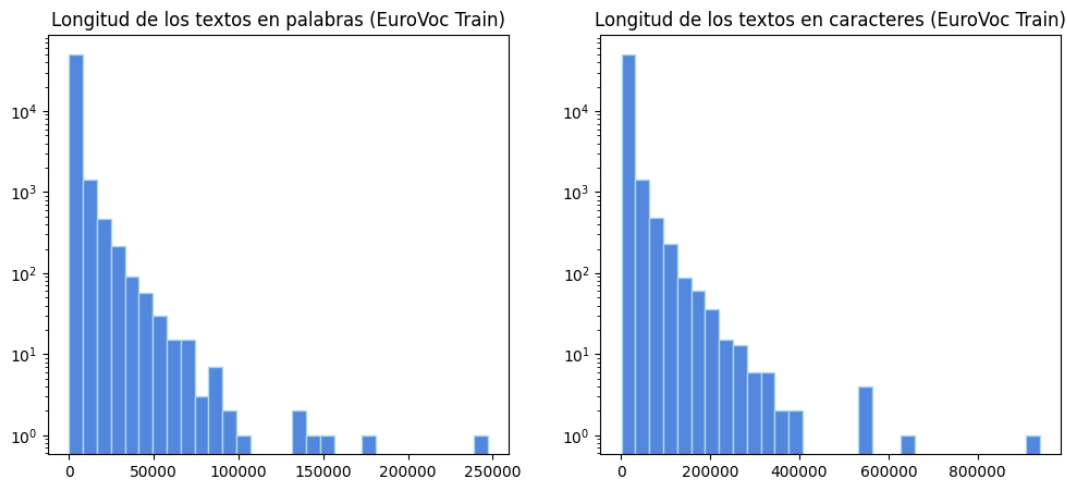


Figura 18: Distribución del número de palabras y caracteres en el dataset del EuroVoc

Es importante destacar que las distribuciones presentadas corresponden al conjunto de entrenamiento, si bien las del conjunto de validación y test muestran patrones muy similares. Para evitar redundancias, se presentan únicamente las distribuciones del conjunto de entrenamiento.

5.4.4 Preprocesado de los datos

El preprocesado de los datos se ha llevado a cabo mediante la implementación de una clase en Python denominada `CustomDataset`, la cual está específicamente diseñada para manejar y preparar los datos destinados a modelos de aprendizaje automático. Este proceso de preprocesado sigue una serie de pasos clave para garantizar que los datos estén en el formato adecuado para el entrenamiento del modelo.

En primer lugar, se extrae el campo `text` de cada registro del conjunto de datos. Este texto se somete a un proceso de limpieza que incluye la eliminación de espacios innecesarios y la normalización del formato, con el fin de asegurar consistencia en el procesamiento posterior.

A continuación, las etiquetas asociadas a los textos, contenidas en el campo `labels`, se convierten en vectores *one-hot*. En su formato original, las etiquetas se representan como una lista de enteros, pero para el entrenamiento del modelo, estas deben ser transformadas a un formato *one-hot*. Por ejemplo, una etiqueta original como `[2, 4, 5]` se convierte en un vector *one-hot* `[0, 1, 0, 1, 1]`, donde cada posición del vector corresponde a la presencia o ausencia de una categoría específica.

Posteriormente, los textos preprocesados se tokenizan utilizando el tokenizador de BERT, proporcionado a través de la clase `CustomDataset`. Este proceso de tokenización genera un diccionario con los siguientes componentes:

- `input_ids`: Identificadores de los tokens generados.
- `attention_mask`: Máscara de atención que indica qué tokens deben ser atendidos durante el procesamiento.

- `token_type_ids`: Identificadores que diferencian entre diferentes tipos de tokens en el texto.
- `targets`: Etiquetas transformadas en el formato one-hot.

El resultado final de cada registro del dataset tiene la siguiente estructura:

```

{
  'input_ids': <identificador de los tokens>,
  'attention_mask': <máscara de atención>,
  'token_type_ids': <tipo de token>,
  'targets': <etiquetas>
}

```

El dataset original presentaba un desbalance significativo, lo cual puede afectar negativamente el rendimiento del modelo. Un dataset desbalanceado implica que algunas etiquetas están representadas con mucha más frecuencia que otras, lo que puede llevar a que el modelo aprenda a predecir principalmente las etiquetas más comunes y pase por alto las menos frecuentes. Esto disminuye la capacidad del modelo para generalizar correctamente sobre las diferentes clases presentes en el dataset.

Para mitigar este problema, se ha implementado un método de balanceo por *batches* mediante el uso de un muestreador (*sampler*) equilibrado, que ajusta la frecuencia de las etiquetas en cada lote de datos durante el entrenamiento. Este proceso consta de varios pasos clave:

1. **Cálculo de Frecuencias de Etiquetas:** Se suman las etiquetas de todas las muestras para obtener la frecuencia de cada clase en el dataset. Esto permite identificar cuántas veces aparece cada etiqueta. La frecuencia de una etiqueta c se representa como:

$$f_c = \sum_{i=1}^N y_{i,c}$$

donde f_c es la frecuencia de la clase c , $y_{i,c}$ es el valor binario que indica la presencia de la clase c en la muestra i , y N es el número total de muestras.

2. **Determinación de Pesos de Clase:** Para equilibrar las frecuencias, se calculan los pesos inversos de las clases. Esto se hace tomando el inverso de la frecuencia de cada clase, de manera que las clases menos representadas tengan un peso mayor. Se añade un pequeño valor constante ϵ para evitar divisiones por cero. La fórmula para calcular el peso de una clase c es:

$$w_c = \frac{1}{f_c + \epsilon}$$

donde w_c es el peso de la clase c .

3. **Asignación de Pesos a Muestras:** Se asignan pesos a cada muestra del dataset en función de las etiquetas que contiene. Esto se logra multiplicando las etiquetas de cada muestra por los pesos de clase correspondientes. El peso de una muestra i , denotado como w_i , se calcula como:

$$w_i = \sum_{c=1}^C y_{i,c} * w_c$$

donde C es el número total de clases.

4. **Creación del *Sampler* Equilibrado:** Finalmente, se crea un *sampler* de muestreo aleatorio ponderado utilizando los pesos de las muestras, usando la clase `WeightedRandomSampler` [80]. Este *sampler* asegura que cada lote de datos esté balanceado en términos de etiquetas, seleccionando las muestras de manera que las clases menos frecuentes tengan una mayor probabilidad de ser incluidas. El muestreador selecciona las muestras con una probabilidad proporcional a sus pesos w_i .

Este método de balanceo es esencial para mejorar la capacidad del modelo de aprendizaje automático para generalizar sobre todas las clases del dataset, intentando mitigar así los efectos negativos del desbalanceo en la inferencia final.

Una vez implementado el balanceo, los datos se utilizan utilizando los *samplers* equilibrados, con un tamaño de *batch* de 32. Este enfoque garantiza que cada lote de datos esté balanceado en términos de etiquetas, mejorando así la capacidad del modelo para aprender de manera efectiva y generalizar correctamente sobre todas las clases presentes en el dataset.

5.4.5 Modelos de BERT elegidos para el entrenamiento de BERT con datos del EuroVoc

Para esta sección del proyecto, se selecciona un conjunto de modelos BERT, algunos de los cuales fueron reentrenados específicamente para esta tarea. Los modelos elegidos fueron:

1. **bert-base-multilingual-cased** [81]: Modelo preentrenado en los 104 idiomas más importantes según Wikipedia, utilizando un objetivo de modelado de lenguaje enmascarado (MLM). Este modelo se preentrenó en un gran corpus de datos multilingües de manera autodirigida, lo que significa que se entrenó en textos sin etiquetar, utilizando un proceso automático para generar entradas y etiquetas a partir de esos textos.
2. **roberta-base-bne** [82]: Modelo basado en RoBERTa para el idioma español, preentrenado con el corpus español más grande conocido hasta la fecha, con un total de 570GB de texto limpio. El corpus fue compilado a partir de rastreos web realizados por la Biblioteca Nacional de España entre 2009 y 2019.
3. **distilbert-base-spanish-uncased** [83]: DistilBERT es un modelo de transformadores más pequeño y rápido que BERT, preentrenado en el mismo corpus de manera autodirigida, utilizando el modelo base de BERT como maestro. Fue reentrenado con un corpus en español.
4. **bert-base-spanish-wwm-uncased** [84]: BERT es un modelo BERT entrenado en un gran corpus español. Tiene un tamaño similar al BERT-Base y se entrenó con la técnica de enmascaramiento de palabras completas. También se presentan algunos resultados en *benchmarks* en español, comparando BERT con *Multilingual BERT* y otros modelos que no son basados en BERT.

Aunque estos modelos parecen distintos, todos están basados en la arquitectura de BERT, lo que asegura que comparten una base común en términos de capacidad de comprensión del lenguaje natural.

5.4.6 Entrenamiento de BERT con datos de taxonomía EuroVoc

Para adaptar los modelos BERT a la tarea específica de clasificación multiclase de textos parlamentarios, se han realizado modificaciones en la arquitectura base de cada modelo. A continuación, se describe la arquitectura adaptada y las configuraciones utilizadas, destacando que este tipo de arquitectura y configuraciones similares fueron las que ofrecieron los mejores resultados.

Todos los modelos base deben modificarse en su parte final, ya que sus tareas primarias están destinadas a otros tipos de problemas. Por tanto, es necesario ajustar la salida para una clasificación multietiqueta.

La arquitectura adaptada es la siguiente:

- **Modelo Base:** La arquitectura comienza con un modelo base de BERT, que se encarga de procesar las secuencias de texto y generar representaciones contextuales de los mismos.
- **Pre-clasificador:** A la salida del modelo BERT se añade una capa lineal (pre-clasificador) que transforma las representaciones generadas por BERT a un espacio dimensional adecuado para la tarea de clasificación.
- **Capa de Tanh:** A continuación, se aplica una función de activación Tanh. Esta función transforma las salidas del pre-clasificador para mejorar la capacidad del modelo de aprender representaciones más complejas y no lineales, facilitando así el proceso de clasificación.
- **Capa de Dropout:** Se introduce una capa de *dropout*, con una tasa de 0.1, para prevenir el sobreajuste durante el entrenamiento. Esta capa ayuda a mejorar la generalización del modelo.
- **Clasificador:** La arquitectura incluye una capa lineal adicional (clasificador) que proyecta las representaciones transformadas al número de etiquetas de la clasificación multiclase (en este caso, 21 etiquetas).
- **Función de Activación:** Finalmente, se aplica una función de activación sigmoide para obtener las probabilidades de pertenencia a cada clase, permitiendo así una clasificación multietiqueta.

Para el entrenamiento del modelo adaptado para la tarea de clasificación multiclase, se emplea el optimizador Adam, reconocido por su eficiencia y efectividad en el ajuste de modelos complejos. El proceso de entrenamiento se extiende a lo largo de 30 épocas, proporcionando al modelo el tiempo necesario para aprender y ajustar sus parámetros a través de múltiples iteraciones sobre el conjunto de datos.

Durante el entrenamiento, se define una función de métricas personalizada que permite evaluar el rendimiento del modelo tanto en el conjunto de entrenamiento como en el de validación. Esta función de métricas abarca varios componentes clave que aseguran una evaluación integral del modelo:

- **Función de pérdida:** Se utiliza la pérdida de error cuadrático medio (*Mean Squared Error* o MSE) como métrica principal para cuantificar la diferencia entre las predicciones del modelo y las etiquetas verdaderas. La función MSE mide la magnitud de los errores de predicción, penalizando las discrepancias entre las salidas del modelo y las etiquetas de referencia.
- **Funciones de precisión, recall y f1:** Para evaluar el rendimiento del modelo en términos de precisión, *recall* y F1, se emplean las funciones implementadas por *scikit-learn*. En este contexto, las predicciones se mapean a través de un umbral de 0.5, donde cualquier valor de predicción igual o superior a 0.5 se considera una etiqueta positiva (1), y cualquier valor inferior a 0.5 se mapea a una etiqueta negativa (0).

Estas métricas son más adecuadas que el *accuracy* en tareas de clasificación multiclase con múltiples etiquetas debido a los siguientes motivos. El *accuracy* mide la proporción de etiquetas globalmente correctas, mientras que, en esta tarea, un documento puede estar etiquetado con múltiples categorías. Por lo tanto, esta métrica no captura correctamente la capacidad del modelo para asignar todas las etiquetas relevantes. Por otro lado, puede ser engañoso en datasets desbalanceados, donde clases minoritarias pueden ser ignoradas. Las métricas de precisión, *recall* y F1 Score proporcionan una evaluación más precisa en estos casos.

Una vez definidas las métricas es turno de crear el proceso de entrenamiento del clasificador. En los procesos de entrenamiento de modelos, es común utilizar una clase *Trainer* personalizada que gestiona de manera integral el ciclo de entrenamiento. Esta clase es responsable de coordinar la iteración a través de las épocas de entrenamiento y de validar el modelo en un conjunto de datos de validación después de cada época. Además, calcula métricas de desempeño clave como precisión, *recall*, *F1-score* y pérdida, utilizando una implementación específica para estas métricas. A continuación, se detallan las funciones y aspectos más relevantes de esta clase, que son fundamentales para entender el desempeño del entrenamiento:

- **Inicializador** (`_init_`): Este método configura todos los parámetros necesarios para el entrenamiento, incluyendo el modelo, el optimizador, los datos de entrenamiento y validación, las rutas para guardar los *checkpoints*, el número de épocas, el dispositivo de cómputo (GPU o CPU), entre otros. También inicializa objetos para el reporte de métricas y el cálculo de las métricas de desempeño.
- **Entrenamiento del modelo** (`_train_model`): Esta función coordina el proceso de entrenamiento del modelo. Se encarga de iterar a través de los datos de entrenamiento en lotes, calcular las salidas del modelo, evaluar la pérdida en relación con las etiquetas verdaderas, realizar la retropropagación del error para ajustar los pesos del modelo utilizando el optimizador, y actualizar las métricas de precisión, *recall*, *F1-score* y pérdida durante el entrenamiento.

Además de estos métodos principales, la clase *Trainer* incluye varias funciones auxiliares que facilitan el flujo de trabajo del entrenamiento:

- **Preparación de datos** (`_prepare_data`): Prepara los datos para ser enviados al dispositivo de cómputo adecuado, ya sea GPU o CPU.
- **Binarización de salidas** (`_binarize_outputs`): Convierte las salidas del modelo en valores binarios (0 o 1) mediante un umbral determinado.
- **Cálculo de pérdida** (`_compute_loss`): Calcula la pérdida entre las predicciones del modelo y las etiquetas verdaderas utilizando una función de pérdida específica para la tarea.
- **Retropropagación del error** (`_backpropagate_loss`): Realiza la retropropagación del error para ajustar los pesos del modelo en función de la pérdida calculada.
- **Actualización de métricas** (`_update_metrics`): Actualiza las métricas de desempeño, como precisión, *recall*, *F1-score* y pérdida, durante las fases de entrenamiento y validación.
- **Reporte de métricas** (`_report_metrics`): Genera informes periódicos sobre el desempeño del modelo, mostrando las métricas calculadas durante el entrenamiento y la validación.
- **Guardado de checkpoints** (`_save_ckp`): Guarda el estado del modelo y del optimizador en archivos de *checkpoint*, lo que permite recuperar el modelo en un estado previo si es necesario.

El método principal de esta clase es `train()`, que orquesta todo el flujo de entrenamiento del modelo. Este método gestiona la ejecución de las etapas de entrenamiento y validación, asegurando que todas las funciones auxiliares trabajen en conjunto para optimizar el desempeño del modelo.

Una vez completado el entrenamiento, los resultados obtenidos se visualizan en la Figura 19, que muestra el desarrollo de la pérdida durante el entrenamiento y la validación.

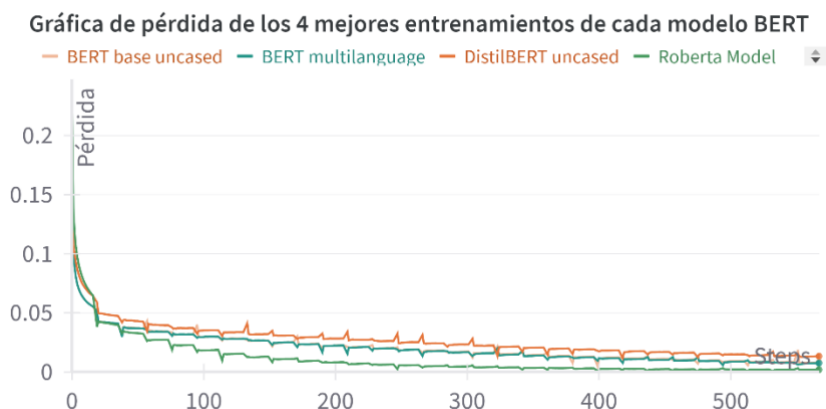


Figura 19: Gráfica de pérdida de los 4 mejores entrenamientos de cada modelo BERT

Al observar la gráfica de pérdida durante el entrenamiento, se puede ver que los modelos muestran una adecuada adaptación a los datos, con la pérdida tendiendo a estabilizarse a lo largo del tiempo. A partir de esta gráfica, el modelo RoBERTa parece ser el más prometedor, ya que alcanza una estabilización más temprana en comparación con los otros modelos.

Para obtener una evaluación definitiva del desempeño de cada modelo y elegir el mejor, es necesario llevar a cabo un testeo riguroso en un conjunto de datos de prueba. Este paso es fundamental para asegurar que los modelos no solo se ajusten bien a los datos de entrenamiento, sino que también puedan generalizar eficazmente a nuevos datos.

5.4.7 Pruebas de evaluación de los modelos

Para llevar a cabo una evaluación exhaustiva de los modelos entrenados, se ha desarrollado una clase personalizada en Python llamada `Tester`. Esta clase tiene como objetivo principal generar predicciones a partir de los datos de prueba, calcular métricas de rendimiento clave, y proporcionar un informe detallado sobre el desempeño de los modelos en términos de precisión, *recall*, y otras métricas. A continuación, se explica el propósito y funcionamiento de cada uno de los métodos de la clase.

- **Inicializador** (`__init__`): Este método es el constructor de la clase. Aquí se inicializan los atributos necesarios para la evaluación, como el modelo a evaluar, el cargador de datos de prueba, el dispositivo de cómputo (GPU o CPU), y el umbral para convertir las salidas del modelo en valores binarios. También se preparan variables para almacenar las métricas de rendimiento, tales como precisión, *recall*, precisión y *F1-score*.
- **Actualización de Métricas** (`_update_metrics`): Este método estático añade las predicciones del modelo y las etiquetas verdaderas a las listas acumulativas, preparándolas para el análisis posterior.

Además de estos métodos principales, la clase `Tester` incluye varias funciones auxiliares que facilitan el flujo de trabajo del entrenamiento:

- **Preparación de Datos** (`_prepare_data`): Este método procesa cada lote de datos para enviarlos al dispositivo de cómputo adecuado. Convierte los datos a los tipos correctos y transfiere los datos al dispositivo (GPU o CPU) para su evaluación. Aquí también se obtienen las etiquetas del conjunto de datos de prueba.
- **Recopilación de Predicciones y Etiquetas** (`_collect_targets_outputs`): Este método estático añade las predicciones del modelo y las etiquetas verdaderas a las listas acumulativas, preparándolas para el análisis posterior.
- **Promedio de Métricas** (`_average_metrics`): Este método calcula el valor promedio de las métricas de rendimiento acumuladas durante la evaluación. El promedio se obtiene dividiendo el total acumulado de cada métrica por el número total de lotes en el conjunto de datos de prueba.

Para finalizar con el desarrollo de la infraestructura de test, el método `test()` orquesta el proceso de evaluación del modelo.

Una vez definida la clase `Tester`, se procede a evaluar los modelos con el objetivo de obtener una visión clara de su rendimiento. Considerando métricas como precisión, *recall* y F1, los resultados de la Tabla 7 muestran que los tres modelos se desempeñan notablemente en la clasificación multietiqueta, con diferencias muy sutiles entre ellos.

Modelo	Precisión	Recall	F1-Score
Distil BERT uncased	0,73	0,80	0,76
BERT base uncased	0,72	0,76	0,78
Multilanguage BERT	0,72	0,76	0,77
RoBERTa	0,74	0,78	0,78

Tabla 7: Métricas de clasificación de los 4 mejores modelos

Con las predicciones obtenidos en esta etapa, se generan 21 matrices de confusión, una para cada etiqueta del conjunto de datos. Cada matriz de confusión ofrece una visión detallada de las predicciones del modelo en comparación con las etiquetas verdaderas, lo que permite identificar patrones de error específicos para cada clase. A partir de estas matrices, se calcula el porcentaje de acierto de cada modelo para cada etiqueta, como se puede observar en la Tabla 8.

Etiqueta	Multilanguage BERT	Distil BERT uncased	RoBERTa	BERT base uncased
Asuntos sociales (0)	43,61%	44,9%	50%	43,5%
Industria (1)	60,1%	58,6%	58,9%	59,7%
Asuntos financieros (2)	73,9%	70,08%	75,1%	73,9%
Intercambios económicos y comerciales (3)	85%	84,9%	91,5%	84,6%
Empresa y competencia (4)	59,01%	56,4%	43,2%	58,1%
Relaciones internacionales (5)	90,4%	90,3%	91,7%	90,1%

Agricultura silvicultura y pesca (6)	70,4%	68,7%	69%	71%
Producción, tecnología e investigación (7)	76,9%	79,1%	80,6%	74,6%
Transportes (8)	35,5%	34,8%	27%	32,5%
Trabajo y empleo (9)	95,5%	95,2%	95,5%	94,9%
Vida política (10)	84,5%	86,7%	87,7%	83,3%
Derecho (11)	58,6%	61,7%	76%	60,2%
Educación y comunicación (12)	70,7%	69,5%	70,7%	70,1%
Organizaciones internacionales (13)	93,5%	90,1%	93,2%	93,4%
Energía (14)	60,7%	61,7%	66,1%	60,5%
Unión Europea (15)	52,1%	58,3%	50,5%	50,4%
Ciencia (16)	33,3%	27,7%	27%	30%
Sector Agroalimentario (17)	65,5%	68%	73,2%	63%
Geografía (18)	59,5%	63,8%	69,1%	58,9%
Economía (19)	70,4%	73,3%	79,8%	70,2%
Medio ambiente (20)	82,2%	85,8%	88,7%	80,7%

Tabla 8: Porcentajes de clasificación de las 21 etiquetas del EuroVoc de los 4 mejores modelos

RoBERTa se destaca como el mejor modelo en la tabla debido a su rendimiento superior en comparación con *Multilanguage BERT*, *Distil BERT uncased* y *BERT base uncased*. Aunque no es el mejor en cada etiqueta individual, RoBERTa ofrece una combinación efectiva de alta precisión en varias categorías críticas y un desempeño global más sólido.

En general, RoBERTa supera a los otros modelos en varias categorías clave. Por ejemplo, RoBERTa logra una precisión del 91.5% en Intercambios económicos y comerciales, el mejor resultado entre todos los modelos. También mantiene una alta precisión en Trabajo y empleo con 95.5% y en Vida política con 87.7%, mostrando su capacidad para manejar eficazmente temas variados.

Además, muestra una mejora global en precisión comparado con *Distil BERT uncased* y *BERT base uncased* en categorías importantes. Por ejemplo, supera a *BERT base uncased* en Intercambios económicos y comerciales por 6.9% y a *Distil BERT uncased* en Medio ambiente por 2.9%.

A pesar de que *Distil BERT uncased* y *BERT base uncased* son versiones más ligeras y eficientes de BERT, RoBERTa ofrece una mayor precisión global, lo que demuestra que su entrenamiento más exhaustivo y su enfoque avanzado en optimización le otorgan una ventaja significativa en tareas de clasificación de textos.

Por lo tanto, la combinación de una precisión sobresaliente en categorías clave y un rendimiento global mejorado hace que RoBERTa sea el modelo más eficaz en esta evaluación. Esta ventaja es particularmente significativa en comparación con *Multilanguage BERT*, *DistilBERT* y *BERT clásico*, que no han sido optimizados específicamente con tanto material en español. Además, la arquitectura del modelo RoBERTa está basado en un modelo robustamente optimizado de BERT original, por lo que es notable su mejoría en los resultados.

A pesar de estos resultados, es evidente que aún hay margen de mejora, ya que solo la mitad de las etiquetas se clasifica de forma notable. Esto sugiere que, aunque RoBERTa [85] supera (cercanamente) al resto, los resultados globales pueden optimizarse con técnicas adicionales o con mayor cantidad de datos específicos para cada etiqueta. Debido a limitaciones de tiempo y recursos, no se han implementado mejoras en el modelo o en el entrenamiento en esta fase del estudio, pero estas oportunidades se plantean como trabajo a futuro.

5.4.8 Literatura relacionada

El análisis de los modelos y técnicas para la clasificación multiclase de textos parlamentarios revela que, aunque se han logrado resultados significativos, el campo aún ofrece numerosas oportunidades para avanzar hacia el estado del arte.

El estado del arte en la clasificación multietiqueta de documentos parlamentarios en Europa, con posibilidad de aplicación al ámbito parlamentario canario, alcanza aproximadamente un 67% de precisión en la clasificación general. Este rendimiento se obtiene en documentos en inglés mediante el uso de modelos RoBERTa [86].

A continuación, se presenta una revisión de tres trabajos recientes que destacan áreas de mejora y sugieren direcciones futuras para la investigación en el ámbito de la clasificación de documentos y la adaptación de modelos preentrenados. Estos estudios subrayan la necesidad de invertir más tiempo, investigación y recursos para superar los límites actuales y alcanzar resultados óptimos en tareas similares.

En [87], los autores abordan una limitación crucial de los modelos preentrenados, como BERT, en tareas de clasificación de documentos multiclase. El estudio revela que las palabras fuera del vocabulario (*Outside Of Vocabulary* o OOV) pueden llevar consigo conocimientos de dominio esenciales para las tareas de clasificación, pero a menudo son ignoradas por los modelos preentrenados, lo que limita su efectividad. La propuesta del VART, un método de preentrenamiento que extiende la capa de incrustación para aprender representaciones de palabras OOV, muestra que esta adaptación puede mejorar significativamente el rendimiento en tareas de clasificación multiclase. Este trabajo resalta que se requiere una adaptación más profunda y especializada de los modelos preentrenados para abordar las particularidades del dominio, lo cual implica que futuros estudios deben explorar cómo extender estas técnicas a escenarios más específicos y variados, como la gestión documental en parlamentos.

Por otro lado, en [88] introduce el modelo *BERT Doc-Label Attention Network* (BeNet), que emplea una red de atención para capturar relaciones entre documentos y etiquetas en tareas de clasificación multiclase. Este enfoque innovador demuestra que la integración de mecanismos de atención para conectar documentos con sus etiquetas puede mejorar la precisión y el rendimiento en la clasificación multiclase. Los resultados experimentales muestran que BeNet supera a los métodos existentes en *benchmarks* de clasificación multiclase. Este trabajo subraya que las técnicas avanzadas de atención y la creación de representaciones más ricas de documentos y etiquetas son áreas prometedoras para futuras investigaciones.

Finalmente, en [89], se explora un modelo híbrido que combina LSTM y GRU para mejorar la clasificación multiclase utilizando incrustaciones de palabras preentrenadas como BERT. Los resultados indican que, aunque el modelo híbrido ofrece mejoras en comparación con métodos anteriores, el modelo BERT base ajustado específicamente para la tarea supera al modelo híbrido en términos de pérdida de validación. Este estudio muestra que, a pesar de los avances actuales, se requiere una mayor

investigación en técnicas híbridas y el uso extensivo de transferencias de aprendizaje para lograr mejoras significativas en la clasificación multiclase. El desarrollo de modelos híbridos más sofisticados y el ajuste de modelos preentrenados en contextos específicos, como el análisis de textos parlamentarios, representan áreas clave para el progreso en este campo

Estos estudios destacan que, aunque se han hecho avances significativos en la clasificación multiclase de textos utilizando modelos preentrenados como BERT, existen varios aspectos que aún requieren desarrollo y mejora. La adaptación a palabras fuera del vocabulario, el uso de mecanismos de atención avanzados y el desarrollo de técnicas híbridas más eficaces son todos desafíos que deben ser abordados para avanzar hacia el estado del arte en la clasificación de textos. Estos puntos representan direcciones prometedoras para futuros trabajos de investigación en la gestión documental del parlamento, donde la exploración de nuevas técnicas y la optimización de modelos existentes serán esenciales para mejorar el desempeño en tareas de clasificación multiclase. Por ello, se propone abrir una nueva línea de investigación sobre estas modalidades, pues la clasificación documental, es una tarea necesaria para la gestión de estos en el Parlamento de Canarias.

5.4.9 Conclusión

A pesar de los esfuerzos realizados en esta investigación, los resultados obtenidos no permiten llegar a conclusiones definitivas sobre cuál es el modelo más adecuado para la clasificación multiclase de textos parlamentarios. Este estudio ha revelado varias limitaciones inherentes que han afectado la capacidad de alcanzar conclusiones definitivas y subraya la necesidad de una mayor profundización en el tema.

En primer lugar, una de las principales limitaciones fue la escasez de datos adecuados para entrenar y evaluar los modelos. La falta de datasets con una taxonomía EuroVoc equilibrada y representativa limita significativamente la capacidad para realizar un análisis exhaustivo. El único dataset disponible está desbalanceado, lo que afecta negativamente la calidad de las pruebas y la capacidad para evaluar de manera justa el rendimiento de los modelos en todas las categorías.

En segundo lugar, los recursos limitados destinados a la experimentación impidieron llevar a cabo una exploración más extensa de los modelos de BERT. La capacidad computacional y el tiempo de entrenamiento disponibles fueron insuficientes para realizar una investigación más profunda que podría haber revelado la verdadera capacidad de los modelos preentrenados en tareas de clasificación multiclase. Las restricciones en los recursos también limitaron la posibilidad de experimentar con configuraciones más avanzadas y exhaustivas.

Finalmente, como se observa en la literatura revisada en el apartado anterior, el tiempo y la profundidad necesarios para alcanzar el estado del arte en la clasificación de textos parlamentarios son significativos. Los estudios actuales muestran que alcanzar el máximo potencial de los modelos de BERT requiere no solo de técnicas avanzadas y datos de alta calidad, sino también de un esfuerzo de investigación prolongado y recursos extensos. La implementación de mecanismos de atención, la adaptación de vocabularios y el desarrollo de modelos híbridos son solo algunos de los aspectos que se deben explorar más a fondo.

6. APLICACIÓN DE GESTIÓN DOCUMENTAL

6.1. Introducción

Una vez examinados y desarrollados los tres frentes principales del proyecto, se ha decidido emprender una primera aproximación a estas herramientas mediante la implementación de una aplicación web. Esta aplicación tiene como propósito principal simular la utilidad y efectividad de las herramientas desarrolladas, proporcionando una interfaz práctica y accesible para demostrar su funcionamiento y evaluar su impacto en un entorno real.

El desarrollo de la aplicación web permite integrar de manera coherente los conocimientos teóricos y metodológicos adquiridos a lo largo del proyecto, creando una plataforma donde los usuarios pueden interactuar directamente con las herramientas y observar sus capacidades en acción. Este entorno de prueba no solo sirve para validar los conceptos y técnicas estudiadas, sino que también ofrece la oportunidad de identificar áreas de mejora y ajustes necesarios para optimizar el rendimiento de las herramientas en escenarios reales.

A través de esta aplicación, se busca lograr dos objetivos principales:

1. **Validación de Conceptos y Técnicas:** La aplicación web actúa como un medio para comprobar la efectividad de las herramientas desarrolladas en un entorno práctico. Al permitir a los usuarios interactuar con estas herramientas, se puede evaluar si los conceptos teóricos y metodológicos se traducen de manera efectiva en aplicaciones prácticas y si cumplen con los objetivos establecidos.
2. **Identificación de Áreas de Mejora:** La implementación de la aplicación web facilita la detección de posibles mejoras y ajustes. La experiencia práctica de los usuarios y el análisis del rendimiento de las herramientas en un contexto real proporcionan información valiosa para optimizar y refinar las soluciones desarrolladas.

6.2. Diseño

El diseño de la aplicación web desarrollada se basa en una arquitectura estructurada en dos componentes principales: el *frontend* y el *backend*. Estos componentes han sido implementados con el objetivo de asegurar modularidad, escalabilidad y facilidad de mantenimiento, utilizando tecnologías y prácticas modernas.

6.2.1. Backend

El *backend* de la aplicación web está desarrollado utilizando Flask [30], un microframework de Python conocido por su capacidad para construir aplicaciones web ligeras y escalables. Flask se eligió por su flexibilidad y simplicidad, características esenciales para el desarrollo ágil de aplicaciones web. La aplicación se ejecuta en un ordenador de sobremesa, una elección que, aunque presenta ciertos riesgos asociados con la apertura de puertos, se justifica por su propósito de desarrollo y demostración en el contexto de este trabajo de fin de grado. Esta configuración no ha sido externalizada a un servidor externo, dado que no es el enfoque principal de esta investigación.

Para garantizar un entorno de desarrollo consistente y una gestión eficiente de dependencias, el *backend* ha sido contenerizado utilizando Docker. Docker facilita la creación de entornos aislados, lo que permite asegurar que todas las dependencias y configuraciones necesarias para el funcionamiento del *backend* estén contenidas en una imagen reproducible.

El diseño del *backend* se organiza de la siguiente manera, siguiendo una estructura clara y modular:

- **auth**: Este directorio se encarga de la funcionalidad de autenticación, que permite el acceso a los modelos de Hugging Face y otras utilidades relacionadas. Aquí se gestionan las credenciales y se controla el acceso a los recursos del *backend*.
- **config**: Contiene el archivo de configuración para el registro de logs en la aplicación. Este archivo es esencial para la gestión y monitoreo de los eventos y errores que ocurren durante la ejecución del *backend*, facilitando el mantenimiento y la depuración.
- **chatbot_memory**: Este directorio actúa como un volumen de montaje para la imagen de ChromaDB en Docker. ChromaDB es utilizado para almacenar y acceder a la memoria del chatbot, asegurando que las interacciones y datos sean persistentes y accesibles durante las sesiones de usuario.
- **Databases**: Se encarga de la inicialización y configuración de las bases de datos utilizadas por la aplicación. Este directorio contiene los scripts y configuraciones necesarias para establecer y mantener las estructuras de datos que la aplicación necesita para funcionar correctamente.
- **src**: Alberga la lógica principal del *backend* de la aplicación. Aquí se encuentran los módulos que implementan las funcionalidades del *backend*, desde el procesamiento de las solicitudes del usuario hasta la interacción con las bases de datos y la ejecución de la lógica del negocio.
- **app.py**: Este archivo es el punto de entrada principal de la aplicación. En él se inicializan y configuran los componentes clave del *backend*, tales como las rutas de la aplicación y la integración con otros servicios.

El esquema general puede verse en la Figura 21:

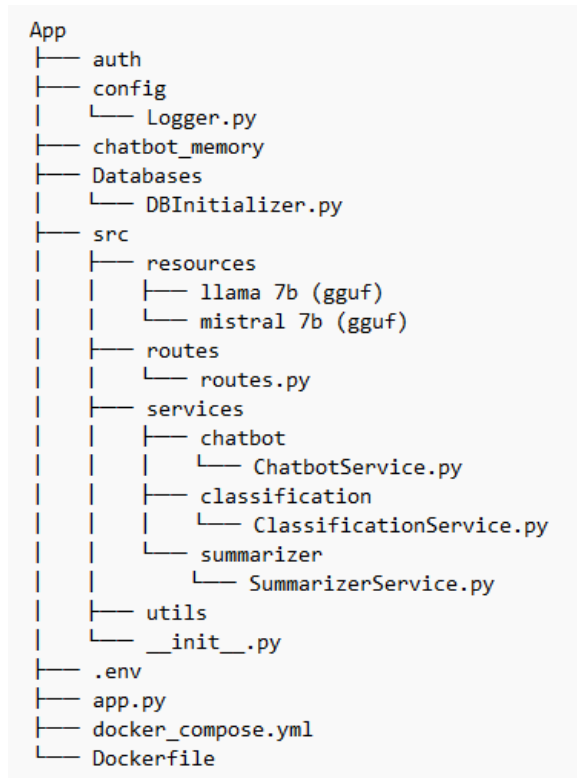


Figura 20: Arquitectura del backend de la aplicación

6.2.2. Frontend

El *frontend* de la aplicación web ha sido desarrollado utilizando la librería Streamlit [32] de Python, una herramienta ideal para la creación rápida de aplicaciones web interactivas. Para garantizar una separación clara entre el *frontend* y el *backend*, así como para simplificar su gestión, el *frontend* también se ha ejecutado dentro de un contenedor Docker. Esta práctica asegura que el entorno de desarrollo sea consistente y que las dependencias estén bien gestionadas, permitiendo una administración independiente y organizada del código.

A continuación, se detalla cada uno de los elementos de la interfaz de usuario:

- **assets**: Este directorio alberga fotos y otros recursos externos necesarios para la interfaz de usuario de la página web. En *assets*, se incluyen imágenes, íconos y otros archivos estáticos que enriquecen la experiencia visual de los usuarios y ayudan a mantener un diseño atractivo y coherente.
- **logs**: Este directorio está destinado a almacenar los registros de eventos y errores que ocurren en la aplicación web.
- **webapp**: La carpeta *webapp* contiene el código fuente y la lógica de la aplicación web, siendo el núcleo del desarrollo del *frontend*. Esta carpeta está organizada en subdirectorios que dividen las distintas funcionalidades y componentes de la aplicación:
 - **config**: Aquí se gestiona la configuración del registro de logs para el *frontend*. Esta configuración asegura que los eventos importantes y los errores sean capturados adecuadamente y almacenados para su análisis posterior.
 - **static_components**: Este directorio incluye componentes estáticos utilizados en la aplicación web, tales como archivos HTML y CSS que no cambian dinámicamente.
 - **views**: En *views*, se organizan las diferentes vistas de la aplicación web. Este directorio separa la lógica y las plantillas para cada funcionalidad específica, facilitando el diseño modular y la gestión de las distintas secciones de la aplicación.
 - **main.py**: Este archivo actúa como el punto de entrada principal para la aplicación web. En *main.py*, se crean y configuran las vistas de la aplicación, inicializando los diferentes componentes y rutas necesarias para que los usuarios puedan interactuar con la aplicación.

Al igual que en el *backend*, en la Figura 22 se podrá ver un esquema general de su arquitectura:

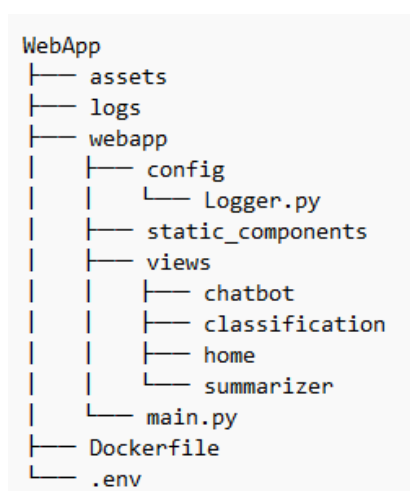


Figura 21: Arquitectura del frontend de la aplicación

6.2.3. Bases de datos

Las bases de datos utilizadas por la aplicación están contenidas dentro de un contenedor Docker, lo que asegura su persistencia y facilita tanto la administración como el respaldo de los datos. En esta implementación, se emplea exclusivamente una base de datos: `Session_diaries_embeddings` de ChromaDB, que se encarga de almacenar la memoria del chatbot.

Esta base de datos se ejecuta en un contenedor Docker, lo que proporciona un entorno aislado y replicable para su operación, simplificando la administración de dependencias y configuraciones.

6.2.4. Orquestación

Para el arranque y la gestión coordinada de los servicios del *frontend*, *backend* y bases de datos, se ha utilizado Docker Compose [57]. Esta herramienta permite definir y ejecutar aplicaciones multicontenedor, asegurando que cada componente del sistema se inicie en el orden correcto y con las configuraciones adecuadas. El flujo general de ejecución de la aplicación sigue el esquema mostrado en la Figura 23.

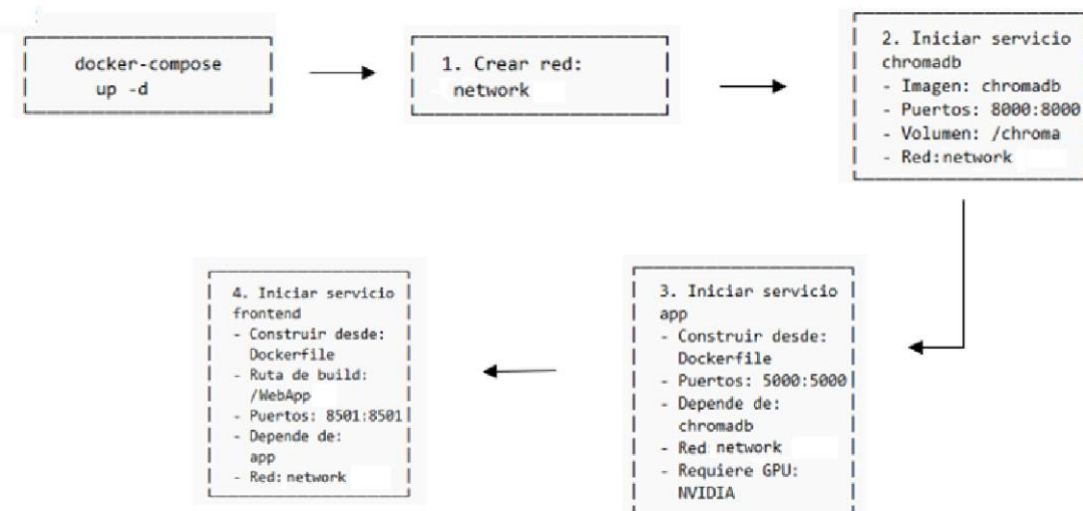


Figura 22: Flujo general de arranque de la aplicación

Cabe destacar que la implementación de esta arquitectura se ha llevado a cabo en un entorno de desarrollo local, utilizando un ordenador personal como servidor. En esta fase del proyecto, no se exploran en profundidad las configuraciones de seguridad necesarias para un despliegue en un entorno de producción, tales como la apertura de puertos y otras medidas de protección. Estos aspectos de seguridad exceden el alcance de este trabajo de fin de grado.

Adicionalmente, es importante mencionar que al iniciar el *backend* se debe esperar aproximadamente 20 minutos para que todos los servicios queden completamente operativos. Este tiempo es necesario para la carga en memoria de todos los modelos del chatbot, los resumidores y el clasificador. Esta fase de inicialización es crucial para asegurar que todos los componentes estén debidamente cargados y listos para su funcionamiento antes de que el sistema pueda ser utilizado de manera efectiva.

6.2.5. Casos de uso

Este documento presenta los casos de uso de una aplicación web con tres servicios principales: un chatbot para consultas sobre los diarios de sesión del Parlamento de Canarias, un resumidor de textos y un clasificador de textos basado en etiquetas del Eurovoc. Cada caso de uso describe las interacciones entre los usuarios y el sistema, incluyendo actores involucrados, precondiciones, flujo principal de eventos, flujos alternativos y postcondiciones. Estos casos de uso proporcionan una guía clara para el desarrollo y utilización de las funcionalidades de la aplicación.

Interacción con el Chatbot: Un usuario interactúa con el chatbot para hacer consultas sobre los diarios de sesión del Parlamento de Canarias, configurando la temperatura del chatbot y el número de tokens de salida.

- Flujo básico:

1. El usuario elige en la opción de “chatbot” en la interfaz principal.
2. El usuario configura la temperatura del chatbot y el número de tokens de salida.
3. El usuario introduce una consulta sobre los diarios de sesión.
4. El sistema envía la solicitud al *backend* y obtiene la memoria de los diarios almacenados más relacionados con el texto de entrada.
5. El modelo procesa la consulta junto a la memoria obtenida como contexto.
6. El usuario ve la respuesta en la interfaz de chatbot.

- Extensiones:

- Si el usuario no escribe el mensaje y da a “enviar”, saldrá un aviso de que debe rellenar el mensaje para poder realizar la consulta.
- Si el chatbot no puede procesar la solicitud, se muestra un mensaje de error al usuario.

Generación de Resúmenes: Un usuario proporciona un texto o un archivo PDF para que el sistema genere un resumen. El usuario puede configurar el porcentaje de resumen y elegir el modelo (Mistral, GPT, Llama).

- Flujo básico:

1. El usuario elige en la opción de “resumen” en la interfaz principal.
2. El usuario introduce un texto o carga un archivo PDF.
3. El usuario configura el porcentaje de resumen y elige el modelo (Mistral, GPT, Llama).
4. El sistema envía el texto y la configuración al *backend*.
5. El usuario ve el resumen en la interfaz de resumen.

- Extensiones:

- Si usuario manda al mismo tiempo un texto de entrada y un fichero PDF de entrada, devolverá un mensaje de aviso, indicándole que sólo puede mandar un texto, no dos simultáneamente.

Clasificación de Textos: Un usuario proporciona un texto o un archivo PDF para que el sistema lo clasifique según las etiquetas del EuroVoc.

- Flujo básico:

1. El usuario elige la opción “clasificación” en la interfaz principal.
2. El usuario introduce un texto o carga un archivo PDF.
3. El sistema envía los datos al *backend*.
4. El usuario ve los resultados en la interfaz de clasificación.

- Extensiones:

1. Si usuario manda al mismo tiempo un texto de entrada y un fichero PDF de entrada, devolverá un mensaje de aviso, indicándole que sólo puede mandar un texto, no dos simultáneamente.

6.2.6 Puntos principales de la implementación

En esta sección se describen las cuatro vistas principales de la aplicación web, cada una diseñada para ofrecer una funcionalidad específica del sistema. Estas vistas permiten a los usuarios interactuar de manera intuitiva y eficiente con las herramientas desarrolladas, demostrando sus capacidades y utilidad en diversos contextos.

La primera vista es el menú de inicio, que ofrece una descripción general del proyecto y las tres funcionalidades principales: clasificación, chatbot y generación de resúmenes. Esta página de inicio sirve como punto de partida para los usuarios, proporcionando una breve introducción y accesos directos a cada una de las vistas principales de la aplicación. Esta vista puede verse en la Figura 24, mostrada a continuación.



Figura 23: Vista del menú de inicio de la aplicación de gestión documental

La segunda vista (mostrada en la Figura 25) está dedicada a la funcionalidad de clasificación. En esta vista, los usuarios pueden introducir texto en un cuadro de entrada o cargar un archivo PDF. Una vez procesado, el clasificador devuelve una tabla con las etiquetas asignadas al documento. Esta vista permite a los usuarios ver de manera clara y organizada las categorías o etiquetas que el modelo ha identificado en el texto.



Figura 24: Vista del clasificador de la aplicación de gestión documental

Continuando con la tercera vista, la Figura 26 muestra como los usuarios los usuarios pueden configurar el modelo que realizará el resumen, elegir la versión del modelo (como Mistral 7B, 8xB, GPT-3.5 o 4.0), y seleccionar el porcentaje de resumen a generar. El texto de entrada se puede proporcionar en un cuadro de entrada o mediante un archivo PDF. Una vez procesado, el resumen generado se muestra en un nuevo cuadro en la aplicación, permitiendo al usuario ver rápidamente una versión condensada del contenido original.



Figura 25: Vista del resumidor de la aplicación de gestión documental

Finalmente, la cuarta y última vista es la del chatbot. En esta vista, los usuarios pueden configurar parámetros como la temperatura del modelo y el máximo de tokens de salida. El texto se introduce en un cuadro de entrada, y el chatbot devuelve la respuesta en formato de chat. Esta interfaz interactiva reflejada en la Figura 27, permite a los usuarios tener conversaciones con el modelo de lenguaje, recibiendo respuestas generadas en tiempo real.



Figura 26: Vista del chatbot de la aplicación de gestión documental

Estas cuatro vistas principales están diseñadas para proporcionar una experiencia de usuario intuitiva y eficiente, permitiendo la interacción directa con las funcionalidades clave de la aplicación y demostrando el potencial de las herramientas desarrolladas en el proyecto. Este desarrollo final, culmina la hipótesis principal del proyecto, pues con los niveles reducidos de recursos con los que se ha trabajado en el proyecto se ha conseguido diseñar una primera aproximación de herramientas de gestión documental funcional, a través de técnicas del procesamiento del lenguaje natural.

7. APORTACIONES

En este trabajo se han realizado diversas contribuciones significativas en el ámbito de la experimentación y evaluación de modelos de lenguaje natural, destacando tres áreas principales.

En este sentido, se ha llevado a cabo la adaptación de un modelo de lenguaje natural para una tarea específica, asegurando su despliegue local sin dependencias de APIs externas. Esta implementación local permite un mayor control sobre los datos y la privacidad, eliminando la necesidad de confiar en servicios de terceros y reduciendo los costos asociados con el uso de APIs externas. Esta contribución es particularmente relevante para organizaciones que manejan información sensible o buscan reducir su dependencia de proveedores externos.

A su vez, se ha desarrollado un estudio exhaustivo sobre las capacidades intrínsecas de los modelos de lenguaje en tareas de generación de resúmenes. Este análisis ha permitido evaluar el rendimiento y eficiencia de los modelos en términos de precisión y coherencia de los resúmenes generados. La importancia de este estudio radica en su capacidad para ofrecer una evaluación detallada y accesible de los modelos en esta tarea, sin necesidad de procesos de ajuste fino.

La capacidad de los modelos BERT para manejar tareas de clasificación compleja con alta precisión abre nuevas oportunidades para mejorar los sistemas de gestión documental, facilitando una organización más efectiva y una recuperación de información más precisa. Además, este estudio ha sentado las bases para futuras investigaciones que podrían explorar técnicas adicionales, como el uso combinado de *embeddings* o arquitecturas de redes avanzadas, para optimizar aún más el rendimiento de la clasificación multietiqueta.

De este modo, se ha comprobado que el desarrollo de aplicaciones de gestión documental dentro de un ámbito de conocimiento específico es posible, y mejorable, a través del uso de técnicas de procesamiento del lenguaje natural.

8. CONCLUSIÓN

La proyección global de este estudio ha revelado la relevancia del uso de modelos del lenguaje locales frente a los grandes modelos emergentes.

Por una parte, se ha demostrado que es posible mantener un chatbot para la asistencia en búsquedas documentales en el sector parlamentario, con la posibilidad de extender su uso a cualquier dominio de conocimiento. Este avance representa un significativo apoyo para los profesionales del área, mejorando la precisión en la recuperación de información relevante.

Asimismo, el uso de modelos del lenguaje sin *fine-tuning* previo para la generación de resúmenes ha evidenciado que esta tarea no requiere de un ajuste fino para alcanzar un desempeño notable. Destaca particularmente el modelo Mistral-8x7B, que ha mostrado una competencia considerable con modelos como GPT-3.5 Turbo.

Adicionalmente, la clasificación multietiqueta de documentos ha revelado que esta puede realizarse de forma automática mediante modelos BERT. Los resultados indican que este modelo es un buen punto de partida para esta tarea, y que es posible mejorar su desempeño mediante el uso combinado de técnicas de *embeddings* (GloVe [90]) o el uso simultáneo de otras arquitecturas de red (recurrentes y LSTM). Sin embargo, debido a limitaciones de tiempo y alcance del estudio, estas mejoras se sugieren para trabajos futuros.

Como conclusión del estudio, se ha demostrado que los modelos del lenguaje locales son alternativas fiables y efectivas frente a los grandes modelos del lenguaje, ofreciendo un rendimiento comparable en ciertas tareas como la generación de resúmenes y solucionando problemas adicionales relacionados con la privacidad y el cumplimiento legal, lo cual amplía sus potenciales usos en diversos campos profesionales.

9. TRABAJOS FUTUROS

El presente proyecto ha permitido explorar diversas facetas de la clasificación multiclase de textos parlamentarios y la adaptación de modelos de lenguaje para la asistencia documental, pero también ha puesto de manifiesto varias áreas de mejora y expansión. A partir de las limitaciones encontradas, surgen diversas oportunidades para avanzar en estos campos de estudio.

En el ámbito de la asistencia en gestión documental, es crucial mejorar la calidad y extensión de los datos etiquetados para el aprendizaje supervisado. Futuros estudios podrían enfocarse en desarrollar estrategias para generar conjuntos de datos etiquetados más extensos y de alta calidad, además de explorar metodologías para la creación de datos sintéticos que amplíen las capacidades del modelo. Este enfoque permitirá mejorar el rendimiento del chatbot en escenarios reales, facilitando su aplicación en contextos de gestión documental.

Por otro lado, uno de los aspectos más prometedores para el futuro de esta línea de investigación es el uso de entrenamiento por refuerzo. Este enfoque puede ofrecer mejoras significativas en la adaptabilidad y precisión del chatbot. El aprendizaje por refuerzo permite que el modelo optimice sus decisiones a través de un sistema de recompensas y penalizaciones, donde las "recompensas" (*rewards*) refuerzan comportamientos deseados y las "penalizaciones" (*penalty*) desincentivan errores. Experimentar con diferentes funciones de recompensa y penalización podría permitir que el asistente personal ajuste dinámicamente sus respuestas basadas en la retroalimentación continua del usuario, mejorando así su desempeño en tareas específicas y aumentando la satisfacción del usuario.

Al igual que en el caso anteriormente mencionado, uno de los aspectos cruciales que debe ser abordado es el desarrollo de datasets más equilibrados para la clasificación multietiqueta. La investigación revela que la disponibilidad de datos con una taxonomía EuroVoc adecuada y equilibrada es una limitación significativa. Para avanzar, es esencial crear conjuntos de datos que no solo sean representativos de una gama más amplia de categorías, sino que también estén bien etiquetados.

Además, la exploración de técnicas y adaptaciones específicas representa una oportunidad para llevar el estudio a un nivel más profundo. Las investigaciones revisadas en este proyecto, como las propuestas en VART [87] para manejar palabras fuera del vocabulario y los mecanismos de atención en BeNet [88], ofrecen valiosas perspectivas para futuras investigaciones. Adaptar estas técnicas para la clasificación de textos parlamentarios podría resultar en mejoras significativas en la precisión. La aplicación de estos enfoques avanzados podría contribuir a resolver problemas persistentes en el análisis de documentos y potenciar el desarrollo de nuevas metodologías.

Las líneas de investigación futura definidas en este estudio establecen un marco sólido para avanzar en la clasificación multiclase de textos parlamentarios y en la adaptación de modelos de lenguaje. Cada una de estas áreas no solo aborda las limitaciones actuales, sino que también abre nuevas oportunidades para optimizar la gestión documental a través del procesamiento del lenguaje natural. Al explorar estas oportunidades, se pueden desarrollar enfoques innovadores que optimicen la efectividad de los sistemas de clasificación textual en el contexto parlamentario.

10. BIBLIOGRAFÍA

- [1] University, Standford. (2024). N-gram Language Models. 30. Recuperado el 5 de Julio de 2024, de <https://web.stanford.edu/~jurafsky/slp3/3.pdf>
- [2] Qader, W. A. (2019). An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges. Recuperado el 5 de Julio de 2024, de https://www.researchgate.net/publication/338511771_An_Overview_of_Bag_of_Wordslmportance_Implementation_Applications_and_Challenges
- [3] Tomas Mikolov, K. C. (2013). Efficient Estimation of Word Representations in Vector Space. Recuperado el 5 de Julio de 2024, de <https://arxiv.org/abs/1301.3781>
- [4] Jordan, M. I. (1985). Serial order: a parallel distributed processing approach. Recuperado el 5 de Julio de 2024
- [5] Schmidhuber, J. (2014). Deep Learning in Neural Networks: An Overview. Recuperado el 5 de Julio de 2024, de <https://people.idsia.ch/~juergen/DeepLearning2July2014.pdf>
- [6] Ashish Vaswani, N. S. (2017). Attention is All You Need. Advances in Neural Information Processing Systems (NIPS). Recuperado el 27 de Mayo de 2024
- [7] Mrs. Drakshayini, M. N. (2023). Evolution of Chatbots Using Artificial Intelligenceand. Recuperado el 5 de Julio de 2024, de <https://ejmcm.com/uploads/paper/b092a2a48528ce4ecbd3b39fc39fd8c5.pdf>
- [8] Alec Radford, K. N. (2018). Improving Language Understanding by Generative Pre-Training. Recuperado el 5 de Julio de 2024, de <https://paperswithcode.com/paper/improving-language-understanding-by>
- [9] OpenAI. (s.f.). OpenAI. Recuperado el 31 de Mayo de 2024, de <https://openai.com/>
- [10] OpenAI. (Septiembre de 2021). OpenAI. Recuperado el 29 de Junio de 2024, de GPT-3.5 Turbo: <https://platform.openai.com/docs/models/gpt-3-5-turbo>
- [11] OpenAI. (13 de Mayo de 2024). OpenAI. Recuperado el 29 de Junio de 2024, de Hello GPT-4o: <https://openai.com/index/hello-gpt-4o/>
- [12] Meta. (18 de Abril de 2024). AI Meta. Recuperado el 27 de Mayo de 2024, de Introducing Meta Llama 3: The most capable openly available LLM to date: <https://ai.meta.com/blog/meta-llama-3/>
- [13] Ley Orgánica 3/2018, d. 5. (2018). Artículo 40. Boletín Oficial del Estado (BOE)., Gobierno de España. Recuperado el 30 de Mayo de 2024, de https://noticias.juridicas.com/base_datos/Laboral/632849-lo-3-2018-de-5-dic-proteccion-de-datos-personales-y-garantia-de-los-derechos.html#t6
- [14] Ley Orgánica 3/2018, d. 5. (2018). Artículo 41. Boletín Oficial del Estado (BOE)., Gobierno de España. Recuperado el 30 de Mayo de 2024, de https://noticias.juridicas.com/base_datos/Laboral/632849-lo-3-2018-de-5-dic-proteccion-de-datos-personales-y-garantia-de-los-derechos.html#t6
- [15] Ley Orgánica 3/2018, d. 5. (2018). Artículo 42. Boletín Oficial del Estado (BOE)., Gobierno de España. Recuperado el 30 de Mayo de 2024, de https://noticias.juridicas.com/base_datos/Laboral/632849-lo-3-2018-de-5-dic-proteccion-de-datos-personales-y-garantia-de-los-derechos.html#t6

- [16] Ley Orgánica 3/2018, d. 5. (2018). Artículo 43. Boletín Oficial del Estado (BOE), Gobierno de España. Recuperado el 30 de Mayo de 2024, de https://noticias.juridicas.com/base_datos/Laboral/632849-lo-3-2018-de-5-dic-proteccion-de-datos-personales-y-garantia-de-los-derechos.html#a44
- [17] Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, d. 2. (2016). Artículo 48. Diario Oficial de la Unión Europea (DOUE), Parlamento y Consejo de la Unión Europea. Recuperado el 30 de Mayo de 2024, de https://noticias.juridicas.com/base_datos/Privado/574082-regl-2016-679-ue-de-27-abr-proteccion-de-las-personas-fisicas-en-lo-que.html#a48
- [18] Gillham, J. (29 de Enero de 2024). Originality.AI. Recuperado el 4 de Junio de 2024, de Hugging Face Statistics: The State of Open-Source Machine Learning: <https://originality.ai/blog/huggingface-statistics>
- [19] Parlamento de Canarias. (s.f.). Parcan. Recuperado el 4 de Junio de 2024, de <https://www.parcan.es/>
- [20] EuroVoc. (s.f.). EuroVoc. Recuperado el 4 de Junio de 2024, de EUR-Lex: <https://eur-lex.europa.eu/browse/eurovoc.html?locale=es>
- [21] Jet Brains. (s.f.). Jet Brains. Recuperado el 12 de Junio de 2024, de IntelliJ IDEA: <https://www.jetbrains.com/es-es/idea/>
- [22] Project Apache Maven. (s.f.). Maven. Recuperado el 12 de Junio de 2024, de Maven's Objectives: <https://maven.apache.org/what-is-maven.html>
- [23] python. (4 de Octubre de 2024). Python. Recuperado el 13 de Junio de 2024, de Python 3.10.0: <https://www.python.org/downloads/release/python-3100/>
- [24] Jet Brains. (s.f.). PyCharm. Recuperado el 13 de Junio de 2024, de El IDE de Python para la ciencia de datos y el desarrollo web: <https://jetbrains.com/es-es/pycharm/>
- [25] OpenAI. (s.f.). openai. Recuperado el 9 de Junio de 2024, de openai-python: <https://github.com/openai/openai-python>
- [26] MongoDB. (s.f.). PyMongo. Recuperado el 9 de Junio de 2024, de PyMongo 4.7.3 Documentation: <https://pymongo.readthedocs.io/en/stable/>
- [27] MongoDB. (s.f.). MongoDB. Recuperado el 9 de Junio de 2024, de <https://www.mongodb.com/>
- [28] Hugging Face. (s.f.). Hugging Face. Recuperado el 9 de Junio de 2024, de Transformers.
- [29] community, G. G. (s.f.). llama-cpp-python. Recuperado el 9 de Junio de 2024, de <https://github.com/abetlen/llama-cpp-python>
- [30] Flask. (s.f.). Flask. Recuperado el 9 de Junio de 2024, de Python Version: <https://flask.palletsprojects.com/en/3.0.x/installation/#python-version>
- [31] PyPI. (s.f.). requests. Recuperado el 9 de Junio de 2024, de <https://pypi.org/project/requests/>
- [32] Streamlit. (s.f.). Streamlit. Recuperado el 9 de Junio de 2024, de A faster way to build and share data apps: <https://streamlit.io/>

- [33] PyTorch. (s.f.). PyTorch. Recuperado el 9 de Junio de 2024, de GET STARTED: <https://pytorch.org/>
- [34] Hugging Face. (s.f.). Hugging Face. Recuperado el 9 de Junio de 2024, de TRL - Transformer Reinforcement Learning: <https://huggingface.co/docs/trl/index>
- [35] Hugging Face. (s.f.). Hugging Face. Recuperado el 9 de Junio de 2024, de PEFT: <https://huggingface.co/docs/peft/index>
- [36] PyPI. (s.f.). PyPI. Recuperado el 9 de Junio de 2024, de Flask-SocketIO: <https://pypi.org/project/Flask-SocketIO/>
- [37] ChromaDB. (s.f.). ChromaDB. Recuperado el 9 de Junio de 2024, de the AI-native open-source embedding database: <https://www.trychroma.com/>
- [38] Docker. (s.f.). Docker. Recuperado el 9 de Junio de 2024, de <https://www.docker.com/>
- [39] Jupyter. (s.f.). Jupyter. Recuperado el 9 de Junio de 2024, de Free software, open standards, and web services for interactive computing across all programming languages: <https://jupyter.org/>
- [40] Weights & Biases. (s.f.). Weights & Biases. Recuperado el 9 de Junio de 2024, de The AI developer platform: <https://wandb.ai/site>
- [41] Microsoft. (s.f.). Microsoft. Recuperado el 13 de Junio de 2024, de PowerBI: <https://www.microsoft.com/es-es/power-platform/products/power-bi>
- [42] Parcan. (2024). Parcan. Recuperado el 4 de Junio de 2024, de Diario de Sesiones: https://www.parcn.es/pub/indice_ds.py
- [43] Jean-Baptiste Cordonnier, A. L. (2020). Multi-Head Attention: Collaborate Instead of Concatenate. Recuperado el 5 de Julio de 2024, de <https://arxiv.org/abs/2006.16362>
- [44] Rico Sennrich, B. H. (2015). Neural Machine Translation of Rare Words with Subword Units. Recuperado el 5 de Julio de 2024, de <https://arxiv.org/abs/1508.07909>
- [45] Zhi Jing, Y. S. (2024). When Large Language Models Meet Vector Databases: A Survey. Recuperado el 5 de Julio de 2024, de <https://arxiv.org/html/2402.01763v2>
- [46] Langchain. (s.f.). Langchain. Recuperado el 11 de Junio de 2024, de <https://www.langchain.com/>
- [47] Jacob Devlin, M.-W. C. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Recuperado el 5 de Julio de 2024, de <https://arxiv.org/abs/1810.04805>
- [48] MistralAI. (s.f.). MistralAI. Recuperado el 11 de Junio de 2024, de <https://mistral.ai/>
- [49] Albert Q. Jiang, A. S.-A. (2023). Mistral 7B. Recuperado el 5 de Julio de 2024, de <https://arxiv.org/abs/2310.06825>
- [50] MistralAI. (29 de Mayo de 2024). MistralAI. Recuperado el 11 de Junio de 2024, de Codestral: Hello, world!: <https://mistral.ai/news/codestral/>
- [51] Meta. (s.f.). Meta. Recuperado el 11 de Junio de 2024, de Llama models and tools: <https://llama.meta.com/>

- [52] Dey, S. (11 de Abril de 2024). Recuperado el 11 de Junio de 2024, de Decoding Llama 3: A Quick Overview of the Model - From Llama 2 to Llama 3: <https://medium.com/@soumava.dey.ai/decoding-llama-3-a-quick-overview-of-the-model-7e69abcdb6a>
- [53] Gemini Team Google: Rohan Anil, S. B.-B. (2023). Gemini: A Family of Highly Capable Multimodal Models. Recuperado el 5 de Julio de 2024, de <https://arxiv.org/abs/2312.11805>
- [54] Sundar Pichai, D. H. (15 de Febrero de 2024). Google The keyword. Recuperado el 12 de Junio de 2024, de Our next-generation model: Gemini 1.5: <https://blog.google/technology/ai/google-gemini-next-generation-model-february-2024/#sundar-note>
- [55] Haya, P. (s.f.). Instituto de Ingeniería del Conocimiento (IIC). Recuperado el 15 de Junio de 2024, de Metodología CRISP-DM en Ciencia de Datos: <https://www.iic.uam.es/innovacion/metodologia-crisp-dm-ciencia-de-datos/>
- [56] Refactoring Guru. (s.f.). Refactoring Guru. Recuperado el 16 de Junio de 2024, de Patrón de diseño Builder: <https://refactoring.guru/es/design-patterns/builder>
- [57] Docker. (s.f.). Docker. Recuperado el 16 de Junio de 2024, de Docker Compose overview: <https://docs.docker.com/compose/>
- [58] MistralAI. (27 de Septiembre de 2023). MistralAI. Recuperado el 19 de Junio de 2024, de Mistral 7B - The best 7B modelo to date, Apache 2.0: <https://mistral.ai/news/announcing-mistral-7b/>
- [59] MistralAI. (s.f.). HuggingFace. Recuperado el 19 de Junio de 2024, de mistralai/Mistral-7B-Instruct-v0.2: <https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2>
- [60] Faro de Vigo. (12 de Febrero de 2007). Faro de Vigo. Recuperado el 19 de Junio de 2004, de Fallecen 6 excursionistas que quedaron atrapados en una gruta en Tenerife: <https://www.farodevigo.es/sucesos/2007/02/12/fallecen-6-excursionistas-que-daron-atrapados-18161500.html#:~:text=Las%20v%C3%ADctimas%20son%20Juan%20Luis,los%20excursionistas%20continuaban%20anoche%20hospitalizados.>
- [61] Long Ouyang, J. W. (2022). Training language models to follow instructions with human feedback. Recuperado el 5 de Julio de 2024, de <https://arxiv.org/abs/2203.02155>
- [62] OpenAI. (s.f.). OpenAI Platform. Recuperado el 19 de Junio de 2024, de GPT-4 Turbo and GPT-4: <https://platform.openai.com/docs/models/gpt-4-turbo-and-gpt-4>
- [63] OpenAI. (s.f.). OpenAI Platform. Recuperado el 19 de Junio de 2024, de Prompt engineering: <https://platform.openai.com/docs/guides/prompt-engineering>
- [64] Edward J. Hu, Y. S.-Z. (2021). LoRA: Low-Rank Adaptation of Large Language Models. Recuperado el 20 de Junio de 2024, de <https://arxiv.org/abs/2106.09685>
- [65] Yun Luo, Z. Y. (2023). An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-tuning. Recuperado el 20 de Junio de 2024, de <https://arxiv.org/abs/2308.08747>

- [66] Valiant, T. R. (30 de Abril de 2024). The Modern Algorithmic Toolbox: The Singular Value Decomposition (SVD) - 4 The Singular Value Decomposition (SVD). Recuperado el 20 de Junio de 2024, de Stanford Edu: <https://web.stanford.edu/class/cs168/l/l9.pdf>
- [67] Valiant, T. R. (30 de Abril de 2024). The Modern Algorithmic Toolbox: The Singular Value Decomposition (SVD) - 5 Low-Rank Approximations from the SVD. Recuperado el 20 de Junio de 2024, de Stanford Edu: <https://web.stanford.edu/class/cs168/l/l9.pdf>
- [68] Hugging Face. (s.f.). Hugging Face. Recuperado el 20 de Junio de 2024, de bitsandbytes: <https://huggingface.co/docs/bitsandbytes/main/en/index>
- [69] Ilya Loshchilov, F. H. (2019). Decoupled Weight Decay Regularization. Recuperado el 6 de Julio de 2024, de <https://arxiv.org/abs/1711.05101>
- [70] Hugging Face. (s.f.). Hugging Face. Recuperado el 22 de Junio de 2024, de Supervised Fine-tuning Trainer: https://huggingface.co/docs/trl/sft_trainer
- [71] OpenAI. (s.f.). OpenAI. Recuperado el 24 de Junio de 2024, de GET STARTED - Embeddings - Embedding models: <https://platform.openai.com/docs/guides/embeddings/use-cases>
- [72] OpenAI. (s.f.). OpenAI. Recuperado el 25 de Junio de 2024, de GET STARTED - Models - GPT-4o: <https://platform.openai.com/docs/models/gpt-4o>
- [73] OpenAI. (s.f.). PyPL. Recuperado el 6 de Julio de 2024, de tiktoken 0.1.1: tiktoken 0.1.1
- [74] TheBloke. (s.f.). Hugging Face. Recuperado el 17 de Junio de 2024, de TheBloke/Llama-2-7B-Chat-GGUF: <https://huggingface.co/TheBloke/Llama-2-7B-Chat-GGUF>
- [75] TheBloke. (s.f.). Hugging Face. Recuperado el 17 de Junio de 2024, de TheBloke/Llama-2-13B-chat-GGUF: <https://huggingface.co/TheBloke/Llama-2-13B-chat-GGUF>
- [76] TheBloke. (s.f.). Hugging Face. Recuperado el 17 de Junio de 2024, de TheBloke/Mistral-7B-Instruct-v0.2-GGUF: <https://huggingface.co/TheBloke/Mistral-7B-Instruct-v0.2-GGUF>
- [77] TheBloke. (s.f.). *Hugging Facae*. Recuperado el 17 de Junio de 2024, de TheBloke/Mixtral-8x7B-Instruct-v0.1-GGUF: <https://huggingface.co/TheBloke/Mixtral-8x7B-Instruct-v0.1-GGUF>
- [78] Refactoring Guru. (s.f.). Refactoring Guru. Recuperado el 17 de Junio de 2024, de Factory Method: <https://refactoring.guru/design-patterns/factory-method>
- [79] cosatalcph. (s.f.). Hugging Face - Datasets. Recuperado el 16 de Junio de 2024, de coastalcph/multi_eurlex: https://huggingface.co/datasets/coastalcph/multi_eurlex
- [80] PyTorch. (s.f.). PyTorch. Recuperado el 21 de Junio de 2024, de torch.utils.data - WeightedRandomSampler: <https://pytorch.org/docs/stable/data.html>
- [81] Google. (s.f.). Hugging Face. Recuperado el 21 de Junio de 2024, de google-bert/bert-base-multilingual-cased: <https://huggingface.co/google-bert/bert-base-multilingual-cased>
- [82] Plan de Tecnologías del Lenguaje - Gobierno de España. (s.f.). Hugging Face. Recuperado el 1 de Julio de 2024, de PlanTL-GOB-ES/roberta-base-bne: <https://huggingface.co/PlanTL-GOB-ES/roberta-base-bne>
- [83] dccuchile. (s.f.). Hugging Face. Recuperado el 1 de Julio de 2024, de dccuchile/distilbert-base-spanish-uncased: <https://huggingface.co/dccuchile/distilbert-base-spanish-uncased>

- [84] dccuchile. (s.f.). Hugging Face. Recuperado el 1 de Junio de 2024, de dccuchile/bert-base-spanish-wwm-cased: <https://huggingface.co/dccuchile/bert-base-spanish-wwm-cased>
- [85] Yinhan Liu, M. O. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. Recuperado el 2 de Julio de 2024, de <https://arxiv.org/abs/1907.11692>
- [86] Ilias Chalkidis, M. F. (2021). MultiEURLEX – A multi-lingual and multi-label legal document. Recuperado el 6 de Julio de 2024, de <https://arxiv.org/abs/2109.00904>
- [87] Zhongguang Zheng, L. F. (2022). VART: Vocabulary Adapted BERT Model for Multi-label Document. 9. Recuperado el 1 de Julio de 2024, de <https://aclanthology.org/2022.paclic-1.18.pdf>
- [88] Bo Li. (2024). BeNet: BERT Doc-Label Attention Network for Multi-Label Text Classification. Proceedings of the 4th International Conference on Signal Processing and Machine Learning. Recuperado el 1 de Julio de 2024, de <https://ace.ewapublishing.org/article/d5807e35d23545a0bce1ce4ba5476a55>
- [89] Shreyashree S, P. S. (2022). BERT-Based Hybrid RNN Model for Multi-class Text Classification to Study the Effect of Pre-trained Word Embeddings. 8. Recuperado el 2 de Julio de 2024, de https://thesai.org/Downloads/Volume13No9/Paper_79-BERT_Based_Hybrid_RNN_Model.pdf
- [90] Jeffrey Pennington, R. S. (2014). Glove: Global Vectors for Word Representation. Recuperado el 7 de Julio de 2024, de https://www.researchgate.net/publication/284576917_Glove_Global_Vectors_for_Word_Representation