Trabajo de Fin de Grado

Estudio de Técnicas de Afinamiento para Modelos del Lenguaje de Dominio Específico

TITULACIÓN: Grado en Ciencia e Ingeniería de Datos

AUTOR: Álvaro Juan Travieso García

TUTORIZADO POR:
Cayetano Nicolás Guerra Artal
José Miguel Santos Espino

Fecha [12/2024]

Agradecimientos

Quiero dedicar este espacio a expresar mi más sincero agradecimiento a todas las personas que, de una manera u otra, han contribuido a la realización de este proyecto.

En primer lugar, agradezco profundamente a mi familia: a mis padres, cuya paciencia, apoyo incondicional y palabras de aliento han sido fundamentales para superar los retos que este trabajo ha presentado; y a mi hermana, cuya confianza en mí y motivación constante han sido una fuente de inspiración a lo largo de este camino.

Quiero también expresar mi gratitud al profesor Cayetano, quien no solo desempeñó un papel crucial como mi tutor en este Trabajo de Fin de Grado, sino también como un profesor cuyo conocimiento y guía académica han marcado una etapa importante de mi formación. Su disponibilidad, consejos y experiencia han sido clave para la concreción de este proyecto.

A todos ellos, gracias por creer en mí y por ser parte esencial de este logro. Este proyecto no habría sido posible sin su apoyo y confianza.

Resumen

Este proyecto se centra en la implementación de un modelo de lenguaje preentrenado (LLM) ajustado mediante Fine-Tuning. Para ello, se desarrolló una herramienta de scraping que extrae datos relevantes de diversas fuentes web, creando un corpus específico para el proyecto. Se realizó el Fine-Tuning de dos LLMs, Qwen2 1.5B y Mistral 7b, adaptándolos para tareas concretas. Además, se diseñó una herramienta de test para evaluar el rendimiento de los modelos ajustados, comparando sus respuestas con las correctas. Los resultados muestran mejoras en el desempeño tras el ajuste, evidenciando el impacto del Fine-Tuning en la personalización de los LLMs para tareas específicas.

Abstract

This project focuses on fine-tuning a pre-trained language model (LLM). A scraping tool was developed to extract relevant data from various web sources, creating a task-specific corpus. Fine-Tuning was performed on two LLMs, Qwen2 1.5B and Mistral 7b, adapting them for specific tasks. Additionally, a testing tool was created to evaluate the performance of the fine-tuned models by comparing their responses to correct answers. The results show performance improvements after fine-tuning, highlighting the impact of this technique in customizing LLMs for specific tasks.

ÍNDICE

1. INTRODUCCIÓN

1.1 Contextualización

En los últimos años, el campo de la inteligencia artificial ha experimentado un crecimiento exponencial, convirtiéndose en un factor clave en prácticamente todos los aspectos de nuestra sociedad moderna. Su impacto es evidente en áreas tan diversas como la medicina, donde ayuda a diagnosticar y tratar enfermedades, hasta el ámbito empresarial, donde optimiza procesos, incrementa la eficiencia y facilita la toma de decisiones (Russell & Norvig, 2021). Además, en el contexto social, la inteligencia artificial ha transformado la forma en que interactuamos y nos comunicamos. Sin embargo, el enfoque de este proyecto se centra en el área específica del Procesamiento del Lenguaje Natural (NLP), un subcampo de la inteligencia artificial que busca mejorar la interacción entre humanos y máquinas a través del lenguaje.

Uno de los avances más significativos en el campo del NLP ha sido el desarrollo de modelos preentrenados, en particular los modelos tipo Transformer, introducidos en 2017 (Vaswani, y otros, 2017). Estos modelos han demostrado capacidades sobresalientes para procesar, generar y comprender el lenguaje humano en sus diversas formas, sin importar el idioma. Los modelos Transformer son entrenados a partir de enormes conjuntos de datos textuales, que abarcan una amplia variedad de temas y contextos, con el objetivo de cumplir una serie de funciones generales relacionadas con el lenguaje. Una vez entrenados en este contexto amplio, estos modelos pueden ser ajustados para tareas más específicas mediante un proceso denominado fine-tuning, una técnica que permite adaptar el conocimiento general del modelo a aplicaciones concretas y especializadas.

El fine-tuning adquiere una relevancia particular cuando se utilizan técnicas de Parameter Efficient fine-tuning (PEFT), desarrolladas en 2021 (Houlsby & al., 2021). La ventaja de PEFT radica en que permite modificar únicamente los parámetros específicos necesarios para una tarea concreta, sin requerir un reentrenamiento completo del modelo. Este enfoque ahorra una gran cantidad de recursos computacionales y hace posible la creación de modelos personalizados y adaptados sin los altos costes asociados a entrenar un modelo completo desde cero (Dettmers, Lewis, Shleifer, & Zettlemoyer, 2021). Así, el fine-tuning eficiente permite que los modelos alcancen un alto grado de especialización y precisión en tareas específicas, empleando solo una fracción de los recursos que se requerirían de otra manera.

A pesar de la utilidad del fine-tuning, este proceso por sí solo no garantiza que el modelo sea capaz de responder adecuadamente a preguntas sobre temas o datos que desconoce. Para abordar este desafío, se recurre al Aprendizaje por Refuerzo, una técnica en la que el modelo es entrenado para no ofrecer respuestas incorrectas o inventadas cuando enfrenta preguntas o temas que van más allá de su conocimiento original (Christiano, y otros, 2017). En este enfoque, el modelo recibe retroalimentación y se ajusta en función de sus interacciones, con el objetivo de mejorar continuamente su capacidad para ofrecer respuestas precisas y fiables.

Este proyecto tiene como objetivo explorar estas técnicas avanzadas en un modelo de lenguaje preentrenado, aplicando todas estas metodologías para crear un sistema más robusto y adaptado. Para ello, se utilizará un conjunto de datos personalizado, seguido de un proceso de fine-tuning para ajustar el modelo a las necesidades específicas de este conjunto de datos. Posteriormente, se empleará Aprendizaje por Refuerzo como fase final de entrenamiento, optimizando la capacidad del modelo para interactuar y responder adecuadamente en el contexto planteado (Brown, y otros, Language models are few-shot learners, 2020). Con esto, se busca obtener un modelo que no solo cumpla con la tarea específica para la que se ha ajustado, sino que también ofrezca respuestas coherentes y fundamentadas, incrementando así la confianza y efectividad en sus respuestas.

1.2 Motivación

Este proyecto surge del deseo de explorar y comprender en profundidad el proceso de ajuste de modelos de lenguaje preentrenados, adaptándolos a tareas específicas mediante técnicas avanzadas de fine-tuning y Aprendizaje por Refuerzo. La evolución reciente en el campo del Procesamiento del Lenguaje Natural (NLP) ha llevado a notables avances en la capacidad de los modelos para comprender y generar lenguaje humano de manera efectiva (Devlin, Chang, Lee, & Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019). Sin embargo, a pesar de estos avances significativos, persiste un desafío importante: la necesidad de desarrollar modelos que no solo sean precisos en sus respuestas, sino que también sean capaces de evitar la generación de respuestas incorrectas o inventadas ante preguntas que caen fuera de su conocimiento (Marcus & Davis, 2019). Este problema es crítico en aplicaciones del mundo real, donde la precisión y la confiabilidad de la información proporcionada son esenciales para el éxito y la seguridad de los sistemas que emplean inteligencia artificial.

Desde una perspectiva académica, este proyecto ofrece una oportunidad valiosa para investigar el diseño y desarrollo de modelos eficientes en tareas concretas. Al centrar la atención en el ajuste de modelos de lenguaje preentrenados, se busca comprender cómo estos modelos pueden ser optimizados para responder adecuadamente en situaciones específicas. Para este propósito, se ha utilizado un conjunto de datos que se extrae de noticias actuales. Este conjunto no solo proporciona información relevante y reciente, sino que también se ha utilizado para generar múltiples preguntas junto con sus respectivas respuestas, con el objetivo de evaluar cómo los modelos de lenguaje gestionan y responden a información nueva. Esta tarea es especialmente interesante porque permite observar la capacidad del modelo para generalizar su conocimiento y aplicar lo aprendido a contextos diferentes, un aspecto fundamental en el desarrollo de sistemas de inteligencia artificial más robustos y adaptables.

La combinación de técnicas de fine-tuning y Aprendizaje por Refuerzo se convierte en un aspecto crucial de este proyecto. El fine-tuning permite que un modelo preentrenado, que ha aprendido de grandes volúmenes de datos textuales, se ajuste a tareas específicas con un mínimo de reentrenamiento de sus parámetros (Raffel, y otros, 2020). Esto no solo optimiza el tiempo y los recursos computacionales requeridos, sino que también mejora la relevancia y precisión del modelo en la tarea deseada. Sin embargo, el fine-tuning por sí solo no es suficiente para garantizar que el modelo maneje adecuadamente situaciones en las que se le presentan preguntas que no ha encontrado previamente. Aquí es donde entra en juego el Aprendizaje por Refuerzo, que proporciona un enfoque para entrenar al modelo a evitar la generación de respuestas incorrectas o inventadas. A través de la retroalimentación constante, el modelo puede aprender a discernir cuándo no debe responder o a buscar información adicional en lugar de inventar datos (Ziegler, y otros, 2019).

Una vez que se haya completado el proceso de ajuste, el proyecto se enfocará en estudiar las activaciones neuronales del modelo. Este análisis tiene como objetivo arrojar luz sobre qué neuronas o capas específicas dentro del modelo tienen un mayor impacto en la comprensión y generación de texto (Vig & Belinkov, 2019). Este nivel de análisis no solo contribuye al entendimiento de cómo los modelos procesan la información, sino que también ofrece pistas sobre cómo se podrían mejorar futuras arquitecturas y técnicas de ajuste. Al investigar las interacciones neuronales que conducen a respuestas exitosas, se espera proporcionar información útil para el desarrollo de modelos de lenguaje más eficientes y precisos.

El objetivo de este proyecto es doble: por un lado, busca implementar y evaluar técnicas de ajuste efectivas para mejorar la capacidad de los modelos de lenguaje en tareas específicas, y por otro, pretende contribuir al conocimiento académico en el área de NLP, al explorar las dinámicas internas de los modelos de lenguaje y cómo estos pueden ser optimizados. A través de la combinación de fine-tuning y Aprendizaje por Refuerzo, se espera lograr un modelo que no solo responda de manera precisa y confiable, sino que también muestre un profundo entendimiento de la información presentada (Stiennon, y otros, 2020), ofreciendo así una base sólida para futuros desarrollos en el campo de la inteligencia artificial.

1.3 Objetivos

El principal objetivo de este proyecto es ajustar un modelo de lenguaje para que rinda adecuadamente en tareas de comprensión y generación de texto, especialmente cuando se enfrenta a datos que no estaban presentes en su entrenamiento original. Este objetivo principal puede desglosarse en varias etapas clave que son fundamentales para el éxito del proyecto:

- **Implementar un software de scraping**

Esta fase inicial implica el desarrollo de una herramienta de scraping que permitirá la obtención de datos necesarios para la creación de un corpus (Mitchell, 2018). Este corpus será la base sobre la cual se entrenará el modelo. Asimismo, se generarán dos conjuntos de preguntas y sus respectivas respuestas, que se diseñarán específicamente para entrenar al modelo de manera efectiva. La importancia de esta etapa radica en la calidad y relevancia de los datos recolectados, ya que un corpus bien construido es crucial para el aprendizaje del modelo (Jurafsky & Martin, 2021).

- **Realizar fine-tuning al modelo**

En esta etapa, se aplicarán técnicas de fine-tuning utilizando métodos de fine-tuning Eficiente en Parámetros (PEFT) (Houlsby, y otros, 2019). El objetivo de este ajuste es permitir que el modelo genere texto coherente y relevante basado en el nuevo conjunto de datos, adaptándose así a las características específicas del corpus creado. Este proceso de fine-tuning es esencial, ya que optimiza el modelo preentrenado para tareas concretas, aumentando su efectividad en situaciones del mundo real.

Además, se empleará el Aprendizaje por Refuerzo (RL) (Christiano, y otros, 2017) como una herramienta adicional para refinar los resultados obtenidos durante el fine-tuning. El aprendizaje por Refuerzo permite a los LLM adaptar sus respuestas en tiempo real basándose en la retroalimentación recibida, optimizando así el rendimiento del modelo para tareas específicas y mejorando su capacidad de generalización. Este proceso complementario asegura que las interacciones con el usuario sean más eficientes, relevantes y personalizadas, contribuyendo a una experiencia de uso más satisfactoria en aplicaciones prácticas.

- **Evaluar el rendimiento del modelo**

Finalmente, se evaluará el rendimiento del modelo tras las etapas de fine-tuning y el entrenamiento mediante Aprendizaje por Refuerzo. Esta evaluación se realizará a través de un conjunto de pruebas que incluirán preguntas con sus respectivas respuestas relacionadas con el corpus empleado (Liang & al., 2021). Este paso es crucial, ya que proporcionará métricas concretas sobre la eficacia del modelo en tareas de comprensión y generación de texto, así como una visión clara sobre áreas que podrían requerir ajustes adicionales.

A través de estas etapas, se busca no solo optimizar un modelo de lenguaje, sino también contribuir al entendimiento académico de su funcionamiento interno, con el objetivo de desarrollar sistemas de inteligencia artificial más eficientes y confiables en el campo del Procesamiento del Lenguaje Natural (Belinkov & Glass, 2017).

2. ESTADO DEL ARTE

El rendimiento de los Grandes Modelos del Lenguaje (LLM) ha aumentado considerablemente en los últimos años, lo que ha tenido un impacto significativo en diversas áreas, tales como la generación de texto, el entendimiento del lenguaje natural, la traducción automática y la asistencia profesional, entre otros. Este avance ha revolucionado no solo el ámbito académico, sino también aplicaciones prácticas en la industria (Brown T. , Mann, Ryder, Subbiah, & Kaplan, 2020), donde la automatización y la mejora en la interacción humano-computadora se han vuelto más efectivas gracias a estos modelos.

2.1 Modelos Preentrenados del Lenguaje

Como se comentó anteriormente, los modelos preentrenados han cambiado por completo la creación de soluciones para tareas de Procesamiento del Lenguaje Natural (NLP). Este tipo de modelos se basa en la arquitectura Transformer, que ha sido el epicentro de los avances en el campo del NLP. La introducción de esta arquitectura ha permitido un tratamiento más eficiente de las secuencias de texto (Vaswani, y otros, 2017), superando las limitaciones de modelos anteriores.

Estos modelos están diseñados para tener la capacidad de procesar secuencias de texto a través de un mecanismo de atención, el cual proporciona al modelo la habilidad de ponderar las diferentes partes del texto de entrada. Esta característica les permite obtener una mejor comprensión del contexto, ya que pueden identificar qué palabras o frases son más relevantes para la tarea en cuestión. La atención permite que los modelos no solo se centren en la secuencia de entrada de manera lineal, sino que también consideren interacciones complejas entre diferentes elementos de texto.

BERT (Bidirectional Encoder Representations from Transformers) fue uno de los primeros modelos preentrenados relevantes (Devlin, Chang, Lee, & Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019). Se trata de un modelo bidireccional que ha sido entrenado para predecir las palabras faltantes en una secuencia de texto, lo que le permite capturar el contexto de una manera más profunda que los modelos unidireccionales. Este enfoque bidireccional es fundamental para entender la relación entre palabras en contextos complejos, lo que resulta en una mejora notable en tareas de clasificación, análisis de sentimientos y respuesta a preguntas.

Otro modelo de gran relevancia fue GPT (Generative Pre-trained Transformer) (Radford, y otros, 2019), que utiliza una metodología autoregresiva para generar texto a partir de una secuencia de entrada. A diferencia de BERT, que se centra en entender el contexto, GPT se destaca en la generación de texto coherente y creativo, lo que lo hace ideal para aplicaciones que requieren redacción y generación de contenido.

Estos tipos de modelos no solo son poderosos en su forma preentrenada, sino que también pueden ser ajustados mediante técnicas de fine-tuning para tareas específicas. Este proceso de ajuste permite que los modelos aprovechen su entrenamiento previo en grandes cantidades de datos y se adapten a dominios concretos o tareas particulares, mejorando así su rendimiento en aplicaciones específicas. A medida que se desarrolla la investigación en este campo, se espera que la capacidad de estos modelos para entender y generar lenguaje humano continúe avanzando, abriendo nuevas oportunidades y aplicaciones en el mundo real.

2.2 Fine-tuning y Técnicas PEFT

El método Fine-Tuning es indispensable para ajustar un modelo del lenguaje preentrenado a una tarea específica, como responder preguntas, generar texto o clasificar texto. Los modelos preentrenados, que han sido entrenados en grandes volúmenes de datos, disponen de un conocimiento sólido del lenguaje, lo que les permite captar patrones y relaciones complejas dentro de la estructura del lenguaje. Sin embargo, para maximizar la eficacia de un modelo en una tarea concreta, el Fine-Tuning permite adaptar este conocimiento general a un conjunto de datos específico, mejorando así su capacidad para abordar tareas más concretas y especializadas.

A pesar de los beneficios que proporciona el Fine-Tuning, este proceso puede resultar muy costoso en términos de recursos computacionales, especialmente cuando se trata de ajustar un modelo de lenguaje completo. Dado que los modelos preentrenados ya tienen un amplio conocimiento que puede ser relevante para muchas aplicaciones, realizar un Fine-Tuning completo de todos los parámetros puede ser ineficiente y, en muchas ocasiones, innecesario. La solución a esta problemática se encuentra en las técnicas de Fine-Tuning Eficiente de Parámetros (PEFT) (Houlsby, y otros, 2019), que permiten ajustar únicamente una pequeña fracción de los parámetros del modelo. Esto no solo mejora de manera significativa el tiempo de entrenamiento, sino que también reduce considerablemente el coste computacional asociado, todo ello sin comprometer el rendimiento en la tarea específica para la que el modelo ha sido entrenado.
Las principales técnicas de PEFT son:

- **Adapter Layers:**

Esta técnica mantiene congelados los parámetros originales del modelo, lo que significa que no se modifican durante el proceso de ajuste. En lugar de ello, se añaden nuevas capas al modelo que son específicas para la tarea requerida (Pfeiffer, Kamath, Rücklé, Cho, & Gurevych, AdapterHub: A Framework for Adapting Transformers, 2020). Esto permite que el modelo preentrenado conserve su conocimiento general del lenguaje, al mismo tiempo que se adapta a necesidades concretas y específicas. Por ejemplo, un modelo que ha sido preentrenado en un amplio corpus de texto general puede ser ajustado para que responda a preguntas sobre un conjunto de datos especializado, como un dominio médico o técnico. De este modo, se facilita la implementación de nuevas tareas sin necesidad de realizar un reentrenamiento completo del modelo, lo que ahorra tiempo y recursos.

- **LoRA (Low-Rank Adaptation):**

Esta técnica factoriza las matrices de pesos del modelo en dos matrices de bajo rango. Este enfoque reduce la complejidad de las actualizaciones de parámetros, lo que significa que se requiere menos computación para ajustar el modelo. Al realizar este ajuste, se disminuye significativamente la cantidad de cálculos requeridos durante el proceso de entrenamiento, permitiendo una mayor eficiencia en el uso de recursos computacionales. Este método ha demostrado ser efectivo en la adaptación de modelos grandes a tareas específicas (Hu, y otros, LoRA: Low-rank adaptation of large language models, 2021), brindando una alternativa viable para aquellos que operan en entornos con recursos limitados.

- **Prompt Tuning**

Esta técnica implica un ajuste más ligero en comparación con otras estrategias, ya que únicamente se ajustan los vectores de entrada que modifican la salida del modelo. En otras palabras, el modelo permanece en gran medida intacto y solo se realiza una modificación en la manera en que se le presentan las solicitudes. Este enfoque simplifica considerablemente el proceso de Fine-Tuning, permitiendo que el modelo se adapte a diferentes contextos o tareas sin la necesidad de cambiar su arquitectura interna (Lester, Al-Rfou, & Constant, 2021). Esto es especialmente útil en escenarios donde el tiempo de ajuste y el uso de recursos deben ser optimizados. La eficiencia de este método lo hace atractivo para aplicaciones en las que se requiere rapidez y flexibilidad.

Mediante la implementación de estas técnicas, se logra realizar una adaptación eficiente de modelos preentrenados a tareas concretas, incluso en entornos donde los recursos computacionales son limitados. Este avance no solo abre la puerta a una mayor versatilidad en la aplicación de modelos de lenguaje, sino que también permite su uso en una variedad más amplia de situaciones y problemas en el ámbito del Procesamiento del Lenguaje Natural (NLP). A medida que la demanda de soluciones basadas en inteligencia artificial continúa creciendo, estas técnicas de ajuste se vuelven cada vez más relevantes para facilitar la implementación de modelos de lenguaje en entornos del mundo real. La capacidad de ajustar modelos de manera eficiente a tareas específicas no solo mejora el rendimiento de los modelos, sino que también contribuye a la sostenibilidad del desarrollo de aplicaciones de inteligencia artificial, permitiendo que sean más accesibles y utilizables en una variedad de contextos.

2.3 Aprendizaje por Refuerzo en NLP

A pesar de la capacidad de los modelos ajustados mediante Fine-Tuning para llevar a cabo tareas concretas, una de las principales limitaciones que presentan es su tendencia a inventar respuestas cuando no tienen suficiente información sobre el contexto de una pregunta. Esta problemática puede comprometer no solo la calidad y la utilidad de las respuestas generadas, sino también la confianza que los usuarios depositan en estos sistemas. Por lo tanto, resalta la necesidad de aplicar métodos adicionales para mejorar el rendimiento de estos modelos en situaciones donde la información es escasa o insuficiente. Una solución prometedora para abordar esta limitación es la implementación de técnicas de Aprendizaje por Refuerzo (RL), que tienen como objetivo entrenar al modelo para evitar este comportamiento no deseado y garantizar respuestas más precisas y coherentes en contextos variados.

El Aprendizaje por Refuerzo es una rama del Machine Learning (ML) que se centra en entrenar a un agente para que aprenda a tomar decisiones a través de la interacción con un entorno específico. En este proceso, el agente recibe recompensas o castigos en función de las acciones que realiza, lo que le permite aprender de la retroalimentación que recibe y mejorar su rendimiento a lo largo del tiempo. Este enfoque se basa en la premisa de que, al maximizar las recompensas acumuladas, el agente puede desarrollar políticas de comportamiento efectivas que lo ayuden a alcanzar sus objetivos, adaptándose así a diferentes situaciones y contextos en los que se encuentra.

En el ámbito del Procesamiento del Lenguaje Natural (NLP), el aprendizaje por refuerzo se puede aplicar para entrenar un modelo ya ajustado de tal manera que el modelo pueda mejorar su desempeño en tareas específicas, adaptándose a diferentes contextos o entradas. Esto permite que el modelo tome decisiones más informadas y efectivas durante la generación de texto o la clasificación de lenguaje natural, lo que puede llevar a un rendimiento superior y una mayor precisión en las tareas de NLP. Este proceso es especialmente relevante en contextos donde la precisión y la veracidad de la información son cruciales, como en aplicaciones médicas, legales o de atención al cliente. La capacidad del modelo para generar respuestas fundamentadas, en lugar de respuestas inventadas, no solo mejora la experiencia del usuario, sino que también contribuye a establecer la confianza en el sistema. Esta aplicación del Aprendizaje por Refuerzo permite al modelo no solo mejorar su precisión en la generación de respuestas, sino también adquirir una mayor comprensión del contexto en el que se están formulando las preguntas. Esto es esencial para mejorar la interacción entre el modelo y el usuario, ya que las respuestas deben alinearse con las expectativas y necesidades de quienes utilizan el sistema.

Uno de los métodos de Aprendizaje por Refuerzo más utilizados para entrenar Grandes Modelos de Lenguaje (LLM) es el PPO (Proximal Policy Optimization) (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017). PPO se caracteriza por ser eficiente y efectivo en la optimización de políticas en entornos continuos, lo que lo convierte en una opción ideal para aplicaciones en NLP. Este enfoque permite al modelo adaptarse a la variabilidad del lenguaje humano, donde las preguntas y respuestas pueden tener múltiples formas de interpretación.

Un enfoque particularmente interesante en la aplicación del Aprendizaje por Refuerzo en LLM es el uso de Aprendizaje por Refuerzo con Retroalimentación Humana (RLHF). Este método combina la retroalimentación humana con el aprendizaje por refuerzo tradicional para entrenar modelos como GPT (Ouyang, y otros, 2022), ajustando las respuestas generadas para que se alineen mejor con las expectativas de los usuarios. Al incorporar la evaluación y los comentarios de los humanos en el proceso de entrenamiento, se logra un refinamiento adicional en la calidad de las respuestas del modelo. Este enfoque no solo ayuda a reducir la tendencia a inventar información, sino que también permite que el modelo se adapte a las preferencias y requerimientos específicos de los usuarios, mejorando así su utilidad en aplicaciones prácticas.

Además, el uso de retroalimentación humana en el entrenamiento del modelo contribuye a una mayor comprensión de los matices del lenguaje y las sutilezas culturales que pueden ser críticas en la generación de texto. La interacción entre los modelos y los evaluadores humanos ayuda a identificar áreas donde el modelo puede estar fallando, permitiendo ajustes más informados y orientados al usuario. Esto se traduce en un proceso de aprendizaje más robusto y adaptable, en el que el modelo no solo aprende de los datos, sino también de la experiencia y la intuición humana.

La integración de técnicas de Aprendizaje por Refuerzo en el ámbito del NLP ofrece un enfoque innovador y efectivo para abordar las limitaciones de los modelos de lenguaje ajustados. Al entrenar modelos para evitar la invención de respuestas y optimizar su rendimiento a través de métodos como PPO y RLHF, se abre un nuevo camino hacia la creación de sistemas de procesamiento de lenguaje natural más robustos y eficientes. Esto no solo mejora la calidad de las interacciones con los usuarios, sino que también contribuye al avance de la inteligencia artificial en general, facilitando su implementación en una variedad de aplicaciones del mundo real, desde asistentes virtuales y chatbots hasta sistemas de recomendación y generación de contenido automatizado. Este enfoque integral promete mejorar la forma en que interactuamos con la tecnología, haciendo que las respuestas sean más relevantes y útiles para las necesidades específicas de cada usuario.

3. JUSTIFICACIÓN DE COMPETENCIAS ESPECÍFICAS

- **ED1: Capacidad para conocer los fundamentos, paradigmas y técnicas propias de los sistemas inteligentes y analizar, diseñar y construir sistemas, servicios y aplicaciones informáticas que utilicen dichas técnicas en cualquier ámbito de aplicación.**

A lo largo de todo el proyecto, se aplica esta competencia mediante la implementación de técnicas de Machine Learning, mas concretamente en técnicas de Fine-Tuning y RL de un LLM preentrenado con el objetivo de ajustarlo a una tarea para la que originalmente no fue entrenado. Se aplicaron técnicas PEFT avanzadas como LoRA para optimizar el entrenamiento, además de técnicas SFTTrainer para entrenamiento basado en instrucciones.

- **ED2: Capacidad para adquirir, obtener, formalizar y representar el conocimiento humano en una forma computable para la resolución de problemas mediante un sistema informático en cualquier ámbito de aplicación, particularmente los relacionados con aspectos de computación, percepción y actuación en ambientes o entornos inteligentes.**

En el proyecto se obtuvo y procesó un conjunto significativo de datos a través de técnicas de scraping, adquiriendo dichos datos de fuentes en línea. Posteriormente se formalizó y estructuró el conjunto de datos en estructuras computables, como JSON. Mediante este proceso se creó un corpus de información que se empleó en posteriores técnicas de Machine Learning.

- **ED3 Capacidad para conocer y desarrollar técnicas de aprendizaje computacional y diseñar e implementar aplicaciones y sistemas que las utilicen, incluyendo las dedicadas a extracción automática de información y conocimiento a partir de grandes volúmenes de datos.**

Se desarrolló un software de scraping con el fin de automatizar la recolección de datos de fuentes web, generando así un corpus representativo. Después, dicho corpus se utilizó en técnicas de Machine Learning. Todo este proceso de extracción y entrenamiento de LLM se adapta a esta competencia específica.

- **ED4 Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software en ámbitos de aplicación de la Inteligencia Artificial en Ciencia e Ingeniería de Datos**

Durante el proyecto, se identifican y solucionan diversas problemáticas relacionadas con el procesamiento de datos, como la extracción, limpieza y estructuración de datos obtenidos de recursos web. Además, se implementa una solución de Fine-Tuning de LLM en la que se incluye verificación y evaluación del modelo pre y post ajuste.

- **ED6 Capacidad para tener un conocimiento profundo de los principios fundamentales y modelos utilizados en Ciencia de Datos, particularmente las relacionadas con el análisis, predicción y prospectiva de grandes volúmenes de datos**

Durante el proyecto, se emplearon conceptos fundamentales de la Ciencia de Datos, tales como el análisis y procesamiento de grandes conjuntos de datos. Dichos datos fueron analizados y procesados para ser utilizados en posteriores tareas de Machine Learning y se evaluó la capacidad predictiva de los modelos ajustados.

- **ED8 Capacidad de concebir sistemas, aplicaciones y servicios basados en tecnologías de la información y las comunicaciones para Ciencia e Ingeniería de Datos, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación móvil.**

Se implementó un sistema automatizado que emplea tecnologías de la información para la extracción de datos web o generación de contenido basado en datos recolectados. El sistema no únicamente integra múltiples tecnologías como scraping, NLP o ML, sino que además facilita la aplicación de estos resultados.

4. METODOLOGÍA Y PLANIFICACIÓN

El proyecto seguirá la metodología CRIDP-DM (Cross-Industry Standard Process for Data Mining), que es un marco estandarizado ampliamente utilizado en proyectos de análisis y minería de datos. Esta metodología es valorada en la comunidad de Ciencia de Datos porque proporciona un enfoque estructurado que simplifica la organización y la ejecución de proyectos, permitiendo un desarrollo más fluido y eficiente. Al seguir esta metodología, se asegura que cada etapa del proceso se lleve a cabo de manera ordenada, facilitando la identificación de problemas y la implementación de soluciones efectivas. En este sentido, la metodología CRIDP-DM abarca varias fases que son esenciales para garantizar el éxito del proyecto. A continuación, se detalla la implementación de las diferentes fases de esta metodología en el contexto del presente proyecto:

4.1 Comprensión del Negocio

La primera etapa de este proceso es la Comprensión del Negocio, donde se persigue un entendimiento claro y conciso de los requisitos y objetivos del proyecto. Este paso es fundamental, ya que sienta las bases para todas las decisiones que se tomarán a lo largo del desarrollo. En este caso particular, el objetivo es ajustar y aumentar el rendimiento de un Modelo de Lenguaje Grande (LLM) para que se desempeñe adecuadamente en tareas específicas. Para lograrlo, es necesario tener en cuenta diversas consideraciones estratégicas. Este proceso se desglosa en los siguientes puntos clave que guiarán la etapa inicial:

- **Elegir un LLM adecuado**

La primera decisión crítica es seleccionar un modelo de lenguaje que cumpla con el propósito de ser adaptado para realizar nuevas tareas a través de los procesos de Fine-Tuning y Aprendizaje por Refuerzo (RL). La elección de un modelo es crucial, ya que cada LLM tiene características únicas, entrenamientos previos y capacidades específicas que afectan su rendimiento en tareas concretas. Un LLM bien elegido puede ser la diferencia entre un proyecto exitoso y uno que no cumple con las expectativas. Por lo tanto, se realizará un análisis detallado de los diferentes modelos disponibles, considerando sus fortalezas y debilidades.

- **Implementar un software automatizado**

A continuación, se desarrollará un software que automatice la recolección, limpieza, procesamiento y estructuración de datos. Esta herramienta es esencial para generar un corpus significativo, que servirá como base para el entrenamiento del modelo. La calidad y relevancia de los datos de entrenamiento son factores determinantes en el éxito del ajuste del modelo. Un corpus bien estructurado y representativo garantizará que el modelo tenga suficiente información para aprender y adaptarse a las tareas específicas. Además, la automatización de este proceso no solo ahorra tiempo y recursos, sino que también minimiza los errores que podrían surgir de un manejo manual. Este enfoque sistemático asegura una mayor consistencia y confiabilidad en los datos.

- **Obtener un conjunto de preguntas y respuestas**

Por último, se establecerá un proceso metódico para generar un conjunto de preguntas de calidad junto con sus respectivas respuestas, utilizando la información extraída del corpus obtenido. Este conjunto de datos es vital para el entrenamiento y evaluación del modelo, ya que permitirá medir su capacidad para comprender y generar texto de manera efectiva en función de la información disponible. La relevancia y la calidad de estas preguntas y respuestas son cruciales para garantizar que el modelo esté bien preparado para las tareas específicas que se le asignarán. Además, este paso proporciona la oportunidad de evaluar cómo el modelo maneja información nueva y cómo responde a preguntas que no han sido parte de su entrenamiento original. Esta evaluación es fundamental para determinar la efectividad del ajuste y el Fine-Tuning realizado.

A través de esta fase de comprensión del negocio, se sentarán las bases para el desarrollo del proyecto, alineando las expectativas del equipo con los objetivos que se desean alcanzar. La claridad en los requisitos y metas permitirá que las siguientes fases de la metodología CRIDP-DM se ejecuten con mayor eficacia, asegurando que cada etapa del proceso se oriente hacia la consecución de los resultados deseados. Además, un entendimiento sólido de los objetivos del proyecto facilitará la identificación de métricas adecuadas para evaluar el rendimiento del modelo ajustado, lo que permitirá realizar ajustes y mejoras a lo largo del desarrollo. Esta etapa inicial no debe ser subestimada, ya que establece el tono y la dirección para el resto del proyecto, garantizando que todos los esfuerzos estén alineados y orientados hacia un mismo propósito, maximizando así las posibilidades de éxito en la implementación del modelo de lenguaje ajustado.

4.2 Comprensión de los Datos

Una vez que se han definido claramente los objetivos del proyecto, el siguiente paso fundamental es centrarse en entender los datos obtenidos, ya que la calidad y la adecuación de estos datos son cruciales para el éxito del proyecto. En este caso particular, los datos fueron obtenidos mediante técnicas de scraping a recursos web, lo que implica la extracción automatizada de información de diversas páginas y plataformas en línea. Este enfoque permite acceder a grandes volúmenes de datos que pueden ser relevantes para el análisis y el entrenamiento del modelo. Sin embargo, es esencial realizar un análisis exhaustivo de estos datos para asegurarse de que cumplan con los estándares requeridos para su uso en procesos posteriores.

En esta fase, se realiza un análisis detallado de toda la información relevante contenida en los datos. Esto incluye evaluar aspectos como la estructura de los datos, los metadatos asociados, la relevancia de la información extraída, así como la calidad y completitud de los datos. La estructura se refiere a cómo están organizados los datos, es decir, qué formatos se han utilizado, cómo se distribuyen los diferentes campos y cómo se relacionan entre sí. Comprender la estructura es esencial para poder manipular y procesar los datos adecuadamente en las etapas posteriores.

Los metadatos son otra dimensión crítica que considerar. Estos son datos que proporcionan información sobre otros datos, como su origen, fecha de creación, y cualquier otra información contextual que pueda ser útil para interpretar y utilizar los datos de manera efectiva. Evaluar los metadatos ayuda a comprender el contexto en el que se recolectaron los datos y puede influir en la decisión de su inclusión o exclusión en el análisis.

La relevancia de los datos es igualmente importante. En esta etapa, se debe determinar qué tan bien los datos extraídos cumplen con los objetivos establecidos en la fase anterior. Esto implica analizar si la información obtenida se alinea con las necesidades del proyecto y si es adecuada para entrenar el modelo en función de las tareas específicas que se desean abordar. Si los datos no son relevantes, podrían desvirtuar el proceso de aprendizaje y generar resultados no deseados.

La calidad de los datos también debe ser evaluada con rigor. Esto incluye comprobar si hay errores, duplicados, o información desactualizada que pueda afectar el rendimiento del modelo. Una alta calidad de datos es fundamental, ya que los modelos de lenguaje son muy sensibles a la calidad de la información con la que se entrenan. La identificación y corrección de problemas de calidad en los datos es un paso esencial para asegurar que el modelo pueda generalizar correctamente a partir de la información que se le proporciona.

Finalmente, la completitud de los datos es un factor que no se puede pasar por alto. Es esencial asegurarse de que se ha obtenido suficiente información para abordar las preguntas o tareas específicas que se pretenden realizar con el modelo. Datos incompletos pueden llevar a un ajuste ineficaz del modelo, limitando su capacidad para entender y generar texto de manera efectiva.

Una vez completado este análisis exhaustivo de los datos, el equipo del proyecto estará en una mejor posición para tomar decisiones informadas sobre el preprocesamiento que será necesario aplicar. Esto incluye determinar qué datos se emplearán en los procesos posteriores, así como identificar cualquier transformación o limpieza que deba realizarse para preparar los datos de manera óptima para el entrenamiento del modelo. Este paso es crucial, ya que el éxito de las fases posteriores del proyecto dependerá en gran medida de la calidad y adecuación de los datos utilizados en el proceso de modelado. En resumen, esta fase de comprensión de los datos es esencial para establecer una base sólida sobre la cual construir el resto del proyecto, garantizando que cada decisión esté fundamentada en un análisis riguroso y completo de la información disponible.

4.3 Preparación de los Datos

En esta fase crucial del proyecto, se llevan a cabo una serie de tareas fundamentales que incluyen la limpieza, filtrado, transformación y estructuración de datos, entre otras. Este paso es absolutamente imprescindible para garantizar que los datos cumplan con los requisitos necesarios de calidad para ser utilizados en los entrenamientos del modelo. A continuación, se describen de manera detallada los pasos involucrados en esta preparación de datos:

- **Limpieza de Textos**

La primera tarea que se realiza es la limpieza de textos, que implica la eliminación de datos irrelevantes que no aportan valor al análisis o al entrenamiento del modelo. Este proceso es un elemento crítico, ya que cualquier tipo de información irrelevante o errónea puede afectar negativamente el rendimiento del modelo al introducir confusión o ruido en el conjunto de datos. La limpieza de textos puede incluir múltiples actividades, tales como la eliminación de caracteres especiales, enlaces web, información redundante, así como palabras o frases que no tienen relación con el contexto general. Se pueden aplicar técnicas de normalización para convertir todas las letras a minúsculas y eliminar espacios en blanco innecesarios. También se considera eliminar elementos no textuales como imágenes o gráficos que pueden haber sido capturados durante el proceso de scraping.

Además, es importante considerar la eliminación de stop words, que son palabras comunes que no aportan significado por sí solas (como "y", "el", "en", etc.). Sin embargo, la decisión de eliminar estas palabras debe ser contextual, ya que, en algunos casos, pueden tener importancia en la comprensión del texto. Este proceso de limpieza es esencial, ya que ayuda a obtener textos que sean coherentes y relevantes para las tareas específicas que se desean realizar, lo que facilita la comprensión del contenido por parte del modelo. Al final de este proceso, se espera contar con un conjunto de textos depurado y enfocado, lo que ayuda a mejorar el desempeño general del modelo al evitar distracciones provocadas por datos no pertinentes.

- **Estructuración de Datos**

Una vez que se han limpiado los textos, el siguiente paso es la estructuración de datos. En esta fase, se almacenan los textos y sus respectivos metadatos en ficheros de texto plano de manera organizada. Este proceso de estructuración es fundamental, ya que permite que los datos sean fácilmente accesibles y manejables durante el proceso de entrenamiento. Se utilizarán formatos estándar para almacenar la información, como JSON o CSV, que permiten una fácil interpretación y manipulación de los datos. Al organizar la información de manera sistemática, se facilita el acceso a los datos durante las fases de modelado y análisis, lo que es esencial para la eficiencia del flujo de trabajo.

Además, contar con metadatos asociados a cada texto proporciona contexto adicional, lo que puede ser útil para el posterior análisis del rendimiento del modelo. Por ejemplo, los metadatos pueden incluir la fecha de publicación del contenido, la fuente, el autor, o categorías temáticas que pueden ser de interés en análisis posteriores. Estos elementos pueden servir para realizar análisis de sensibilidad, permitiendo al modelo adaptarse mejor a diferentes contextos o fuentes de información. La estructuración adecuada de datos no solo facilita el acceso, sino que también optimiza el rendimiento del modelo al asegurar que el mismo tenga toda la información relevante a su disposición.

- **Generación de Corpus**

A partir de los datos previamente estructurados, se procede a la generación de un corpus. Este corpus es un conjunto de datos específico que se utiliza para el entrenamiento del modelo. La creación de un corpus implica seleccionar cuidadosamente las muestras de texto que serán representativas del dominio específico que se está abordando. Es crucial que el corpus esté bien diseñado y sea representativo de la información que se desea que el modelo aprenda. Esta selección debe incluir una variedad de ejemplos que reflejen la diversidad del lenguaje y los contextos en los que se aplicará el modelo.

La calidad del corpus influye directamente en la capacidad del modelo para generalizar y manejar tareas relacionadas con la comprensión y generación de texto. Un corpus bien construido puede facilitar un aprendizaje más efectivo y una mejor adaptación a las tareas específicas para las que se está entrenando el modelo. El corpus debe abarcar tanto ejemplos positivos como negativos, asegurando así que el modelo no solo aprenda a reconocer patrones, sino también a identificar excepciones o errores. Esta atención al detalle en la creación del corpus es fundamental para maximizar el rendimiento y la utilidad del modelo.

- **Generación de Conjuntos de Preguntas y Respuestas**

Finalmente, en base al corpus creado, se lleva a cabo la generación de conjuntos de preguntas y respuestas. En este proceso, se desarrolla un conjunto de preguntas junto con sus respuestas correspondientes que se utilizarán para el entrenamiento del modelo, así como otro conjunto que se empleará para realizar pruebas o tests. Estos conjuntos de preguntas y respuestas son esenciales para evaluar la capacidad del modelo para comprender y generar texto en respuesta a consultas específicas.

La formulación de preguntas debe ser variada y representar diferentes tipos de interrogantes, desde preguntas de sí/no hasta preguntas más complejas que requieran un análisis más profundo del contenido. Además, es importante que las respuestas sean precisas y se correspondan directamente con el contenido del corpus. La creación de un conjunto de pruebas permite medir el rendimiento del modelo de manera objetiva, proporcionando una referencia clara sobre su efectividad en tareas concretas de comprensión del lenguaje.

La preparación de los datos es, por lo tanto, un proceso meticuloso que establece las bases para un entrenamiento exitoso del modelo. Asegurarse de que los datos estén bien limpios, organizados y estructurados es fundamental para maximizar la efectividad del modelo y garantizar que pueda enfrentar con éxito los desafíos planteados en las tareas de procesamiento de lenguaje natural. Cada uno de estos pasos contribuye a la creación de un conjunto de datos robusto y fiable que será utilizado en las fases de modelado y análisis, siendo una etapa que no debe ser subestimada en el proceso general del proyecto.

4.4 Modelado

En esta etapa crítica del proyecto, se lleva a cabo la selección y configuración de los modelos de Machine Learning (ML) que serán utilizados para el entrenamiento del sistema. La elección de los modelos es un paso fundamental, ya que determina la capacidad del sistema para cumplir con los objetivos planteados en fases anteriores. Concretamente, se utilizan dos tipos de entrenamiento, cada uno diseñado para abordar aspectos específicos del procesamiento del lenguaje natural y mejorar el rendimiento del modelo.

- **Entrenamiento Auto-regresivo**

El entrenamiento autoregresivo en el contexto de fine-tuning se utiliza para integrar nueva información en un modelo de lenguaje previamente entrenado. Este proceso se basa en ajustar los parámetros del modelo mediante la exposición a un corpus actualizado o especializado, lo que le permite aprender patrones y detalles específicos relacionados con la nueva información.

Durante el entrenamiento, el modelo predice iterativamente la siguiente palabra en una secuencia utilizando como referencia las palabras precedentes. Este enfoque no solo refuerza la coherencia y la fluidez en la generación de texto, sino que también facilita que el modelo adquiera un entendimiento contextual más preciso del contenido actualizado. De esta manera, el modelo puede adaptarse a contextos cambiantes o a nuevas áreas de conocimiento sin perder las capacidades generales adquiridas en etapas previas de entrenamiento.

Además, se aplica la técnica de LoRA (Low-Rank Adaptation) para mejorar la eficiencia del entrenamiento del modelo. Esta técnica permite ajustar solo una parte de los parámetros del modelo, lo que no solo reduce la carga computacional, sino que también acorta el tiempo necesario para completar el proceso de entrenamiento. La implementación de LoRA es especialmente beneficiosa en entornos donde los recursos son limitados, permitiendo que el modelo se adapte a nuevas tareas sin la necesidad de reentrenar completamente el modelo original.

- **Entrenamiento Tipo Instruct**

El segundo tipo de entrenamiento que se implementa es el entrenamiento tipo instruct. Este enfoque se centra en ajustar el modelo para que genere respuestas concretas a preguntas específicas. Para llevar a cabo este proceso, se utilizó un conjunto de preguntas y respuestas que fue generado anteriormente. Este conjunto se convierte en la base sobre la cual el modelo aprende a formular respuestas adecuadas y pertinentes a las consultas planteadas.

El entrenamiento tipo instruct es crucial para garantizar que el modelo no solo genere texto de forma general, sino que también sea capaz de responder a interrogantes de manera precisa y directa. Este tipo de ajuste permite al modelo adaptarse a tareas de comprensión y generación de lenguaje más complejas, donde se requiere una mayor especificidad y relevancia en las respuestas. La implementación de este entrenamiento asegura que el modelo sea eficaz en contextos donde se necesiten respuestas claras y útiles, lo que es esencial para aplicaciones en campos como la atención al cliente, la educación o cualquier otro dominio donde la interacción humana con máquinas sea necesaria. En este entrenamiento también se aplica LoRA.

- **Aprendizaje por Refuerzo desde Retroalimentación Humana (RLHF)**

Una vez completados los entrenamientos auto-regresivo y tipo instruct, se lleva a cabo un entrenamiento adicional conocido como Aprendizaje por Refuerzo desde Retroalimentación Humana (RLHF). Este método es fundamental para afinar el comportamiento del modelo, ya que permite optimizar la calidad de las respuestas generadas. Durante esta fase, se recopila retroalimentación de los usuarios o evaluadores sobre las respuestas proporcionadas por el modelo, lo que se utiliza para ajustar y mejorar su rendimiento.

El uso de RLHF es particularmente valioso, ya que la retroalimentación humana puede guiar el proceso de aprendizaje de manera que el modelo no solo cumpla con los requisitos técnicos, sino que también se alinee con las expectativas y preferencias de los usuarios. Esto permite una mejora continua en la calidad de las respuestas generadas, haciendo que el modelo sea más útil y relevante en situaciones prácticas. La implementación de este tipo de aprendizaje refuerza la capacidad del modelo para manejar variaciones en las preguntas y adaptarse a diferentes estilos de interacción, contribuyendo así a un sistema más robusto y eficiente.

La combinación de todas estas estrategias de modelado entrenamiento auto-regresivo, entrenamiento tipo instruct, aprendizaje por refuerzo con retroalimentación humana, y la aplicación de técnicas de eficiencia como LoRA proporciona una base sólida para desarrollar un modelo de lenguaje preentrenado que sea eficaz, adaptable y capaz de enfrentar una variedad de tareas en el campo del procesamiento de lenguaje natural. Este enfoque multifacético no solo mejora el rendimiento del modelo, sino que también facilita su implementación en aplicaciones del mundo real, donde la interacción con el usuario y la generación de respuestas precisas son de suma importancia.

4.5 Evaluación

Tras cada etapa de fine-tunig del modelo, se realizaron rigurosas pruebas de rendimiento con el objetivo de evaluar la mejora del modelo frente a tareas predefinidas. Estas pruebas están diseñadas para medir cómo el modelo se adapta a nuevas tareas utilizando conjuntos de datos generados en fases previas del desarrollo. Dichas pruebas son esenciales para comprender la efectividad del modelo y su capacidad para generalizar en situaciones del mundo real, donde se espera que maneje datos nuevos y realice inferencias precisas a partir de ellos.

- **Proceso de evaluación del modelo**

Para realizar esta evaluación, se implementó un software especializado que realiza pruebas detalladas sobre el modelo en un entorno controlado. Este software emplea un conjunto de preguntas y respuestas seleccionadas específicamente para simular una variedad de escenarios que el modelo podría enfrentar en su uso en un entorno real. Las respuestas generadas por el modelo se comparan exhaustivamente con las respuestas correctas previamente definidas, utilizando la API de OpenAI, lo que permite medir de manera precisa la capacidad de respuesta del modelo.

El análisis de similitud entre las respuestas generadas y las respuestas correctas se realiza utilizando el modelo GPT-4o de OpenAI. Este modelo se ha entrenado para comprender el lenguaje de manera avanzada, lo que lo convierte en una herramienta ideal para realizar comparaciones detalladas entre respuestas. La API de OpenAI se utiliza para calcular un porcentaje de similitud entre la respuesta generada por el modelo y la respuesta correcta proporcionada. Este proceso es crítico, ya que permite una evaluación objetiva y cuantitativa del rendimiento del modelo, lo que facilita su análisis y mejora continua.

- **Umbral de similitud y su impacto en la evaluación**

El proceso de comparación entre las respuestas generadas y las respuestas correctas se basa en un umbral de similitud fijado en un 70%. Este umbral es muy exigente, ya que solo las respuestas cuya similitud con las respuestas correctas supere este porcentaje se consideran correctas. Este enfoque riguroso asegura que las respuestas aceptadas como correctas sean altamente precisas, lo cual es crucial cuando se requiere un alto nivel de fiabilidad en las respuestas del modelo.

Es importante destacar que la elección del umbral de similitud tiene un impacto directo en los resultados de la evaluación. Si el umbral se ajustara a un valor más bajo, como el 50%, se observaría un aumento en el porcentaje de respuestas correctas, ya que muchas respuestas que actualmente no alcanzan el 70% de similitud serían aceptadas como correctas. Sin embargo, este ajuste reduciría la exigencia en cuanto a la calidad de las respuestas, lo que podría llevar a que respuestas menos precisas o completamente incorrectas fueran clasificadas como correctas. Por lo tanto, mantener un umbral alto permite mantener un estándar elevado en la evaluación, pero también exige que las respuestas generadas sean más exactas.

- **Uso de GPT-4o**

El modelo GPT-4o de OpenAI es utilizado en este proceso de comparación para evaluar la similitud entre las respuestas generadas y las respuestas correctas. Para esto, se emplea un rol inicial y un prompt exhaustivo que estructura la comparación de manera clara y precisa. El rol refleja el comportamiento predefinido del modelo, mientras que el prompt está diseñado para guiar al modelo de OpenAI a generar un porcentaje numérico que refleje la calidad de la respuesta generada en relación con la correcta. A continuación se detalla el rol definido:

"Eres un asistente que devuelve un porcentaje de acierto dada una pregunta, una respuesta correcta y la respuesta a evaluar."

Y el prompt empleado:

Dada la siguiente pregunta y la "respuesta 1", ¿sería una respuesta correcta a la pregunta la "respuesta 2"? Devuelve únicamente un porcentaje de acierto, no hagas comentarios.

Pregunta: {question}
Respuesta 1: {sentence1}
Respuesta 2: {sentence2}

Este prompt se presenta al modelo de OpenAI de manera estructurada, lo que permite que el modelo evalúe las respuestas de forma precisa. El modelo es instruido para devolver únicamente un porcentaje de similitud entre 0% y 100%, sin hacer comentarios adicionales. Esto facilita una comparación clara y directa de las respuestas, garantizando que la evaluación sea objetiva y basada
únicamente en la similitud cuantitativa entre las respuestas generadas y las respuestas correctas.

- **Resultados y análisis de desempeño**

Una vez realizadas las pruebas, los resultados obtenidos se recopilan en informes detallados que proporcionan tanto un análisis cuantitativo como cualitativo del rendimiento del modelo. Los informes incluyen métricas clave, como el porcentaje de respuestas correctas, el número total de respuestas correctas e incorrectas, y el porcentaje de similitud para cada respuesta generada.

Además, los informes detallados contienen información sobre las preguntas formuladas, las respuestas esperadas, las respuestas generadas por el modelo y los porcentajes de similitud obtenidos para cada caso. Este análisis permite identificar con precisión las áreas donde el modelo ha tenido un buen desempeño, así como aquellas en las que ha cometido errores o donde la precisión de sus respuestas podría mejorarse. La información detallada también permite realizar ajustes finos en el modelo, como la revisión de los datos de entrenamiento o la modificación de parámetros específicos para mejorar la calidad de las respuestas.

El software también genera un informe resumido que proporciona un porcentaje general de respuestas correctas, lo que facilita una evaluación rápida del rendimiento del modelo en su conjunto. Estos informes se almacenan y pueden ser utilizados para realizar un seguimiento del progreso del modelo a lo largo de diferentes etapas de entrenamiento y ajuste.

- **La importancia de la evaluación en el proyecto**

Este proceso de evaluación es un componente esencial en el desarrollo y refinamiento del modelo. La evaluación exhaustiva no solo permite medir el rendimiento del modelo, sino que también ofrece información valiosa sobre cómo este se adapta a diferentes tipos de tareas. Gracias al análisis de los resultados, es posible identificar áreas específicas para la mejora continua del modelo, lo que garantiza que el sistema esté preparado para enfrentar los desafíos en un entorno de procesamiento de lenguaje natural cada vez más complejo.

La implementación de un proceso de evaluación riguroso y detallado, basado en un umbral de similitud estricto y el uso del modelo GPT-4o para la comparación de respuestas, constituye una base sólida para el desarrollo de modelos de lenguaje más precisos y eficaces. Esta metodología es clave para asegurar que el modelo no solo cumpla con los objetivos de rendimiento establecidos, sino que también esté preparado para enfrentar tareas y situaciones específicas en el mundo real, donde la precisión y la fiabilidad son cruciales.

Este enfoque de evaluación exhaustivo y detallado sienta las bases para futuras mejoras y adaptaciones del modelo, asegurando que se mantenga relevante y eficaz en un entorno en constante evolución. La combinación de métricas cuantitativas con un análisis cualitativo profundo garantiza que el modelo continúe mejorando, ofreciendo siempre respuestas más precisas y útiles para una variedad de aplicaciones en el procesamiento de lenguaje natural.

4.6 Planificación del Proyecto

La planificación del proyecto se ha estructurado en diversas fases con el fin de asegurar el cumplimiento de los objetivos establecidos desde el inicio. Esta división del trabajo no solo permite un enfoque ordenado y sistemático, sino que también facilita la asignación de recursos y el seguimiento del progreso en cada etapa del desarrollo. A continuación, se describen las fases programadas para el proyecto:

Mes 1: Comprensión del Negocio y de los Datos

- **Determinación de los Objetivos del Proyecto**: Durante este primer mes, se lleva a cabo una reunión inicial con las partes interesadas para definir claramente los objetivos del proyecto. Esto incluye identificar las metas específicas que se desean alcanzar, así como las expectativas sobre el rendimiento del modelo.

- **Estructuración y Análisis Exploratorio Inicial de los Datos**: Se realiza un análisis exploratorio de los datos disponibles para comprender su estructura y calidad. Esto implica identificar las fuentes de datos, la relevancia de la información y cualquier posible desafío en la obtención y manipulación de los datos.

Mes 2: Preparación de los Datos

- **Implementación del Software de Scraping**: Se desarrolla y despliega un software de scraping automatizado para recolectar datos relevantes desde diversas fuentes en la web. Esta herramienta es crucial para obtener un conjunto de datos robusto que se utilizará posteriormente en el entrenamiento del modelo.

- **Limpieza y Procesamiento de los Datos Obtenidos**: Una vez recolectados, los datos se someten a un proceso de limpieza y procesamiento. Esto incluye la eliminación de información irrelevante, la corrección de errores y la estructuración de los datos en un formato adecuado para su análisis y uso en el modelo.

Mes 3: Modelado

- **Entrenamiento Auto-regresivo y Tipo Instruct**: Se inicia el entrenamiento del modelo utilizando el corpus generado en la fase anterior. Durante esta etapa, se aplican técnicas de entrenamiento auto-regresivo y de tipo instruct para adaptar el modelo a las tareas específicas que se desean ejecutar.

- **Entrenamiento mediante Aprendizaje por Refuerzo**: Posteriormente, se implementa un entrenamiento adicional utilizando aprendizaje por refuerzo, lo que permitirá afinar las respuestas del modelo y mejorar su capacidad de no inventar información al enfrentar preguntas sobre las que carece de datos.

Mes 4: Evaluación

- **Evaluación del Modelo**: Una vez completado el entrenamiento, se realizan pruebas exhaustivas para evaluar el rendimiento del modelo. Se analizan las métricas obtenidas y se comparan con los objetivos iniciales establecidos en la primera fase.

- **Optimización del Modelo**: Basado en los resultados de la evaluación, se llevan a cabo ajustes y mejoras en el modelo. Esto puede incluir reentrenamiento, ajuste de parámetros y la implementación de técnicas adicionales para maximizar la eficacia y precisión del modelo.

Mediante esta planificación estructurada, se garantiza que cada fase de la metodología elegida (CRISP-DM) se aplique correctamente. Esto no solo facilita el seguimiento del progreso, sino que también permite una identificación temprana de posibles obstáculos y desafíos, asegurando que el proyecto se mantenga en el camino correcto hacia la consecución de sus objetivos. La planificación cuidadosa y la ejecución de cada fase son fundamentales para el éxito global del proyecto, asegurando que se logren resultados significativos y de alta calidad en el ámbito del procesamiento del lenguaje natural.

5. ENTORNO DE TRABAJO

El desarrollo de este proyecto se ha realizado en un equipo proporcionado por la ULPGC, que cuenta con un conjunto de especificaciones técnicas avanzadas, además de medidas de seguridad. Se requiere de una conexión segura para el acceso al equipo, lo que permite la protección de datos y procesos ejecutados en el entorno. A continuación, se explican estos puntos de forma más detallada.

5.1 Configuración del Entorno

Este entorno de trabajo destaca no solo por las características técnicas avanzadas del equipo utilizado, sino también por su optimización para tareas que requieren un uso intensivo de recursos computacionales. Las especificaciones técnicas detalladas son las siguientes:

- **Sistema Operativo**
    - Versión: Microsoft Windows 11 Pro
    - Versión del sistema operativo: 10.0.22631

Este sistema operativo es altamente adecuado para un entorno empresarial, ya que integra características de seguridad avanzadas que protegen tanto al sistema como a los datos sensibles. Windows 11 Pro ofrece funcionalidades como cifrado de datos mediante BitLocker, acceso remoto seguro y soporte para aplicaciones empresariales, lo que lo convierte en una elección ideal para proyectos que implican la manipulación de información crítica.

- **Procesador (CPU)**
    - Modelo: Intel Core I7-13700K
    - Frecuencia: 3.4 GHz

Este procesador, que forma parte de la arquitectura x64, está diseñado para ejecutar de manera eficiente procesos complejos y permite la paralelización de tareas. Su capacidad de manejar múltiples hilos a la vez lo hace idóneo para llevar a cabo tareas de alta demanda computacional, como el procesamiento de grandes volúmenes de datos o la ejecución de algoritmos de machine learning que requieren un considerable poder de cálculo.

- **Memoria RAM**
    - Capacidad: 64 GB

La capacidad de 64 GB de memoria RAM es crucial para gestionar grandes conjuntos de datos y realizar operaciones intensivas con ellos. Esta cantidad de memoria permite que el sistema ejecute múltiples aplicaciones simultáneamente sin experimentar caídas en el rendimiento, lo que es particularmente importante durante el entrenamiento de modelos de aprendizaje automático, donde se requiere un acceso rápido y eficiente a la memoria.

- **Unidad de Procesamiento Gráfico (GPU)**
    - Modelo: Nvidia RTX 4090
    - Memoria: 24 GB

La Nvidia RTX 4090 es reconocida por su capacidad y potencia, lo que la hace ideal para tareas que requieren cálculos intensivos, como el entrenamiento de modelos de machine learning y deep learning. Su arquitectura optimizada para el procesamiento paralelo y su amplia memoria permiten manejar modelos complejos y grandes conjuntos de datos de manera eficiente. Esto es especialmente ventajoso en el ámbito del procesamiento del lenguaje natural, donde los modelos pueden ser extremadamente grandes y requieren de recursos significativos para su entrenamiento y evaluación.

Este entorno de trabajo, por lo tanto, está diseñado para proporcionar un rendimiento robusto y seguro, asegurando que todas las fases del proyecto se ejecuten de manera fluida y efectiva. La combinación de un sistema operativo seguro, un potente procesador, una amplia memoria RAM y una GPU de última generación crea una infraestructura capaz de abordar los desafíos que presenta el desarrollo de modelos avanzados en el campo de la inteligencia artificial y el procesamiento del lenguaje natural. La inversión en estas tecnologías no solo facilita el trabajo en el proyecto actual, sino que también establece una base sólida para futuras investigaciones y desarrollos en esta área.

5.2 Acceso al Entorno

Debido a la necesidad de mantener altos estándares de acceso controlado y seguridad, el acceso al equipo se divide en dos pasos fundamentales que garantizan la protección de la información y la integridad del sistema.

- **OpenVPN**

OpenVPN es una herramienta de código abierto que permite establecer una conexión cifrada entre un equipo local y la red de la Universidad de Las Palmas de Gran Canaria (ULPGC). Este protocolo de red virtual privada (VPN) proporciona un túnel seguro que protege los datos que se transfieren, evitando accesos no autorizados y mitigando el riesgo de ataques cibernéticos. Al implementar OpenVPN, se asegura que las comunicaciones sean privadas y confidenciales, lo cual es crítico en el contexto de investigación y desarrollo donde se manejan datos sensibles y recursos académicos.

OpenVPN utiliza tecnologías de cifrado avanzadas, lo que garantiza que toda la información transmitida sea ilegible para posibles interceptores. Esto no solo protege la confidencialidad de los datos, sino que también proporciona autenticación y verificación, asegurando que solo los usuarios autorizados puedan acceder a los recursos de la red. Además, la posibilidad de establecer conexiones a través de diferentes plataformas y dispositivos hace que OpenVPN sea una opción versátil y accesible para el personal y los estudiantes de la ULPGC.

- **Escritorio Remoto de Windows**

El Escritorio Remoto de Windows es una funcionalidad que permite el acceso remoto al entorno de trabajo desde un equipo remoto autorizado, brindando la flexibilidad de utilizar los recursos del sistema desde cualquier ubicación. Esta herramienta es especialmente útil para investigadores y desarrolladores que necesitan trabajar en proyectos de manera remota, permitiendo un acceso fácil y rápido a las aplicaciones y datos almacenados en el equipo de la ULPGC.

Mediante el Escritorio Remoto, los usuarios pueden interactuar con el sistema como si estuvieran físicamente presentes en el entorno de trabajo. Esto incluye la capacidad de ejecutar programas, acceder a archivos y realizar configuraciones necesarias para el desarrollo del proyecto. La interfaz gráfica intuitiva facilita la navegación y el uso de herramientas de software, lo que permite mantener la productividad incluso cuando se trabaja de forma remota.

Adicionalmente, el Escritorio Remoto de Windows también incluye características de seguridad, como la autenticación de usuario y el cifrado de la conexión, que ayudan a proteger el acceso y garantizar que solo los usuarios autorizados puedan iniciar sesión en el sistema. Esto es particularmente importante en entornos académicos donde se manejan datos sensibles y se llevan a cabo investigaciones críticas.

La combinación de OpenVPN y el Escritorio Remoto de Windows proporciona un acceso seguro y eficiente al entorno de trabajo, asegurando que los datos y recursos estén protegidos mientras se mantiene la flexibilidad y la accesibilidad necesarias para el desarrollo del proyecto. Estos métodos de acceso controlado son esenciales para la protección de la información y para la continuidad del trabajo en un entorno de investigación cada vez más dinámico y remoto.

5.3 Relevancia del Entorno para el Proyecto

Utilizar un entorno de alto rendimiento, combinado con un acceso remoto seguro, es fundamental para el desarrollo exitoso de este proyecto. Las configuraciones y especificaciones del equipo empleado están diseñadas para abordar los diversos desafíos técnicos y de procesamiento que se presentan en proyectos de esta naturaleza, especialmente aquellos relacionados con el aprendizaje automático y el procesamiento del lenguaje natural. A continuación, se detallan las principales razones por las cuales este entorno es crucial para el proyecto:

- **Capacidad de Cómputo**

La combinación de hardware de alto rendimiento, que incluye un potente procesador, una capacidad de memoria RAM generosa y una GPU de última generación, permite ejecutar entrenamientos de modelos de machine learning de gran tamaño de manera eficiente y rápida. Esta capacidad de cómputo es esencial durante el desarrollo del proyecto, ya que el entrenamiento de modelos de lenguaje puede ser intensivo en recursos. La velocidad de procesamiento facilita la experimentación y la iteración, lo que permite realizar ajustes y optimizaciones en tiempos más cortos. Esto, a su vez, contribuye a una mayor productividad, ya que los desarrolladores pueden concentrarse en el diseño y la implementación de nuevas estrategias sin verse limitados por las restricciones de hardware.

- **Optimización en NLP**

El entorno de trabajo también proporciona un marco ideal para ejecutar modelos de NLP (Procesamiento del Lenguaje Natural) en tareas como el análisis y la generación de texto. Gracias a la potencia del equipo, se pueden llevar a cabo diversos experimentos simultáneamente, como el ajuste de hiperparámetros, la validación cruzada y la evaluación de diferentes arquitecturas de modelos. Esta capacidad de realizar múltiples experimentos en paralelo no solo acelera el proceso de desarrollo, sino que también permite explorar una variedad de enfoques y metodologías, lo que puede resultar en mejores resultados y un rendimiento más robusto del modelo.

Además, la configuración técnica del entorno es especialmente relevante para manejar los grandes volúmenes de datos asociados con los proyectos de NLP. La capacidad de procesar y analizar estos datos de manera eficiente es crucial para el éxito del proyecto, ya que la calidad y la cantidad de datos utilizados durante el entrenamiento tienen un impacto directo en la efectividad del modelo final.

- **Continuidad y Seguridad del Proyecto**

La infraestructura de acceso remoto que se ha establecido permite que el equipo esté disponible para la supervisión y actualización del proyecto desde ubicaciones fuera de la red de la ULPGC. Esto resulta ser un factor clave en términos de continuidad y flexibilidad del proyecto, ya que permite a los investigadores y desarrolladores trabajar de manera colaborativa y acceder a los recursos necesarios sin importar su ubicación física. La capacidad de realizar tareas de mantenimiento, ajustes y supervisión del modelo de manera remota asegura que el proyecto avance de manera constante, incluso si los miembros del equipo no están presentes en el lugar.

Además, este acceso remoto se complementa con las medidas de seguridad implementadas, que garantizan que los datos y recursos del proyecto estén protegidos contra accesos no autorizados. Esto es particularmente importante en el contexto de la investigación, donde la protección de los datos y la integridad del trabajo realizado son primordiales. En resumen, el entorno de trabajo no solo facilita el desarrollo técnico del proyecto, sino que también asegura la continuidad y la seguridad, elementos que son esenciales para la investigación y el desarrollo en el ámbito del aprendizaje automático y el procesamiento del lenguaje natural.

6. TECNOLOGÍAS Y HERRAMIENTAS EMPLEADAS

Para llevar a cabo este proyecto, se ha empleado un conjunto de herramientas y tecnologías que permiten realizar de manera eficiente las tareas de extracción, procesamiento y modelado de datos, así como el entrenamiento y la validación de los modelos de lenguaje de gran tamaño (LLM). Estas tecnologías son fundamentales para asegurar que el proceso de desarrollo sea fluido, eficaz y se ajuste a los requerimientos específicos del proyecto. A continuación, se detallan las principales tecnologías empleadas en este proceso.

6.1 Lenguaje de Programación

Python ha sido el lenguaje de programación seleccionado para este proyecto debido a su facilidad de uso y flexibilidad, lo cual permite un desarrollo ágil y rápido de prototipos. Su sintaxis clara y baja complejidad lo han posicionado como el estándar de facto en proyectos de ciencia de datos y análisis, así como en el ámbito de la inteligencia artificial. La elección de Python se justifica por varias razones clave:

- **Adopción en la Comunidad Científica**

Python cuenta con una adopción significativa en la comunidad científica y en el ámbito de la inteligencia artificial, lo que facilita el acceso a conocimientos y recursos compartidos. Esta comunidad activa no solo ofrece una rica base de documentación y tutoriales, sino que también mantiene un ecosistema vibrante de foros y grupos de discusión, lo que es invaluable para resolver problemas y obtener nuevas ideas.

- **Bibliotecas y Frameworks Específicos**

Python ofrece una amplia variedad de bibliotecas y frameworks especializados que optimizan el procesamiento y análisis de datos. Entre ellos se destacan bibliotecas como Pandas, que proporciona estructuras de datos eficientes y herramientas de análisis; NumPy, que es fundamental para cálculos numéricos; y SciPy, que se utiliza para resolver problemas científicos y técnicos. Además, en el contexto del aprendizaje automático y el procesamiento del lenguaje natural, bibliotecas como TensorFlow, Keras y PyTorch son esenciales para el desarrollo y entrenamiento de modelos avanzados.

- **Facilidad de Integración**

La capacidad de Python para integrarse fácilmente con otras tecnologías y lenguajes también es un punto a favor. Esto permite que las soluciones desarrolladas sean más flexibles y adaptables, pudiendo incorporarse en diferentes entornos de producción o de investigación sin mayores complicaciones.

- **Recursos de Aprendizaje y Desarrollo**

La abundancia de recursos de aprendizaje disponibles, como cursos en línea, libros y tutoriales, hace que Python sea accesible para desarrolladores de todos los niveles, desde principiantes hasta expertos. Esto es especialmente valioso en un entorno académico o de investigación, donde los equipos pueden estar compuestos por individuos con diferentes niveles de experiencia.

- **Comunidad de Soporte**

La comunidad de Python es conocida por su disposición a ayudar y compartir conocimientos. Esto significa que, en caso de encontrar problemas o desafíos técnicos, es probable que ya exista una solución o una discusión relevante en línea, lo que acelera la resolución de problemas.

Python se ha consolidado como la elección preferida para este proyecto debido a sus numerosas ventajas en términos de facilidad de uso, flexibilidad, y el vasto ecosistema de bibliotecas y recursos que ofrece. Estas características son fundamentales para el desarrollo eficaz y eficiente de soluciones en el ámbito de la inteligencia artificial y el procesamiento del lenguaje natural. La combinación de estas capacidades hace que Python no solo sea un lenguaje potente, sino también una herramienta indispensable para llevar a cabo los objetivos planteados en este proyecto.

6.2 Entorno Virtual

Para asegurar un desarrollo eficiente y ordenado, se emplea un entorno virtual en el proyecto. Este recurso es fundamental en el ámbito de la programación en Python, ya que permite crear una instancia aislada del intérprete de Python donde se pueden gestionar dependencias y bibliotecas de manera independiente del sistema operativo. Esta práctica es especialmente útil en proyectos que pueden requerir diferentes versiones de bibliotecas o paquetes, así como para evitar conflictos entre dependencias. A continuación, se describen los beneficios y la implementación del entorno virtual en este proyecto:

- **Aislamiento de Dependencias**

El uso de un entorno virtual permite que todas las bibliotecas y dependencias específicas del proyecto se instalen en un espacio aislado, sin interferir con otras configuraciones del sistema operativo o de otros proyectos. Esto es crucial en entornos de desarrollo, donde diferentes proyectos pueden necesitar diferentes versiones de la misma biblioteca.

- **Compatibilidad y Estabilidad**

Al crear un entorno virtual, se asegura que el proyecto utiliza versiones específicas de dependencias y bibliotecas que han sido probadas y validadas para funcionar correctamente. Esto evita que actualizaciones o cambios en el sistema operativo afecten el funcionamiento del proyecto, manteniendo su estabilidad a lo largo del tiempo.

- **Facilidad de Replicación**

Utilizar entornos virtuales facilita la replicación del entorno de trabajo en otros sistemas. Esto es especialmente útil en entornos colaborativos, donde varios desarrolladores pueden trabajar en el mismo proyecto. Con un entorno virtual, es posible compartir un archivo de configuración que detalla las bibliotecas y versiones necesarias, permitiendo que otros desarrolladores reproduzcan exactamente el mismo entorno con un simple comando.

- **Prevención de Conflictos de Versiones**

Los entornos virtuales son una solución eficaz para evitar conflictos de versiones que pueden surgir al trabajar con múltiples proyectos que requieren diferentes versiones de bibliotecas. Esto reduce significativamente el tiempo y el esfuerzo necesarios para resolver problemas de compatibilidad.

- **Miniconda**

Esta herramienta es una distribución mínima de Conda, diseñada para crear entornos virtuales ligeros y fáciles de gestionar. En este proyecto, se ha configurado un entorno virtual utilizando Miniconda para instalar las dependencias y bibliotecas necesarias. Las características de Miniconda que lo hacen particularmente útil incluyen:

  - **Ligero y Eficiente**: A diferencia de Anaconda, que incluye un gran número de bibliotecas preinstaladas, Miniconda proporciona un enfoque más ligero, permitiendo a los desarrolladores instalar solo lo que necesitan. Esto ahorra espacio en disco y reduce el tiempo de configuración inicial.

  - **Gestión de Paquetes**: Miniconda incluye la herramienta conda, que permite no solo la creación y gestión de entornos virtuales, sino también la instalación y actualización de paquetes de manera sencilla. Esto facilita la gestión de dependencias y asegura que todas las bibliotecas necesarias estén disponibles.

  - **Reproducibilidad**: Miniconda permite crear archivos de especificaciones que documentan las dependencias y versiones instaladas en el entorno virtual. Esto asegura que otros desarrolladores puedan replicar exactamente el mismo entorno en sus propias máquinas, lo cual es crucial para la colaboración y la resolución de problemas.

  - **Integración con Python**: Dado que Miniconda se basa en Python, se integra perfectamente con el lenguaje y sus bibliotecas, facilitando la instalación de paquetes y su uso en proyectos de ciencia de datos y aprendizaje automático.

El uso de un entorno virtual mediante Miniconda no solo proporciona un espacio de trabajo aislado y controlado, sino que también garantiza que el proyecto mantenga su integridad y funcionalidad a lo largo del tiempo. Esta práctica es una parte esencial del flujo de trabajo en el desarrollo de software moderno, especialmente en proyectos que implican el uso de múltiples bibliotecas y tecnologías.

6.3 Entornos de Desarrollo

Los entornos de desarrollo son fundamentales para llevar a cabo proyectos de análisis de datos, machine learning y, en general, cualquier tipo de desarrollo de software. Proporcionan herramientas que facilitan la escritura, depuración y visualización de código, además de permitir una gestión más eficaz de los proyectos. En este proyecto, se han utilizado varios entornos de desarrollo que ofrecen diversas funcionalidades, permitiendo un enfoque integral y eficiente en el desarrollo. A continuación, se describen los principales entornos de desarrollo utilizados:

- **Jupyter Notebook**

Jupyter Notebook es una herramienta de código abierto ampliamente utilizada en el ámbito del análisis de datos y machine learning. Su principal atractivo radica en su capacidad para combinar código, visualización y documentación en un solo archivo, facilitando así la interacción durante el análisis y la experimentación.

  - **Facilitación del Proceso de Experimentación**: En el contexto de este proyecto, Jupyter Notebook se ha utilizado para llevar a cabo experimentos de manera interactiva. Permite la ejecución secuencial de código, lo que significa que los desarrolladores pueden ejecutar celdas individuales de código y observar los resultados en tiempo real. Esta característica es especialmente útil para explorar datos, ya que los usuarios pueden realizar análisis preliminares, ajustar parámetros y observar cómo esos cambios afectan a los resultados inmediatamente.

  - **Visualización de Resultados**: Además de ejecutar código, Jupyter Notebook permite la integración de gráficos y visualizaciones directamente en el flujo de trabajo. Esto es crucial para la interpretación de datos y el entrenamiento de modelos, ya que proporciona una comprensión visual de cómo se están comportando los datos y los modelos a medida que se llevan a cabo diferentes experimentos.

  - **Documentación y Presentación**: Jupyter también permite añadir texto formateado y notas, lo que convierte a los notebooks en una excelente herramienta para documentar el proceso de análisis y los resultados obtenidos. Esto es valioso tanto para el seguimiento personal como para la presentación de resultados a otros miembros del equipo o partes interesadas.

- **Visual Studio Code (VS Code)**

Visual Studio Code es un Entorno de Desarrollo Integrado (IDE) altamente personalizable, diseñado para ofrecer flexibilidad y adaptabilidad según las necesidades de cada proyecto. Gracias a su compatibilidad con una amplia gama de lenguajes de programación y su capacidad para extenderse mediante extensiones, VS Code se ha convertido en una de las herramientas más populares entre desarrolladores de todo el mundo.

  - Flexibilidad y Personalización: Una de las características más destacadas de VS Code es su extensibilidad. Los usuarios pueden instalar extensiones para añadir nuevas funcionalidades, lo que permite integrar herramientas avanzadas para análisis de datos, depuración y visualización, entre otras. Esto proporciona un entorno de trabajo completo y adaptable a las exigencias específicas del proyecto.

  - ntegración con Jupyter Notebook: En el marco de este proyecto, se ha realizado la integración de Jupyter Notebook directamente en VS Code. Esta característica permite a los desarrolladores beneficiarse de las capacidades interactivas de los notebooks mientras trabajan en el entorno robusto de VS Code. Esto mejora significativamente el flujo de trabajo, ya que permite a los usuarios ejecutar celdas de código, visualizar resultados y realizar modificaciones en un solo lugar.

  - Eficiencia en el Desarrollo de Modelos: Esta integración resulta especialmente beneficiosa en el proceso de entrenamiento de modelos de lenguaje a gran escala (LLM). Los desarrolladores pueden gestionar el código y los notebooks en un solo entorno, lo que aumenta la eficiencia y la organización del trabajo.

- **PyCharm**

PyCharm es un IDE especializado en el lenguaje de programación Python. Este entorno de desarrollo ofrece una serie de funcionalidades avanzadas que mejoran la experiencia de codificación y facilitan la creación de código de alta calidad.

  - **Características Avanzadas**: Entre las funcionalidades destacadas de PyCharm se encuentran la refactorización de código inteligente, herramientas de depuración avanzadas y el autocompletado de código. Estas herramientas son esenciales para mejorar la legibilidad del código y garantizar que se sigan las mejores prácticas de programación.

  - **Optimización para Proyectos de Gran Tamaño**: PyCharm está diseñado para gestionar proyectos de Python de gran envergadura, lo que lo convierte en una opción ideal para la implementación de módulos complejos. Su estructura de proyecto organizada ayuda a mantener la claridad y la coherencia en el desarrollo.

  - **Extensiones**: Otra ventaja de PyCharm es su capacidad para añadir extensiones que proporcionan funcionalidades adicionales que no están disponibles de serie. Esto permite a los desarrolladores personalizar su entorno de trabajo según las necesidades del proyecto.

  - **Uso en Implementación**: En este proyecto, PyCharm se ha utilizado específicamente para implementar los softwares de scraping y las pruebas de los modelos de lenguaje (LLM). Su capacidad para manejar bibliotecas de Python y proporcionar herramientas avanzadas de depuración ha sido crucial para el éxito en estas tareas.

6.4 Bibliotecas para Machine Learning y NLP

Las herramientas y bibliotecas para Machine Learning (ML) y Procesamiento de Lenguaje Natural (NLP) son fundamentales para el desarrollo de modelos eficientes y precisos. A continuación, se detallan las principales bibliotecas empleadas en los procesos de ML y NLP en este proyecto:

- **Transformers**

La biblioteca Transformers, desarrollada por Hugging Face, es esencial para trabajar con modelos preentrenados de NLP. Esta biblioteca ofrece una interfaz optimizada y flexible que facilita el acceso a una amplia colección de modelos de lenguaje de última generación (LLM), incluyendo algunos de los más reconocidos en la comunidad de inteligencia artificial, como BERT, Mistral y GPT, entre otros.

  - **Carga y Uso de Modelos**: Con Transformers, es posible cargar en memoria estos modelos preentrenados para su uso inmediato. Esto permite a los desarrolladores comenzar a trabajar con LLM sin necesidad de entrenarlos desde cero, lo que ahorra tiempo y recursos computacionales.

  - **Preparación para Fine-Tuning**: Además de cargar modelos, la biblioteca también proporciona herramientas para preparar estos modelos preentrenados para el Fine-Tuning posterior. Esto incluye la configuración de arquitecturas de red, la optimización de hiperparámetros y la implementación de técnicas de ajuste específicas según la tarea en cuestión.

  - **Amplia Comunidad y Soporte**: La popularidad de la biblioteca Transformers se debe en gran medida a su comunidad activa, que proporciona recursos, tutoriales y ejemplos prácticos. Esto facilita el aprendizaje y la implementación de modelos complejos, incluso para aquellos que son nuevos en el campo del NLP.

- **PEFT (Parameter-Efficient Fine-Tuning)**

PEFT es una biblioteca diseñada específicamente para realizar ajustes de modelos a tareas concretas de manera eficiente. En lugar de reentrenar todo el modelo, PEFT permite modificar únicamente parámetros específicos, lo que resulta en un ajuste más ágil y menos costoso computacionalmente.

    - **Reducción del Coste Computacional**: Este enfoque de Fine-Tuning eficiente no solo reduce el tiempo necesario para adaptar un LLM a una tarea específica, sino que también minimiza la necesidad de recursos computacionales. Esto es especialmente beneficioso cuando se trabaja con modelos grandes que de otro modo requerirían una considerable potencia de cálculo.

    - **Aplicaciones Prácticas**: PEFT se ha utilizado en diversas aplicaciones en el campo de NLP, donde la personalización y la eficiencia son cruciales. Esta biblioteca facilita la adaptación de modelos a contextos específicos, mejorando su rendimiento sin la carga de entrenamiento completo.

- **TRL (Transformer Reinforcement Learning)**

TRL es otra biblioteca ofrecida por Hugging Face que se centra en la combinación de LLM de la biblioteca Transformers con algoritmos de Aprendizaje por Refuerzo (RL). Este enfoque permite ajustar modelos mediante retroalimentación constante, mejorando su capacidad para interactuar de manera efectiva con su entorno.

  - **Ajuste Posterior al Fine-Tuning**: Después de realizar el Fine-Tuning de un modelo, TRL se emplea como un paso final para optimizar la interacción del modelo con las diversas situaciones que puede enfrentar. Esto se traduce en una mejora significativa en la calidad de las respuestas generadas por el modelo, permitiendo una mayor adaptabilidad a diferentes contextos.

  - **Optimización del Rendimiento**: La integración del aprendizaje por refuerzo a través de TRL convierte a esta biblioteca en una herramienta crucial para aplicaciones donde la precisión y la capacidad de respuesta son vitales. Permite que los modelos se ajusten dinámicamente en función de la retroalimentación recibida, lo que resulta en un rendimiento optimizado en tareas interactivas.

  - **Aplicaciones en el Mundo Real**: TRL se ha utilizado en una variedad de aplicaciones que requieren modelos altamente adaptables, como asistentes virtuales, chatbots y sistemas de recomendación. Su capacidad para aprender de la interacción con los usuarios mejora la experiencia general y la utilidad de los modelos de lenguaje.

6.5 Control de Versiones

El control de versiones es un componente esencial en el desarrollo de software, ya que proporciona un sistema robusto para gestionar las modificaciones realizadas en los archivos de un proyecto. Este sistema se basa en un historial detallado que registra todas las alteraciones efectuadas, lo que permite a los desarrolladores volver a versiones anteriores del código cuando sea necesario, así como gestionar diferentes versiones de manera simultánea. Además, el control de versiones facilita la colaboración de múltiples desarrolladores en un mismo proyecto, asegurando una organización y estructuración adecuadas del trabajo.

- **Git**

Git es una herramienta de control de versiones ampliamente reconocida y utilizada en la industria del software. Esta herramienta permite a los desarrolladores llevar un registro minucioso de todos los cambios realizados en el proyecto, desde la adición de nuevas funcionalidades hasta la corrección de errores. Gracias a su diseño distribuido, cada desarrollador tiene una copia completa del repositorio, lo que mejora la velocidad y la eficiencia en la gestión de cambios.

  - **Registro Detallado**: Git mantiene un historial completo de todas las modificaciones, permitiendo a los desarrolladores identificar y revertir cambios problemáticos fácilmente.

  - **Ramas**: Git permite la creación de ramas (branches), lo que facilita el trabajo en nuevas características o arreglos sin afectar el código principal. Esto permite experimentar y desarrollar de manera segura.

  - **Fusión (Merging)**: Después de trabajar en una rama, los desarrolladores pueden fusionar (merge) sus cambios con el código principal, garantizando que todas las mejoras y correcciones se integren de forma controlada.

En el contexto de este proyecto, Git ha sido fundamental para llevar un seguimiento efectivo del código desarrollado. Las funcionalidades de Git han permitido revertir ciertas acciones en casos específicos, facilitando la corrección de errores y la optimización del flujo de trabajo. Además, la creación de ramas ha permitido realizar experimentos y ajustes en el modelo sin comprometer la estabilidad del código principal.

- **GitHub**

GitHub es una plataforma de colaboración y almacenamiento en la nube que se basa en Git. Esta herramienta no solo centraliza el proyecto, sino que también ofrece un entorno seguro y confiable para el almacenamiento del código, generando copias de seguridad automáticas en la nube. Esto asegura la disponibilidad del código en caso de fallos locales o pérdidas de datos.

    - **Colaboración**: GitHub facilita la colaboración entre varios desarrolladores, permitiendo revisiones de código y comentarios en tiempo real. Los pull requests permiten que los cambios propuestos sean discutidos y revisados antes de ser incorporados al código principal.

    - **Issues**: GitHub proporciona una funcionalidad de seguimiento de problemas (issues), lo que ayuda a gestionar tareas, errores y mejoras en el proyecto de manera organizada.

    - **Documentación**: La posibilidad de incluir documentación directamente en el repositorio (README.md y wikis) permite que cualquier colaborador entienda rápidamente la estructura y el propósito del proyecto.

Durante el desarrollo de este proyecto, GitHub ha sido esencial para el almacenamiento y seguimiento del código. La plataforma ha permitido centralizar el trabajo, asegurando que todas las contribuciones se registren adecuadamente y estén disponibles para todos los colaboradores. Esto ha facilitado la gestión de cambios y la colaboración continua entre los miembros del equipo.

7. IMPLEMENTACIÓN

En este apartado se describen las tecnologías y el LLM empleados en cada fase del proyecto. Cada componente desarrollado en este proyecto se divide en los siguientes puntos:

- **LLM empleado**
- **Software de obtención y generación de datos**
- **Fine-Tuning del modelo combinando Auto-regresión y Entrenamiento tipo instruct**
- **Software de test del modelo**

7.1 LLM Empleados

En este proyecto se han empleado dos modelos preentrenados: **"Qwen/Qwen2-1.5B-Instruct"** (Academy, 2023) y **"unsloth/mistral-7b-instruct-v0.2-bnb-4bit"** (Unsloth, 2023). Ambos modelos fueron seleccionados por sus características y capacidades complementarias, siendo utilizados de forma secuencial para llevar a cabo los experimentos y evaluar sus rendimientos en diversas tareas de procesamiento del lenguaje natural (NLP).

- **Qwen/Qwen2-1.5B-Instruct**

Este modelo cuenta con 1.5 billones (miles de millones) de parámetros y está especializado en la comprensión y generación de texto, lo que lo convierte en una opción poderosa para diversas tareas de NLP. Como su nombre indica, forma parte de la familia Qwen, reconocida por su habilidad para realizar tareas específicas tras ser adaptada mediante técnicas de fine-tuning. Esta capacidad de adaptación es especialmente relevante para el proyecto, ya que permite ajustar el modelo a necesidades particulares sin requerir un entrenamiento exhaustivo desde cero.

La elección de Qwen2-1.5B-Instruct se basó en varios factores, entre los que destacan su equilibrio entre tamaño y eficiencia en el uso de recursos. Dado que el entrenamiento y la implementación de modelos de lenguaje grandes suelen demandar un alto coste computacional y de tiempo, seleccionar un modelo con una arquitectura optimizada como Qwen2 resulta clave para llevar a cabo experimentos en entornos con recursos limitados. Este modelo permite una generación de texto avanzada en una amplia variedad de contextos, lo que lo hace adecuado para aplicaciones en las que la calidad y la relevancia del contenido generado son críticas.

Además, Qwen2-1.5B-Instruct es altamente compatible con técnicas de PEFT (Parameter-Efficient Fine-Tuning), que permiten ajustar el modelo para tareas específicas al modificar solo un conjunto reducido de parámetros, en lugar de reentrenar la totalidad del modelo. Esta característica no solo reduce el tiempo y el coste asociados al ajuste del modelo, sino que también facilita la implementación en escenarios donde los recursos son limitados. La combinación de un modelo potente como Qwen2-1.5B-Instruct y las técnicas de PEFT se traduce en un enfoque eficiente para abordar las diversas tareas del proyecto, desde la extracción de datos hasta la generación de respuestas relevantes, optimizando así todo el proceso de desarrollo y asegurando resultados de alta calidad.

- **unsloth/mistral-7b-instruct-v0.2-bnb-4bit**

El modelo unsloth/mistral-7b-instruct-v0.2-bnb-4bit fue el segundo empleado en este proyecto. Este modelo, con 7 billones (miles de millones) de parámetros, es significativamente mayor que el Qwen2-1.5B-Instruct, lo que le otorga un mayor potencial para capturar patrones complejos y ofrecer respuestas más matizadas. Su arquitectura optimizada en 4 bits permite un uso más eficiente de los recursos computacionales, haciéndolo ideal para realizar experimentos en infraestructuras con limitaciones de hardware.

Al igual que Qwen2-1.5B-Instruct, este modelo es compatible con técnicas de Fine Tuning, pero también destaca por su capacidad para manejar tareas de mayor complejidad gracias a su tamaño ampliado. En los experimentos realizados, unsloth/mistral-7b-instruct-v0.2-bnb-4bit fue utilizado después de probar con Qwen2-1.5B-Instruct, permitiendo una comparación directa entre ambos en términos de rendimiento, calidad de las respuestas y consumo de recursos.

La combinación de ambos modelos ofrece una perspectiva rica sobre las fortalezas y debilidades de cada uno en diferentes contextos. Mientras que Qwen2-1.5B-Instruct se mostró eficiente en escenarios donde los recursos computacionales eran limitados, unsloth/mistral-7b-instruct-v0.2-bnb-4bit sobresalió en tareas más exigentes, aprovechando su mayor capacidad para modelar patrones complejos y generar texto de alta calidad.

7.2 Software de Obtención y Generación de Datos

Para realizar fine-tuning al modelo seleccionado, es necesario obtener un corpus de datos con la información requerida para la nueva tarea, un conjunto de preguntas con sus respectivas respuestas generadas a partir del corpus para entrenamiento de RL, y otro conjunto de preguntas y respuestas para realizar tests de rendimiento del modelo.

Para ello se desarrolló un software de scraping mediante el cual se generó el corpus necesario a partir de información en línea, obteniendo un conjunto de datos representativo de varias temáticas que el modelo inicial desconoce. Este software fue diseñado para obtener tanto el texto plano de los documentos como diferentes metadatos de los mismos.

A continuación, se muestra el método de acción:

- **Extracción de URLs**

“UrlScraper” es el principal módulo, este se encarga de la obtención de las URLs utilizando la librería **“requests”** (Reitz, 2024), que realiza peticiones HTTP al dominio web designado en el fichero de configuración “config.json”. Esta clase emplea el parser HTML **“BeautifulSoup”** (Richardson, 2024) para el análisis del contenido de la web y obtener los enlaces que conducen a información relevante. Dichos enlaces son identificados mediante patrones que indica si el enlace conduce a información relevante que debe ser capturada. Se almacenan todas las URLs de las que se ha extraído información en un fichero de texto denominado **“url.txt”**. Este proceso se repite un número determinado de veces que se indica en el fichero de configuración anteriormente indicado.

- **Extracción de texto**

Tras obtener las URLs, un módulo denominado **“ReadUrl”** descarga todo el contenido relevante de los enlaces, esto lo consigue accediendo a cada enlace del fichero **“url.txt”** generado en el paso anterior. Para este proceso, se utiliza de nuevo **“BeautifulSoup”** (Richardson, 2024) que analiza todo el contenido HTML de cada web y extrae los párrafos relevantes que se identifican mediante patrones también añadidos al fichero de configuración. Todo el texto extraído es almacenado en ficheros de texto individuales en un directorio de salida, el nombre de cada fichero incluye los metadatos extraídos del enlace (dominio, tipo de contenido y fecha).

- **Limpieza de datos**

Tras extraer y almacenar individualmente la información, el módulo **“FileManager”** procesa y limpia los datos. Este proceso empieza eliminando las líneas irrelevantes que contienen menos de un número determinado de palabras mediante la clase **“ContentRemover”**, este número se define en el fichero de configuración. Tras filtrar los textos, se combinan como texto plano en un único fichero llamado **“merged_text.txt”**, obteniendo el corpus de información que se utilizará más adelante para el entrenamiento autoregresivo.

- **Generación de preguntas y respuestas**

El último paso es generar dos conjuntos de preguntas y sus respuestas en base al corpus creado anteriormente. Para ello se utiliza la API de OpenAI (OpenAI, 2024), indicando un LLM para el proceso y el número de preguntas para el conjunto de train y test en el fichero de configuración. Las preguntas se generan en formato JSON (JSON.org, 2024) y se almacenan en dos ficheros tipo JSONL: **“train.jsonl”** y **“test.jsonl”**. Por lo tanto, mediante este proceso se automatiza la creación de dos conjuntos de preguntas basadas en la información previamente recolectadas.

7.3 Fine-Tuning del Modelo Combinando Auto-regresión y Entrenamiento tipo Instruct

Una vez que se han obtenido y estructurado los datos necesarios, se procede al proceso de fine-tuning de los modelos seleccionados. En primer lugar, se realiza el ajuste utilizando el modelo "Qwen/Qwen2-1.5B-Instruct" mediante técnicas PEFT (Parameter-Efficient Fine-Tuning). Posteriormente, se emplea el modelo "unsloth/mistral-7b-instruct-v0.2-bnb-4bit", aplicando el mismo enfoque de ajuste eficiente para comparar el rendimiento entre ambos modelos y evaluar su capacidad para abordar las tareas específicas definidas en el proyecto.

El proceso de Fine-Tuning se organiza en los siguientes pasos:

- **Preparación del Modelo y Tokenización**

Se inicia cargando el modelo y su correspondiente tokenizador utilizando la biblioteca **“Transformers”** de Hugging Face. Esta biblioteca proporciona las herramientas necesarias para manipular modelos de lenguaje preentrenados de forma sencilla y eficiente, facilitando la integración del modelo en el flujo de trabajo de Fine-Tuning. La tokenización es un paso crucial, ya que convierte el texto en un formato comprensible por el modelo, dividiendo el texto en tokens que pueden ser procesados durante el entrenamiento.

- **Preparación de Datasets**

El proceso de Fine-Tuning se realiza utilizando dos conjuntos de datos:

  - **Dataset Auto-regresivo**: este conjunto de datos consiste en el corpus completo, que ha sido previamente almacenado en el fichero **“merged_text.txt”**. Este corpus contiene la información necesaria para que el modelo aprenda a generar texto coherente basado en la secuencia de entrada. El modelo se ajustará para predecir la siguiente palabra en una secuencia dada, lo que es fundamental para mejorar su capacidad de comprensión y generación de texto.

  - **Dataset Tipo Instruct**: este conjunto de datos se deriva del fichero **“train.jsonl”**. Contiene preguntas y respuestas generadas a partir del corpus, y está diseñado para entrenar al modelo en un formato de instrucción. Este tipo de dataset permite que el modelo se entrene en la tarea específica de responder preguntas de manera más efectiva, mejorando su capacidad para seguir instrucciones y producir respuestas adecuadas.

- **Ajuste del Modelo con LoRA**

Para optimizar el proceso de entrenamiento, se implementa la técnica LoRA (Low-Rank Adaptation), que permite ajustar el modelo a tareas específicas utilizando significativamente menos recursos computacionales. Esta técnica se centra en modificar solo ciertos parámetros del modelo, en lugar de reentrenar todos sus componentes. Al hacerlo, se reduce la necesidad de almacenamiento y procesamiento, facilitando un entrenamiento más rápido y eficiente.

Con LoRA, el enfoque se centra en ajustar ciertas capas del modelo mientras se mantienen los pesos originales intactos. Esto no solo optimiza el uso de recursos, sino que también preserva la capacidad del modelo preentrenado para manejar una amplia variedad de tareas, permitiendo un Fine-Tuning más efectivo en las áreas específicas de interés del proyecto.

Al combinar las técnicas de auto-regresión y el entrenamiento tipo instruct, se busca mejorar la versatilidad y el rendimiento del modelo en tareas específicas, asegurando que sea capaz de generar y comprender texto de manera más efectiva. Esta metodología no solo se alinea con los objetivos del proyecto, sino que también establece una base sólida para la siguiente etapa: el entrenamiento del modelo mediante aprendizaje por refuerzo.

7.4 Software de Test del Modelo

El software de test implementado tiene el objetivo de evaluar el rendimiento de un modelo preentrenado o ajustado (Fine-Tuned) mediante un conjunto de preguntas y sus respectivas respuestas generado específicamente para el test.

La evaluación se realiza comparando las respuestas generadas por el modelo a las preguntas del conjunto de prueba con las respuestas correctas esperadas. Para llevar a cabo esta comparación, se emplea la API de OpenAI, configurada específicamente para utilizar el modelo GPT-4o. Este enfoque garantiza un análisis exhaustivo y fiable de la similitud entre las respuestas generadas y las correctas. En particular, el proceso de evaluación utiliza un prompt diseñado con precisión para asegurar una interpretación adecuada de las respuestas. El prompt incluye la pregunta original, la respuesta generada por el modelo, y la respuesta esperada, solicitando explícitamente un porcentaje que indique el grado de similitud entre ambas.

La API devuelve un porcentaje que refleja la concordancia entre las respuestas, basado en la evaluación lingüística y contextual proporcionada por GPT-4o. Este porcentaje se compara luego con un umbral de similitud, configurado en un valor del 70%, considerado como altamente exigente. Esto significa que solo aquellas respuestas con un porcentaje igual o superior a este umbral se consideran correctas. Si este umbral se ajustara a un valor más bajo, como por ejemplo el 50%, el porcentaje de respuestas clasificadas como correctas aumentaría significativamente, aunque a costa de disminuir la exigencia en la precisión.

Posteriormente, se generan dos tipos de informes detallados

- **Informe Resumido**: Proporciona una visión general del rendimiento del modelo, incluyendo el porcentaje de respuestas correctas, así como un conteo de aciertos y errores.

- **Informe Detallado**: Presenta un desglose completo de cada interacción, indicando la pregunta planteada, la respuesta correcta, la respuesta generada por el modelo, y el porcentaje de similitud asignado por la API de OpenAI.

Ambos informes son almacenados automáticamente en el directorio de salida especificado durante la configuración inicial del programa. Este proceso no solo permite evaluar la precisión global del modelo, sino que también facilita un análisis cualitativo profundo de las interacciones. La inclusión de detalles como el porcentaje de similitud en cada entrada resulta crucial para identificar patrones de error y posibles áreas de mejora, haciendo de esta herramienta un componente esencial en la evaluación iterativa y refinamiento de modelos ajustados o preentrenados.

El proceso de test se realiza en los siguientes pasos

- **Carga y Configuración del Software**

El programa comienza cargando la configuración desde un archivo JSON mediante la clase **“ConfigManager”**, que asegura la existencia del directorio de salida donde se almacenarán los resultados. Los parámetros del modelo, la ruta a los datos de prueba y otros valores clave se extraen del archivo de configuración.

- **Preparación de los Datos de Prueba**

El módulo **“QuestionLoader”** se utiliza para cargar las preguntas y respuestas desde un archivo JSONL. Cada entrada incluye:

  - Una pregunta.
  - Una respuesta esperada.
  - Su formato estructurado para facilitar la evaluación.

Los datos son almacenados en un diccionario donde las preguntas son las claves y las respuestas esperadas son los valores.

- **Generación de Respuestas por el Modelo**

La clase **“ModelToTest”** se encarga de preparar el modelo para generar respuestas. Según la configuración:

  - Si no hay un adaptador LoRA, se utiliza el modelo base.
  - Si existe un adaptador LoRA, este se carga, y el modelo se configura para incorporar los ajustes realizados en el proceso de Fine-Tuning.

El método **output** genera la respuesta a cada pregunta mediante un pipeline de generación de texto.

- **Evaluación de Respuestas**

El software compara las respuestas generadas por el modelo con las esperadas utilizando la clase **Similarity**, que emplea un modelo adicional configurado a través de la API de OpenAI. Este modelo determina el porcentaje de similitud entre las respuestas generadas y las correctas, devolviendo un valor numérico que indica el grado de concordancia.

El método **are_sentences_similar** formula las comparaciones basándose en el siguiente formato:

- **Pregunta**: La pregunta planteada.
- **Respuesta 1**: La respuesta generada por el modelo.
- **Respuesta 2**: La respuesta esperada.

El resultado es un porcentaje que indica qué tan precisa es la respuesta generada.

- **Manejo de Resultados**

La clase **ResultHandler** gestiona los resultados obtenidos:

  - **Clasificación de Respuestas**: Cada respuesta es evaluada como correcta o incorrecta en función de un umbral de similitud definido en la configuración.

  - **Reporte Resumido**: Se genera un resumen con el porcentaje de respuestas correctas, junto con un conteo de respuestas acertadas e incorrectas.

  - **Reporte Detallado**: Incluye la pregunta, la respuesta esperada, la respuesta generada y el porcentaje de similitud para cada entrada.

Ambos reportes se guardan en archivos dentro del directorio de salida especificado en la configuración.

8. PRUEBAS Y RESULTADOS

En este apartado se presentan los resultados obtenidos durante las pruebas de rendimiento realizadas sobre los modelos seleccionados, tanto antes como después de las diferentes etapas del fine-tuning. Además, se analizan los recursos consumidos por los modelos y su desempeño en términos de precisión.

8.1 Datos Utilizados

El conjunto de datos empleados fue generado a partir de noticias recientes del periódico **Canarias7**, recolectadas mediante el software de scraping anteriormente descrito. Este proceso permitió extraer un total de **132 noticias**, las cuales fueron sometidas a un proceso de filtrado y preprocesamiento. El resultado de esta preparación de datos dio lugar a los siguientes elementos clave:

- **Corpus de texto plano**

Un archivo de texto que recopila el contenido procesado de las **132 noticias**, proporcionando una base textual homogénea para el entrenamiento autoregresivo.

- **Preguntas y respuestas para entrenamiento**

Un conjunto de **973 preguntas** junto con sus respectivas respuestas correctas, almacenado en el archivo “train.jsonl”. Este conjunto fue empleado en la etapa de fine-tuning para adaptar los modelos a la tarea específica.

- **Preguntas y respuestas para evaluación**

Un segundo conjunto independiente de **222 preguntas** con sus respuestas correspondientes, contenido en el archivo “test.json”. Este conjunto se utilizó exclusivamente en la etapa de prueba para evaluar el rendimiento de los modelos en las distintas etapas de fine-tuning.

Cabe destacar que, aunque la mayoría de las noticias tienen un carácter local, **algunas poseen relevancia internacional**, lo que explica el **porcentaje de acierto inicial** de los modelos. Muchas de estas noticias ya eran conocidas con meses de antelación y estaban incluidas en los datos de preentrenamiento de los modelos, lo que les proporcionó una ventaja inicial en la tarea específica.

8.2 Evaluación Rigurosa de las Respuestas

El criterio utilizado para evaluar las respuestas generadas por los modelos se definió como altamente restrictivo. Para que una respuesta fuera considerada correcta, debía alcanzar un nivel de similitud superior al 70% respecto a la respuesta esperada. Este umbral elevado exige una precisión casi perfecta, lo que limita la percepción de mejoras significativas.

Si este umbral se redujera, los resultados mostrarían un incremento más notable en el rendimiento de los modelos tras las etapas de ajuste, ya que muchas respuestas generadas, sin ser completamente correctas, demuestran una notable adquisición de conocimiento.

8.3 Requisitos de Recursos

El análisis de los requisitos de recursos de los modelos empleados fue un componente fundamental del proyecto, dado que los entornos con recursos limitados imponen restricciones importantes. Se observaron las siguientes características en términos de memoria GPU:

- **Qwen/Qwen2-1.5B-Instruct**

Este modelo requiere aproximadamente **8 GB de memoria en GPU** para su operación, lo que lo posiciona como una opción viable para entornos con recursos moderados.

- **unsloth/mistral-7b-instruct-v0.2-bnb-4bit**

Gracias a la cuantización a **4 bits**, Mistral 7B ocupa únicamente **6.3 GB de memoria en GPU**, reduciendo significativamente sus necesidades computacionales sin comprometer de manera evidente su rendimiento. Esta característica lo hace especialmente atractivo para escenarios donde la disponibilidad de hardware es limitada.

8.4 Rendimiento del Modelo Qwen

El modelo **Qwen/Qwen2-1.5B-Instruct** fue evaluado a lo largo de las distintas etapas del proyecto, obteniendo los siguientes resultados:

- **Modelo base (preentrenado)**

En su estado preentrenado, el modelo alcanzó un rendimiento del **18.92%**, lo que refleja sus capacidades generales previas a cualquier adaptación específica.

- **Tras entrenamiento autoregresivo e instruct**

La aplicación de técnicas de fine-tuning permitió mejorar su precisión hasta el **22.07%**, representando una ganancia de **+3.15%** gracias a la adaptación al corpus de noticias locales.

- **Tras Aprendizaje por Refuerzo (RLHF)**

La implementación de técnicas de aprendizaje por refuerzo incrementó la precisión del modelo al **23.42%**, resultando en una mejora total del **+4.50%** en comparación con su estado inicial.

8.5 Rendimiento del Modelo Mistral

El modelo **Mistral 7B (cuantizado a 4 bits)** demostró un rendimiento más robusto en comparación con **Qwen**, especialmente tras las etapas de fine-tuning:

- **Modelo base (preentrenado)**

En su estado inicial, Mistral obtuvo un rendimiento del **18.02%**, ligeramente inferior al de Qwen.

- **Tras entrenamiento autoregresivo e instruct**

Este modelo mostró una notable mejora al alcanzar un rendimiento del **28.38%**, lo que implica un incremento de **+10.36%** tras su ajuste al corpus específico.

- **Tras Aprendizaje por Refuerzo (RLHF)**

La precisión del modelo aumentó aún más hasta el **29.28%**, representando una mejora total del **+11.26%** respecto a su desempeño inicial.

8.6 Comparación de Modelos

A continuación se presenta un resumen comparativo del rendimiento de los modelos en las distintas etapas:

Modelo	Sin entrenar	Autoregresivo	RLHF	Mejora total
Qwen 1.5B	18.92%	22.07%	23.42%	+4.50%
Mistral 7B	18.02%	28.38%	29.28%	+11.26%

Mistral 7B cuantizado a 4 bits demostró ser más eficiente en términos de rendimiento y consumo de recursos, además, las mejoras logradas tras las etapas de fine-tuning fueron significativamente mayores en el caso de Mistral.

8.7 Conclusiones de las Pruebas

Los resultados obtenidos permiten extraer varias conclusiones importantes:

- **Eficiencia en Ajuste**

Mistral 7B demostró ser más eficiente para ajustarse a tareas específicas, especialmente tras el fine-tuning, logrando un incremento del **11.26%** en precisión tras esta etapa.

- **Restricciones de Recursos**

La cuantización a **4 bits** de Mistral 7B lo posiciona como una opción altamente competitiva en términos de consumo de recursos, sin que esto afecte de forma significativa su capacidad de adaptación.

- **Impacto del RLHF**

Aunque el Aprendizaje por Refuerzo ofreció mejoras adicionales, su impacto fue menos pronunciado, lo que sugiere que otros factores, como la calidad del entorno de retroalimentación, podrían influir en su efectividad.

9. CONCLUSIONES Y TRABAJO FUTURO

El presente proyecto ha permitido explorar y aplicar técnicas avanzadas de ajuste y mejora de modelos de lenguaje a gran escala (LLM), utilizando herramientas modernas como PEFT y Aprendizaje por Refuerzo (RLHF). A través de un diseño sistemático, se implementaron y evaluaron metodologías para adaptar dos modelos preentrenados, **Qwen 1.5B** y **Mistral 7B**, a tareas específicas mediante Fine-Tuning. Aunque los resultados evidencian mejoras en el rendimiento, también abren nuevas vías de investigación para futuros trabajos.

9.1 Conclusiones Principales

- **Fine-Tuning como Método Eficiente de Adaptación:**

El Fine-Tuning mediante técnicas PEFT ha demostrado ser una estrategia altamente efectiva para mejorar el rendimiento de los modelos con un uso optimizado de recursos. Este enfoque permitió aumentos significativos en precisión, especialmente en el caso de **Mistral 7B**, que logró una mejora de **más del 10%** tras esta etapa.

- **Impacto Limitado del Aprendizaje por Refuerzo:**

Aunque el Aprendizaje por Refuerzo contribuyó a incrementos adicionales en el rendimiento, el impacto de esta técnica fue menor en comparación con el resto de técnicas de fine-tuning. La mejora en precisión tras RLHF se limitó a un rango de **1-2%** en ambos modelos, lo que sugiere que su potencial podría depender de factores como el diseño del entorno de entrenamiento o la calidad de los datos de retroalimentación.

- **Desempeño Relativo de los Modelos:**

Mistral 7B superó consistentemente a Qwen 1.5B en todas las etapas de entrenamiento, consolidándose como la opción más robusta para este proyecto. Su menor consumo de memoria GPU, combinado con una mayor capacidad de adaptación, lo convierte en una alternativa altamente eficiente en términos de recursos y precisión.

9.2 Trabajo Futuro

Dado el desempeño limitado del Aprendizaje por Refuerzo observado en este proyecto, es pertinente considerar otras técnicas y enfoques para mejorar aún más los resultados. Algunas líneas de investigación futura incluyen:

- **Evaluación de Otras Técnicas de Ajuste:**

Experimentar con métodos como **prompt tuning** o **instruction tuning**, que permiten ajustar modelos con menor requerimiento de recursos computacionales y sin la necesidad de modificar directamente sus parámetros.

- **Incorporación de Evaluadores Humanos:**

Aunque en este proyecto se utilizó un modelo de similitud para medir la calidad de las respuestas, un sistema de evaluación basado en retroalimentación humana podría aportar mayor precisión y refinamiento.

- **Ampliación del Corpus de Entrenamiento:**

Incorporar datos más diversos o específicos para cubrir un mayor rango de casos de uso, lo que podría mejorar la generalización y el rendimiento del modelo.

- **Experimentación con Técnicas de Ensemble:**

Combinar las predicciones de múltiples modelos ajustados para mejorar la precisión y robustez en tareas complejas.

- **Implementación de Métodos Avanzados de RLHF:**

Considerar enfoques avanzados de aprendizaje por refuerzo, como el uso de políticas jerárquicas o entornos simulados más realistas, para explorar su impacto en tareas altamente dinámicas.

- **Evaluación de Nuevos Modelos:**

Con el ritmo acelerado de innovación en LLM, futuros trabajos podrían considerar la evaluación de modelos emergentes con arquitecturas más eficientes o capacidades específicas para fine-tuning.

9.3 Reflexión Final

El proyecto ha logrado cumplir con sus objetivos principales, demostrando la efectividad de las técnicas implementadas para ajustar modelos de lenguaje a tareas específicas. Sin embargo, los hallazgos obtenidos también ponen de manifiesto las limitaciones de ciertas metodologías y abren oportunidades para explorar enfoques más avanzados. Este trabajo establece una base sólida para futuras investigaciones en la optimización de LLM, destacando la importancia de seguir innovando en este campo en rápida evolución.

10. BIBLIOGRAFÍA

(Russell & Norvig, 2021: , (Russell & Norvig, 2021),
(Vaswani, y otros, 2017: , (Vaswani, y otros, 2017),
(Houlsby & al., 2021: , (Houlsby & al., 2021),
(Dettmers, Lewis, Shleifer, & Zettlemoyer, 2021: , (Dettmers, Lewis, Shleifer, & Zettlemoyer, 2021),
(Christiano, y otros, 2017: , (Christiano, y otros, 2017),
(Brown, y otros, Language models are few-shot learners, 2020: , (Brown, y otros, Language models are few-shot learners, 2020),
(Devlin, Chang, Lee, & Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019: , (Devlin, Chang, Lee, & Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019),
(Marcus & Davis, 2019: , (Marcus & Davis, 2019),
(Raffel, y otros, 2020: , (Raffel, y otros, 2020),
(Ziegler, y otros, 2019: , (Ziegler, y otros, 2019),
(Vig & Belinkov, 2019: , (Vig & Belinkov, 2019),
(Stiennon, y otros, 2020: , (Stiennon, y otros, 2020),
(Mitchell, 2018: , (Mitchell, 2018),
(Jurafsky & Martin, 2021: , (Jurafsky & Martin, 2021),
(Houlsby, y otros, 2019: , (Houlsby, y otros, 2019),
(Liang & al., 2021: , (Liang & al., 2021),
(Belinkov & Glass, 2017: , (Belinkov & Glass, 2017),
(Brown T. , Mann, Ryder, Subbiah, & Kaplan, 2020: , (Brown T. , Mann, Ryder, Subbiah, & Kaplan, 2020),
(Radford, y otros, 2019: , (Radford, y otros, 2019),
(Pfeiffer, Kamath, Rücklé, Cho, & Gurevych, AdapterHub: A Framework for Adapting Transformers, 2020: , (Pfeiffer, Kamath, Rücklé, Cho, & Gurevych, AdapterHub: A Framework for Adapting Transformers, 2020),
(Hu, y otros, LoRA: Low-rank adaptation of large language models, 2021: , (Hu, y otros, LoRA: Low-rank adaptation of large language models, 2021),
(Lester, Al-Rfou, & Constant, 2021: , (Lester, Al-Rfou, & Constant, 2021),
(Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017: , (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017),
(Ouyang, y otros, 2022: , (Ouyang, y otros, 2022),
(Reitz, 2024: , (Reitz, 2024),
(Richardson, 2024: , (Richardson, 2024),
(OpenAI, 2024: , (OpenAI, 2024),
(JSON.org, 2024: , (JSON.org, 2024),