UNIVERSIDAD DE CONCEPCIÓN

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

EVALUACIÓN DEL USO DE FINE-TUNING EN MODELOS DE LENGUAJE GRANDE COMO HERRAMIENTA DE APRENDIZAJE AJUSTADA A LAS ÁREAS DE FÍSICA Y MATEMÁTICAS EN LA EDUCACIÓN

POR

Luis Andrés Melita Cruces

Memoria de Título presentada a la Facultad de Ingeniería de la Universidad de Concepción para optar al título profesional de Ingeniero Civil Industrial

Profesor Guía

Carlos Camilo Navarrete Lizama

Profesora Co-Guía

Alejandra Marcela Maldonado Trapp

Agosto 2024

Concepción (Chile)

Dedicatoria

A mi familia, por su apoyo incondicional y por enseñarme el valor del esfuerzo y la perseverancia. Sus sacrificios y consejos me han guiado en cada etapa de mi vida. Sus palabras de aliento y su comprensión han sido fundamentales para mi crecimiento personal y profesional. Cada logro alcanzado es también suyo.

A mis profesores y mentores, por guiarme y compartir su conocimiento conmigo. Su paciencia, dedicación y sabiduría han sido importantes en mi camino académico, su confianza en mis capacidades me ha inspirado a superar mis propios límites.

A mis amigos, por estar a mi lado en cada paso del camino y por su constante ánimo. Su compañía y amistad han hecho de este viaje una experiencia más rica y gratificante.

A todos aquellos que creen en el poder de la educación y la investigación, aquellos que inspiran a seguir explorando y aprendiendo cada día.

Agradecimientos

Quiero expresar mi más profundo agradecimiento a todas las personas que han hecho posible la realización de este proyecto.

En primer lugar, agradezco a mi profesor guía, Carlos Camilo Navarrete Lizama, y a mi profesora co-guía, Alejandra Marcela Maldonado Trapp, por su invaluable orientación, paciencia y apoyo durante todo el proceso de investigación. Sus conocimientos, consejos y retroalimentación han sido esenciales para la culminación de esta Memoria de Título.

A la Facultad de Ciencias Físicas y Matemáticas, Facultad de Ingeniería y al Departamento de Ingeniería Industrial de la Universidad de Concepción, por proporcionar los recursos y el entorno académico necesarios para llevar a cabo este trabajo. Agradezco a todos los profesores y personal administrativo que han contribuido a mi formación profesional y académica.

Agradezco también a mis compañeros de estudio, por los momentos de colaboración y aprendizaje compartido. Sus perspectivas y sugerencias han enriquecido significativamente este proyecto.

Finalmente, quiero agradecer a todas las personas que de alguna manera han contribuido a este trabajo, ya sea a través de discusiones académicas, apoyo técnico o simplemente ofreciendo su amistad y apoyo moral. Cada uno de ellos han dejado una huella en este proyecto y en mi desarrollo como profesional.

Sumario

El presente trabajo titulado *"EVALUACIÓN DEL USO DE FINE-TUNING EN MODELOS DE LENGUAJE GRANDE COMO HERRAMIENTA DE APRENDIZAJE AJUSTADA A LAS ÁREAS DE FÍSICA Y MATEMÁTICAS EN LA EDUCACIÓN"*, tiene como objetivo principal desarrollar estrategias de fine-tuning para modelos de lenguaje grande pre-entrenados, con el propósito de optimizar su desempeño en la resolución de problemas matemáticos y físicos en contextos educativos.

Se utilizó una metodología mixta que incluye la cuantización de modelos, preparación del entorno de desarrollo por limitaciones software, selección de modelos pre-entrenados, recopilación y preparación de datos, elección de técnicas de fine-tuning, y evaluación del rendimiento de los modelos entrenados. Se llevaron a cabo experimentos utilizando la técnica fine-tuning más adecuada para las herramientas disponibles, y para la tarea definida de mejorar el rendimiento de los LLMs en tareas específicas de matemáticas y física.

Se concluyó que el fine-tuning de LLMs es una técnica eficaz para adaptar modelos pre-entrenados a tareas educativas específicas, mejorando su rendimiento y utilidad en la enseñanza de matemáticas y física. Los resultados sugieren que el uso de LLMs ajustados puede ser una herramienta valiosa para el desarrollo educativo, proporcionando no solo respuestas precisas, sino también un medio para que los estudiantes comprendan mejor los conceptos y mejoren sus habilidades de resolución de problemas.

Abstract

The present work, titled *"EVALUACIÓN DEL USO DE FINE-TUNING EN MODELOS DE LENGUAJE GRANDE COMO HERRAMIENTA DE APRENDIZAJE AJUSTADA A LAS ÁREAS DE FÍSICA Y MATEMÁTICAS EN LA EDUCACIÓN"*, has the main objective of developing fine-tuning strategies for pre-trained large language models in order to optimize their performance in solving mathematical and physical problems within educational contexts.

A mixed methodology was used, including model quantization, setting up the development environment due to software limitations, selection of pre-trained models, data collection and preparation, choice of fine-tuning techniques, and evaluation of the trained models' performance. Experiments were conducted using the most suitable fine-tuning technique for the available tools and for the defined task of improving LLM performance on specific mathematics and physics tasks.

It was concluded that the fine-tuning of LLMs is an effective technique to adapt pre-trained models to specific educational tasks, improving their performance and usefulness in the teaching of mathematics and physics. The results suggest that fine-tuned LLMs can be a valuable tool for educational development, providing not only accurate answers but also a means for students to better understand concepts and improve their problem-solving skills.

Tabla de Contenidos

Lista de Tablas

Lista de Figuras

Capítulo 1

Introducción

Un modelo de lenguaje grande (large language model, LLM, por su sigla en inglés) es un tipo de inteligencia artificial avanzada especializada en el procesamiento del lenguaje. Se trata de un sistema informático complejo entrenado con enormes cantidades de texto para analizar patrones y relaciones entre palabras. Con el auge de la inteligencia artificial, los LLMs se han convertido en una herramienta crucial y han traído consigo avances significativos en el campo del procesamiento del lenguaje natural. Se ha demostrado que estos modelos, entrenados con grandes cantidades de datos, son capaces de producir textos coherentes, relevantes y de calidad similar a la humana (Wang et al., 2024).

Dentro del ámbito educativo se ha encontrado que, los LLMs pueden lograr un desempeño a nivel de un estudiante en pruebas estandarizadas en una variedad de áreas, tales como las matemáticas, física y ciencias de la computación (con la generación de código). Estos modelos han demostrado ser eficaces tanto en problemas de opción múltiple como en problemas de desarrollo (OpenAI et al., 2024). El estudio de Susnjak (2022), revela que ChatGPT (chatbot basado en un LLM) es capaz de generar respuestas consistentes y lógicas en una variedad de disciplinas, lo que subraya su potencial para asistir en la educación. El análisis cuantitativo de Malinka et al. (2023), muestra que los estudiantes que utilizan ChatGPT para generar, revisar y refinar sus respuestas, tienden a desempeñarse mejor que los estudiantes promedio en cursos de seguridad informática. Estos hallazgos destacan la capacidad de los LLMs como herramientas efectivas para el desarrollo educativo, proporcionando no solo respuestas inmediatas, sino también un medio para que los estudiantes comprendan mejor los conceptos y mejoren sus habilidades de resolución de problemas. Sin embargo, los LLMs no poseen un conocimiento absoluto, por eso es que uno de los desafíos más relevantes en la actualidad es resolver la limitación de los modelos en su capacidad para realizar tareas que necesiten de un razonamiento matemático complejo. Aunque existen versiones más pequeñas y más grandes de un mismo modelo, las versiones más grandes y avanzadas, con un mayor número de parámetros (cantidad de variables que el modelo

utiliza para aprender y representar el conocimiento a partir de los datos de entrenamiento), presentan un mejor rendimiento (un ejemplo son los modelos Llama 3 de Meta AI, con 8 y 70 mil millones de parámetros). No obstante, estos modelos avanzados también requieren un mayor poder computacional, lo que incrementa los costos asociados a su uso (Yuan et al., 2023). Esta necesidad de recursos computacionales más potentes hace que el acceso a los modelos más avanzados no sea factible para todos, limitando su disponibilidad y utilidad en contextos educativos más amplios.

Aunque los LLMs pueden ser herramientas valiosas para ciertas tareas, su aplicabilidad puede estar limitada por la naturaleza de los datos con los que fueron entrenados y por su capacidad para generalizar más allá de esos datos, los LLMs no siempre tienen una comprensión profunda de los conceptos matemáticos subyacentes en esos datos. A diferencia de los humanos, que pueden aprender y aplicar principios lógicos o matemáticos para resolver problemas. Los LLMs no ejecutan algoritmos matemáticos internamente ni deducen nuevas reglas matemáticas. Su proceso es puramente predictivo, basado en correlaciones encontradas en el texto durante el entrenamiento. Esto puede llevar a que los LLMs produzcan resultados matemáticamente incorrectos o sin sentido, incluso si la sintaxis y la gramática del texto generado son correctas. Algunos LLMs están diseñados para procesar lenguaje natural y son expertos solo en esa tarea, lo que significa que pueden tener dificultades para manejar símbolos y ecuaciones matemáticas. Esto se debe a que se entrenan con conjuntos de datos específicos para una tarea o problema en particular y pueden tener dificultades para generalizar sus habilidades a nuevos problemas o contextos. Si los datos de entrenamiento están sesgados hacia ciertos tipos de problemas matemáticos, el LLM puede ser menos efectivo para resolver otros tipos de problemas, por ejemplo, un modelo de lenguaje que fue entrenado principalmente con problemas de álgebra y ecuaciones lineales, habiendo visto muchos textos relacionados con estos temas, puede predecir respuestas o continuaciones de textos que se ajustan a este tipo de matemática, sin embargo, si se utiliza este modelo para resolver un problema de geometría, como calcular el área de un círculo, si el modelo no fue expuesto suficientemente a principios y problemas de geometría durante su entrenamiento, es posible que genere respuestas incoherentes o incorrectas. Esto se debe a que no ha “aprendido” los patrones de texto que se usan típicamente para describir y resolver problemas de geometría. En el ámbito educativo, esto plantea un desafío significativo.

El presente trabajo aborda la relevancia del fine-tuning en los modelos de lenguaje grande y su potencial para mejorar su desempeño en aplicaciones educativas, específicamente en la enseñanza

de matemáticas y física. El fine-tuning es una técnica utilizada en el aprendizaje profundo y en el entrenamiento de modelos de lenguaje grande, esta permite adaptar un modelo pre-entrenado a una tarea específica. Este proceso implica tomar un modelo que ya ha sido entrenado con una gran cantidad de datos generales y especializarlo mediante el entrenamiento adicional con un conjunto de datos específico y relevante para la tarea que se desea mejorar. La investigación documentada en este trabajo se centra en desarrollar estrategias efectivas de fine-tuning para LLMs pre-entrenados, con el objetivo de optimizar su desempeño en la resolución de problemas matemáticos y físicos.

1.1 Objetivo General

Desarrollar estrategias de fine-tuning para modelo de lenguaje grande pre-entrenados que ayuden a mejorar su rendimiento con respecto a modelos base, aplicado a ámbitos educativos y académicos, ajustado a las áreas de física y matemáticas, mejorando su capacidad en la resolución de problemas para la futura aplicación en revisión, retroalimentación y preparación de exámenes.

1.2 Objetivos Específicos

- Explorar y seleccionar modelo de lenguaje grande pre-entrenados que se adecuen mejor a la tarea de asistencia educativa.
- Recopilar conjuntos de datos basados en problemas con sus respuestas en las áreas de física y matemáticas, se deberá establecer la estructura del conjunto de datos necesaria para el entrenamiento.
- Entrenar los modelo de lenguaje grande pre-entrenados con los conjuntos de datos recopilados.
- Evaluar el rendimiento de los modelo de lenguaje grande entrenados y comparar su rendimiento con sus respectivos modelos base no entrenados para la tarea en cuestión.

Capítulo 2

Marco Teórico

El presente marco teórico abordará los distintos fundamentos de los LLMs y su aplicación en el ámbito educativo, especialmente su aplicación en las áreas de física y matemáticas, además de distintas técnicas para mejorar el rendimiento de los LLMs. Lo siguiente pretende comprender los diferentes temas abordados en el desarrollo de esta Memoria de Título.

2.1 Inteligencia Artificial, Aprendizaje Automático y Aprendizaje Profundo

La inteligencia artificial (artificial intelligence, AI, por su sigla en inglés) es un campo de la informática que se ocupa de la creación de sistemas inteligentes capaces de realizar tareas que normalmente requieren de la inteligencia humana. Estos sistemas se basan en algoritmos que pueden aprender y mejorar su rendimiento a partir de la experiencia. La AI ha experimentado un crecimiento exponencial en los últimos años, impulsada por el aumento de la potencia de cómputo y la disponibilidad de grandes cantidades de datos (Grace et al., 2024).

Un sub-campo de la AI es el aprendizaje automático (machine learning, ML, por su sigla en inglés), que se centra en el desarrollo de sistemas que pueden aprender a partir de los datos sin ser programados explícitamente. En otras palabras, los sistemas de ML son capaces de mejorar su rendimiento en una tarea específica en el tiempo a medida que se exponen a más datos. Los sistemas de ML se basan en algoritmos que pueden identificar patrones y relaciones en los datos. Estos algoritmos se entrenan utilizando conjuntos de datos, que son colecciones de ejemplos etiquetados. Los ejemplos pueden ser datos de texto, imágenes, audio o cualquier otro tipo de información.

A su vez, un tipo de ML es el aprendizaje profundo (deep learning, DL, por su sigla en inglés). El DL se basa en redes neuronales artificiales, las cuales están inspiradas en el funcionamiento del cerebro humano (LeCun et al., 2015). Estas redes están compuestas por capas de neuronas artificiales, las cuales se conectan entre sí mediante sinapsis artificial. Las neuronas procesan información y transmiten señales a otras neuronas, permitiendo que la red aprenda patrones complejos a partir de los datos. Las redes neuronales utilizadas en el DL tienen múltiples capas, lo que permite que aprendan representaciones de los datos en diferentes niveles de abstracción. Las capas iniciales extraen características simples de los datos, mientras que las capas posteriores combinan estas características para formar representaciones más complejas.

¿Cómo se relacionan estos términos? La AI es el objetivo final, busca crear máquinas o aplicaciones que puedan pensar, aprender y actuar como humanos. Por otro lado, el ML ayuda a las inteligencias artificiales en el desarrollo de algoritmos para que puedan analizar y aprender de los datos por sí mismos, sin que un programador necesite escribir reglas o instrucciones específicas que definan cómo realizar tareas específicas. A su vez, el DL es un tipo de ML que utiliza redes neuronales artificiales con múltiples capas para aprender representaciones complejas de los datos.

2.2 La Arquitectura de los Transformadores

La arquitectura de los Transformers, presentada en el artículo *"Attention is All You Need"* de Vaswani et al. (2023), revolucionó el campo del procesamiento del lenguaje natural. Esta arquitectura se basa en mecanismos de autoatención, que permiten al modelo evaluar la importancia de diferentes palabras en una oración sin necesidad de recurrir a estructuras secuenciales como las redes neuronales recurrentes (tipo de red neuronal diseñada para reconocer patrones en secuencias de datos). Los Transformers emplean capas totalmente conectadas, llamadas Multi-Head Attention y Feed-Forward, organizadas en bloques de codificador y decodificador, lo que permite una paralelización más eficiente y un aprendizaje de dependencias a largo plazo. Esta capacidad de manejar contextos largos y complejos de manera eficiente estableció la base para el desarrollo de los LLMs, que pueden generar y entender texto de manera mucho más efectiva y coherente a gran escala.

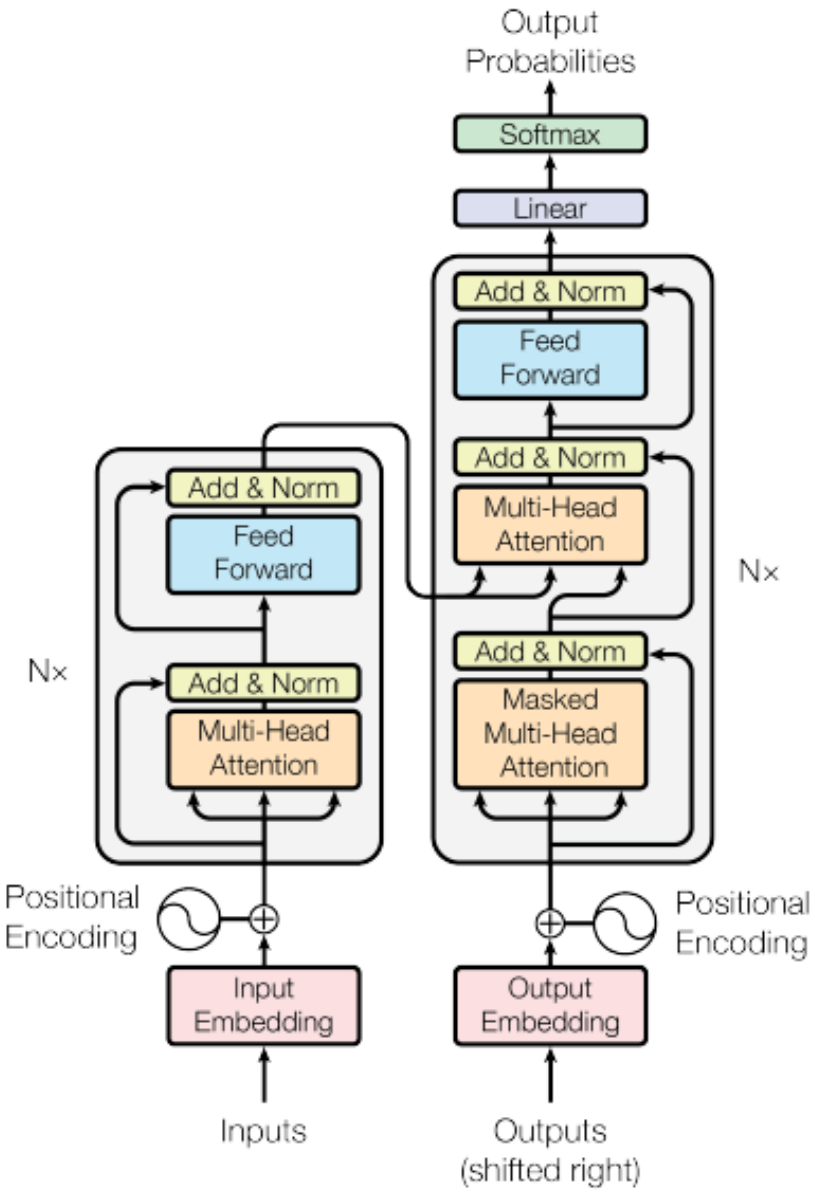

Figura 2.1: Arquitectura del Transformer (Vaswani et al., 2023).

2.2.1 Principales Componentes de la Arquitectura de Transformadores

El mecanismo de autoatención escalonada permite al modelo asignar diferentes pesos a distintas palabras de una oración mediante la multiplicación de matrices de consulta, clave y valor, mientras que Multi-Head Attention utiliza múltiples cabezas de atención para capturar diferentes tipos de relaciones contextuales simultáneamente. Dado que los Transformers no procesan datos secuenciales

en orden, se introduce información posicional para que el modelo pueda diferenciar la posición relativa de las palabras. Después del mecanismo de atención, se aplican capas de normalización y redes neuronales Feed-Forward que procesan la información atendida. En la generación de texto, la capa de decodificación utiliza un mecanismo similar de atención para generar la salida, prestando atención a las palabras generadas y al contexto de entrada.

2.2.2 Funcionamiento del Transformador

La entrada se convierte en embedding (representaciones vectoriales densas de las palabras), a las cuales se les añade información posicional para mantener el orden secuencial. La capa Multi-Head Attention calcula los pesos de atención utilizando las matrices de consulta, clave y valor, y los resultados de estas atenciones se combinan y pasan a través de una red Feed-Forward. Finalmente, la salida de las capas Multi-Head Attention y Feed-Forward se convierte en predicciones finales mediante una capa densa y una función de activación.

2.3 Modelos de Lenguaje Grande

Previamente, se introdujo el concepto de LLM, un tipo de inteligencia artificial avanzada. Una de las ideas más importantes de estos modelos es que pueden procesar información de manera similar a como lo hace el cerebro humano, teniendo la capacidad de analizar patrones y relaciones entre palabras, lo que les permite realizar una variedad de tareas relacionadas con el lenguaje (Vaswani et al., 2023). Los LLMs han revolucionado el procesamiento del lenguaje natural, abriendo un abanico de posibilidades para la interacción humano-máquina. Su capacidad para generar texto similar al humano y entablar conversaciones fluidas los convierte en herramientas ideales para la creación de chatbots. Los LLMs se caracterizan por su capacidad para procesar y generar grandes cantidades de texto, aprendiendo patrones y relaciones lingüísticas a partir de vastos conjuntos de datos. Esta habilidad les permite comprender el lenguaje natural, interpretando el significado de frases, oraciones y párrafos, extrayendo información relevante y entendiendo el contexto en el que se emplea el lenguaje; generar texto similar al humano, produciendo texto fluido y coherente en diversos formatos como resúmenes,

traducciones, respuestas a preguntas y contenido creativo; mantener conversaciones de manera natural, respondiendo preguntas, siguiendo instrucciones y adaptándose al contexto de la interacción.

El entrenamiento de un LLM implica exponerlo a una enorme cantidad de texto y código de diversas fuentes, como libros, artículos, sitios web, código abierto y conversaciones. El proceso de obtención del conjunto de entrenamiento para un LLM es meticuloso y abarca varias etapas. Los investigadores y desarrolladores de modelos seleccionan una diversidad de fuentes de datos textuales para asegurar una amplia cobertura de conocimiento y perspectivas. Estas fuentes pueden incluir libros, artículos académicos, sitios web de noticias, entradas de blogs, foros en línea, código fuente y otros materiales disponibles públicamente o mediante licencias. Los datos se recopilan de las fuentes elegidas. Esto puede implicar procesos automatizados como web scraping (extracción de datos de sitios web) y la utilización de APIs (conjunto de instrucciones y normas que define cómo una aplicación puede solicitar información o funcionalidad de otra aplicación) para recoger datos de plataformas de medios sociales, repositorios de código abierto, etc. Los datos recopilados suelen requerir limpieza y preprocesamiento para ser útiles en el entrenamiento. Esto incluye eliminar datos duplicados, corregir errores, filtrar contenido irrelevante o inapropiado y transformar los datos en un formato estandarizado que el modelo pueda procesar eficazmente. Antes de alimentar los datos al modelo, se convierten en *tokens*, bloques básicos del texto que el modelo puede entender. Esto incluye dividir el texto en palabras, sub-palabras o caracteres, según el enfoque del modelo. En algunos casos, especialmente cuando se manejan datos sensibles, se realiza un proceso de anonimización para eliminar o alterar información que podría usarse para identificar a individuos. Se verifica que el conjunto de datos sea representativo y diverso, cubriendo una amplia gama de temas, estilos de escritura y perspectivas para evitar sesgos en las respuestas del modelo. Finalmente, los datos se dividen en conjuntos de entrenamiento, validación y prueba. El conjunto de entrenamiento es el más grande y se usa para ajustar los parámetros del modelo, el conjunto de validación se usa para ajustar hiperparámetros y evitar el sobre-ajuste, y el conjunto de prueba se utiliza para evaluar la capacidad general del modelo antes de su despliegue. A medida que el modelo procesa esta información, aprende a identificar patrones y relaciones entre las palabras, lo que le permite generar texto coherente y relevante Basu et al. (2024).

2.3.1 Modelos

En el contexto del procesamiento de lenguaje natural (natural language processing, NLP, por su sigla en inglés), los LLMs se utilizan para una variedad de tareas que van desde la generación de texto hasta la traducción automática y la respuesta a preguntas. A continuación se presentan algunos de los modelos que serán mencionados en esta Memoria de Título.

Gemini

Gemini es un modelo de lenguaje desarrollado por Google AI que ha evolucionado más allá de su versión inicial conocida como Bard. Gemini está diseñado como una herramienta de procesamiento del lenguaje natural (NLP) multimodal avanzada, capaz de comprender y generar no solo texto, sino también interpretar y responder a contenido visual como imágenes (Team et al., 2024).

GPT

GPT o Generative Pre-trained Transformer, es una serie de modelos de lenguaje desarrollados por OpenAI. La versión más reciente, GPT-4, es capaz de generar texto coherente y relevante en una amplia gama de contextos. GPT se utiliza en aplicaciones que requieren la generación de texto, asistencia en redacción y traducción automática (Yenduri et al., 2023).

Llama

Llama es un modelo de lenguaje desarrollado por Meta AI (anteriormente Facebook). Llama está diseñado para ser eficiente y accesible, permitiendo su uso en una variedad de aplicaciones de procesamiento del lenguaje natural. Este modelo se enfoca en ofrecer un equilibrio entre rendimiento y requisitos de recursos computacionales, lo que lo hace adecuado para investigaciones académicas y aplicaciones prácticas (Touvron et al., 2023).

Mistral

Mistral es un modelo de lenguaje avanzado desarrollado por Mistral AI, se enfoca en la eficiencia y el rendimiento en tareas específicas de NLP. Mistral se utiliza en aplicaciones donde se requiere un procesamiento rápido y preciso del lenguaje. Este modelo se caracteriza por su capacidad de adaptarse rápidamente a nuevas tareas y su habilidad para manejar grandes volúmenes de datos (Jiang et al., 2023).

PaLM

PaLM es un modelo de lenguaje grande desarrollado por Google AI. Está diseñado para aprovechar la arquitectura Pathways, que permite a un solo modelo ser entrenado y utilizado para miles o millones de tareas diferentes. PaLM se destaca por su capacidad para manejar tareas complejas y multidisciplinarias, lo que lo convierte en una herramienta poderosa para aplicaciones avanzadas de inteligencia artificial (Chowdhery et al., 2022).

2.3.2 Chatbots Basados en LLMs

Un chatbot es un programa informático diseñado para la simulación de una conversación, comúnmente se encuentran a través de interfaces de chat, ya sea texto o voz en aplicaciones web. Estos programas utilizan inteligencia artificial para entender el lenguaje natural y proporcionar respuestas.

Los modelos de lenguaje grande (LLM) como GPT, desarrollado por OpenAI, y Gemini, de Google AI, ofrecen interfaces conversacionales que permiten a los usuarios interactuar de manera fluida y natural. ChatGPT, con la versión GPT-4, se especializa en conversaciones y generación de texto, mientras que Gemini es una herramienta conversacional multimodal que puede procesar tanto texto como imágenes. Ambos modelos son capaces de entender y responder a preguntas complejas en lenguaje natural, adaptándose al contexto para proporcionar respuestas precisas y relevantes.

Los chatbots basados en LLMs se caracterizan por su capacidad de procesamiento y generación de lenguaje natural, su personalización y su aprendizaje continuo. Estas características los convierten

en herramientas valiosas para el ámbito educativo. En el contexto del estudio que se llevará a cabo, estos chatbots tienen un gran potencial para mejorar la enseñanza y el aprendizaje en las áreas de física y matemáticas. Algunas de sus aplicaciones más relevantes incluyen la explicación de conceptos complejos, la resolución de problemas, la práctica y evaluación. La implementación de chatbots basados en LLM ofrece beneficios, como el acceso a la educación, el aprendizaje personalizado, la flexibilidad, la interactividad y la retroalimentación inmediata. A pesar de sus beneficios, el uso de chatbots basados en LLM en educación también presenta algunos desafíos, como la precisión y confiabilidad, la evaluación del impacto, la aceptación por parte de los educadores y los aspectos éticos (Hellas et al., 2024).

2.4 LLMs en la Educación

Aunque los LLM han tenido éxito en varios campos, la creación de un sistema educativo basado en LLMs sigue siendo un desafío para la amplia gama de habilidades educativas requeridas. El paradigma de los LLMs en relación con el razonamiento matemático se centra en mejorar la capacidad de los LLMs para resolver problemas matemáticos. Actualmente, los LLMs han mostrado avances en operaciones matemáticas básicas, pero aún enfrentan desafíos con cálculos que involucran valores numéricos grandes y ecuaciones más complejas (Li et al., 2024). Los LLMs están comenzando a abordar problemas que requieren comprender información tanto textual como visual, como en problemas geométricos, aunque esto plantea requisitos altos para la formación y calidad de los datos.

Debido a que hay LLMs lo suficientemente grandes como para responder problemas complejos de matemática, también se busca evaluar su habilidad para identificar respuestas incorrectas basadas en conceptos erróneos. Algunos experimentos muestran que, aunque los LLMs pueden responder correctamente a preguntas de matemáticas de nivel escolar, tienen dificultades para replicar el comportamiento de aprendices novatos que cometen errores y de tutores expertos que entienden por qué se cometen esos errores (Liu et al., 2023a), puesto que al momento de indicarle a un LLM que una respuesta incorrecta es correcta, muchos no tienen la capacidad de discutir esta idea. El paradigma actual sugiere que hay oportunidades significativas para mejorar las capacidades de razonamiento matemático de los LLMs, lo que podría ser beneficioso para sistemas de tutoría inteligente en la educación.

2.5 Prompt Engineering

La ingeniería de instrucciones (prompt engineering, en inglés), es la técnica de crear prompts para obtener una respuesta de un LLM. Estas instrucciones pueden ser preguntas, solicitudes o instrucciones de escritura creativa.

2.5.1 Técnicas de Prompt Engineering

Zero-Shot Learning, se trata de presentar al modelo un problema sin proporcionar ejemplos previos. El modelo debe generar una respuesta basándose únicamente en su entrenamiento previo. Ejemplo: *" ¿Qué es la fotosíntesis?"*.

One-Shot Learning, se proporciona un solo ejemplo para guiar al modelo. Este ejemplo actúa como un contexto que ayuda al modelo a entender mejor la pregunta o tarea. Ejemplo: *"Traducción: 'casa' en inglés es 'house'. ¿Cómo se dice 'libro' en inglés?"*.

Few-Shot Learning, se proporcionan varios ejemplos para establecer un patrón o formato que el modelo debe seguir. Esto es útil para tareas complejas o cuando se desea una respuesta en un formato específico. Ejemplo: *"Resumen: Texto A: [Texto]. Texto B: [Texto]. Resumir Texto C: [Texto]"*.

Chain of Thought Prompting, se incentiva al modelo a desglosar su proceso de pensamiento en pasos lógicos antes de llegar a una conclusión. Esto es particularmente útil para problemas complejos o de razonamiento. Ejemplo: *"Para calcular el área de un círculo, primero necesito el radio. El radio es la mitad del diámetro. Luego uso la fórmula del área..."*.

Prompt Chaining, consiste en utilizar las respuestas de un modelo como nuevos prompts para obtener información adicional o clarificaciones. Es una técnica iterativa donde cada respuesta del modelo informa el siguiente prompt. Ejemplo: *" ¿Qué países forman el Reino Unido?* $\rightarrow$ *¿Cuál es la capital de Escocia?"*.

Negative Prompting, especificar lo que no se quiere en la respuesta. Esto ayuda a evitar respuestas indeseadas. Ejemplo: *"Explica qué es un reptil sin mencionar cocodrilos ni serpientes"*.

Hybrid Prompting, combinar diferentes técnicas de prompt engineering para abordar tareas complejas. Por ejemplo, usar One-Shot Learning para establecer un formato y Chain of Thought para resolver un problema de matemáticas dentro de ese formato.

En esta Memoria de Título, se ha decidido no evaluar técnicas de prompt engineering debido a que el enfoque principal del estudio está en el fine-tuning de modelos de lenguaje grande (LLMs) pre-entrenados. El fine-tuning permite ajustar los modelos de manera específica para tareas concretas dentro de las áreas de física y matemáticas, mejorando así su rendimiento y precisión. Por otro lado, el prompt engineering se centra en la creación de instrucciones detalladas para guiar la generación de respuestas por parte de los modelos. Aunque esta técnica es útil para optimizar interacciones a corto plazo, no proporciona las mejoras estructurales en el rendimiento del modelo que se pueden lograr mediante el fine-tuning. Además, el fine-tuning aborda las limitaciones inherentes de los LLMs en el razonamiento matemático complejo, una capacidad crítica para el objetivo educativo de esta investigación. Por estas razones, se ha priorizado el estudio y aplicación del fine-tuning como la técnica principal de ajuste y mejora del rendimiento de los LLMs en contextos educativos.

2.6 Fine-Tuning, Mejora del Rendimiento y Ajuste de los LLMs

El desarrollo de chatbots basados en modelos de lenguaje grande requiere una cuidadosa selección de estrategias para maximizar su efectividad. Una decisión crucial para los desarrolladores es cómo gestionar el modelo subyacente. Un enfoque que se destaca en este contexto es el fine-tuning (ajuste fino, en español). En el contexto del aprendizaje profundo, el fine-tuning es una técnica que permite adaptar un LLM pre-entrenado a una tarea específica. Se basa en la idea de que un modelo pre-entrenado con una gran cantidad de datos generales puede ser reutilizado y especializado, permite aprovechar el conocimiento y las capacidades de un LLM pre-entrenado para la generación de texto y se especializa para resolver problemas específicos. Este proceso implica entrenar el modelo con datos específicos de la tarea en cuestión, aprovechando el conocimiento y las habilidades ya adquiridas en el entrenamiento inicial (Balaguer et al., 2024). El fine-tuning ha demostrado ser una técnica efectiva para mejorar significativamente el rendimiento de los LLM en una variedad de tareas (Mahapatra and Garain, 2024). Esta técnica permite a los desarrolladores crear aplicaciones de inteligencia artificial

de forma rápida y eficiente, como los chatbots nombrados anteriormente, sin necesidad de entrenar un modelo desde cero, evitando excesivos costos en software.

Al implementar fine-tuning para mejorar el rendimiento de los LLMs, es necesario establecer el objetivo por el cual se quiere realizar el entrenamiento y luego es importante considerar los siguientes pasos:

1. **Preparación del Entorno:** Configurar el entorno de desarrollo necesario para llevar a cabo el fine-tuning. Esto incluye la instalación de las librerías y dependencias requeridas, como frameworks de machine learning, así como la configuración de hardware especializado (GPU/TPU).
2. **Selección del Modelo Pre-Entrenado:** Elegir un modelo que ya ha sido pre-entrenado en una gran cantidad de datos y que sea relevante para la tarea específica que se desea abordar.
3. **Recopilar los Datos:** Crear o recopilar un conjunto de datos de alta calidad para la tarea específica. Es crucial que los datos sean representativos del problema que se intenta resolver y que estén etiquetados de manera adecuada si se trata de un problema supervisado.
4. **Elección de la Técnica de Fine-Tuning:** Seleccionar la técnica de entrenamiento más adecuada para la tarea y el conjunto de datos.
5. **Preparar los Datos:** Los datos deben estar en un formato adecuado, que se ajuste a la arquitectura del modelo seleccionado y a la técnica de fine-tuning elegida.
6. **Ajustar Parámetros de Entrenamiento:** Ajustar los parámetros de manera adecuada puede tener un impacto significativo en el rendimiento del modelo y en el uso de recursos computacionales.
7. **Entrenar Modelo:** Ejecutar el proceso de fine-tuning utilizando el modelo pre-entrenado y el conjunto de datos preparado. Durante este proceso, el modelo ajusta sus pesos y parámetros para mejorar el rendimiento en la tarea específica.
8. **Evaluación del Rendimiento:** Evaluar el rendimiento del modelo entrenado con fine-tuning y compararlo con el modelo pre-entrenado. Utilizar métricas de evaluación adecuadas para la tarea específica y analizar los resultados para determinar si el fine-tuning ha mejorado el rendimiento del modelo.

El fine-tuning es una técnica crucial en el desarrollo y mejora de modelos de lenguaje grande debido a su capacidad para adaptar modelos pre-entrenados a tareas específicas con mayor precisión y eficiencia. Este proceso permite aprovechar el conocimiento general adquirido por el modelo durante su entrenamiento inicial y especializarlo mediante un entrenamiento adicional con datos específicos de la tarea en cuestión. Esto no solo mejora significativamente el rendimiento del modelo en la resolución de problemas complejos, como los matemáticos y físicos, sino que también optimiza el uso de recursos computacionales y reduce los costos asociados al entrenamiento desde cero. El estudio del fine-tuning es fundamental para avanzar en la aplicación de LLMs en contextos educativos, proporcionando herramientas más precisas y útiles para la enseñanza y el aprendizaje en áreas como la física y las matemáticas.

2.6.1 Aplicaciones del Fine-Tuning

Este método es crucial debido a los elevados costos de entrenar un LLM desde cero en términos de hardware. Por ejemplo, el fine-tuning de un modelo avanzado como GPT-3.5 Turbo puede requerir recursos significativos, como se detalla en las políticas de precios de OpenAI para sus servicios de API, el costo por entrenamiento es 8,00 US$ / 1 M training tokens. Si se aprovechan los conocimientos ya recogidos en el modelo pre-entrenado, se puede conseguir un alto rendimiento en una tarea específica con muchos menos datos y recursos. La decisión de ajustar un modelo depende los objetivos y la aplicación de la tarea que se le quiere dar al LLM. A continuación se nombran algunos escenarios clave en los que se debe tomar en cuenta el fine-tuning.

La disponibilidad limitada de datos es una realidad frecuente en muchos contextos, y el fine-tuning se presenta como una solución eficiente para aprovechar al máximo la información disponible. Al aplicar fine-tuning, es posible mejorar significativamente el rendimiento de un modelo, incluso cuando se cuenta con una cantidad reducida de datos específicos para la tarea en cuestión. En términos de optimización de tiempo y recursos, el fine-tuning ofrece ventajas notables. Al omitir las etapas iniciales de un entrenamiento completo, este proceso ahorra tiempo y recursos computacionales, permitiendo que el modelo converja más rápidamente hacia una solución efectiva. Esto resulta especialmente valioso en entornos donde el tiempo y los recursos son limitados. La adaptación a tareas específicas es otra ventaja crucial del fine-tuning. Este proceso permite ajustar un modelo preentrenado para realizar

tareas específicas, como responder preguntas sobre un conjunto de documentos particulares o abordar áreas educativas específicas, como la física y las matemáticas. Esta capacidad de adaptación incrementa la relevancia y precisión del modelo en contextos determinados. El aprendizaje continuo es esencial en escenarios donde la información está en constante cambio. El fine-tuning facilita la actualización continua del modelo, permitiéndole adaptarse a nueva información o cambios en el entorno. Esto asegura que el modelo permanezca relevante y preciso a lo largo del tiempo. La mitigación de sesgos es otra área donde el fine-tuning demuestra su utilidad. Mediante el ajuste fino, es posible reducir o eliminar sesgos presentes en el modelo original, proporcionando datos de entrenamiento más equilibrados y representativos. Esto mejora la equidad y la precisión de las predicciones del modelo. En términos de seguridad y ética, el fine-tuning permite que los chatbots sean más cautelosos al proporcionar diagnósticos y recomendaciones. En contextos donde se trabaja con datos sensibles, el ajuste fino permite entrenar el modelo localmente en un entorno seguro, protegiendo la privacidad y la confidencialidad de la información. Esta característica es fundamental para mantener altos estándares de seguridad y ética en el uso de modelos de lenguaje grande.

2.6.2 Técnicas de Fine-Tuning

El fine-tuning es un proceso crítico en el entrenamiento de LLMs para mejorar su rendimiento en tareas específicas. Existen varias técnicas de fine-tuning que se utilizan comúnmente en la industria y en la investigación. A continuación, se describen algunas de las técnicas más importantes y que posteriormente serán referenciadas en aplicaciones:

SFT

El ajuste fino supervisado (Supervised Fine-Tuning, SFT, por su sigla en inglés) es un método de fine-tuning que se basa en el uso de datos etiquetados para entrenar un modelo pre-entrenado en una tarea específica. Este proceso implica proporcionar ejemplos de entrada junto con las respuestas esperadas, permitiendo que el modelo aprenda a generar respuestas correctas en contextos similares en el futuro (Mecklenburg et al., 2024).

El SFT requiere un conjunto de datos con entradas y salidas etiquetadas que representen correctamente la tarea objetivo. El modelo se ajusta utilizando técnicas de aprendizaje supervisado, donde las predicciones se comparan con las respuestas correctas y se ajustan los parámetros del modelo para minimizar el error. El SFT se utiliza en una variedad de aplicaciones, como traducción automática, clasificación de texto y respuesta a preguntas.

RLHF

El aprendizaje por refuerzo con retroalimentación humana (Reinforcement Learning from Human Feedback, RLHF, por su sigla en inglés) es una técnica que combina el aprendizaje por refuerzo con retroalimentación humana para ajustar un modelo de lenguaje. En este método, el modelo recibe recompensas basadas en la calidad de sus respuestas, que son evaluadas por humanos (Li et al., 2023).

La retroalimentación humana involucra evaluadores humanos que califican la calidad de las respuestas del modelo, proporcionando una señal de recompensa. Utiliza algoritmos de aprendizaje por refuerzo para ajustar los parámetros del modelo en función de las recompensas recibidas. El modelo mejora continuamente a medida que recibe más retroalimentación y ajusta sus respuestas para maximizar las recompensas.

LoRA

La adaptación de bajo rango (Low-Rank Adaptation, LoRA, por su sigla en inglés) es una técnica de fine-tuning que optimiza la atención racional de los modelos de lenguaje utilizando recursos computacionales limitados. LoRA se basa en descomponer las matrices de atención en componentes de bajo rango, lo que reduce la necesidad de complejidad computacional para el entrenamiento (Hu et al., 2021).

La optimización de recursos permite ajustar modelos con menos recursos computacionales al reducir la dimensionalidad de las matrices de atención. La atención racional mejora la eficiencia del modelo al enfocarse en las partes más relevantes del texto de entrada. Esta técnica es ideal para entornos con recursos limitados.

QLoRA

La adaptación cuantificada de bajo rango (Quantized Low-Rank Adaptation, QLoRA, por su sigla en inglés) es una extensión de la técnica LoRA que incorpora la cuantificación de los parámetros del modelo para reducir aún más el uso de memoria y la complejidad computacional (Dettmers et al., 2023).

La cuantificación de parámetros reduce la precisión de los parámetros del modelo para disminuir el tamaño del modelo y acelerar el entrenamiento. Esto permite el entrenamiento de modelos grandes en hardware con recursos de memoria limitados. A pesar de la reducción en la precisión de los parámetros, QLoRA mantiene un desempeño comparable al de los modelos no cuantificados en muchas tareas.

ORPO

Optimización de la preferencia del índice de probabilidades (Odds Ratio Preference Optimization, ORPO, por su sigla en inglés), combina SFT con una penalización basada en la relación de probabilidades entre respuestas favorecidas y desfavorecidas. Este enfoque utiliza una función de log-sigmoide para maximizar la relación de probabilidades, penalizando la generación de respuestas no deseadas durante el ajuste fino. La función objetivo de ORPO integra la pérdida convencional de SFT con una pérdida de relación relativa ponderada, permitiendo que el modelo adapte su generación de texto a estilos preferidos sin un proceso de alineación de preferencias separado. Empíricamente, ORPO ha demostrado su eficacia al superar a modelos de lenguaje de última generación con más parámetros, mostrando un mejor rendimiento en tareas de seguimiento de instrucciones y generación de texto (Hong et al., 2024).

QLoRA fue escogida debido a su capacidad para permitir la cuantización de modelos grandes en solo 4 bits, lo que reduce el uso de memoria mientras mantiene un rendimiento alto. Esta técnica, implementada en la plataforma Unsloth AI, se encarga de optimizar el proceso de fine-tuning al aplicar la cuantización NormalFloat (NF4) y la doble cuantización, permitiendo un entrenamiento eficiente en hardware limitado. Además, Unsloth AI facilita la integración con bibliotecas como `transformers` y `bitsandbytes`, optimizando aún más el proceso de despliegue de modelos.

2.6.3 Uso del Fine-Tuning para la Mejora del Razonamiento Matemático de los LLMs

La técnica del fine-tuning permite enfocar el conocimiento general del modelo hacia tareas específicas, por ende tiene la capacidad para adaptarlos a tareas dentro del área de las matemáticas (Hu et al., 2024). Actualmente, muchos LLMs presentan dificultades en la resolución de problemas matemáticos complejos y no siempre tienen una comprensión profunda de los conceptos matemáticos, pueden tener dificultades para manejar símbolos y ecuaciones (Liu et al., 2023b). El fine-tuning puede abordar estas limitaciones enfocando el conocimiento general del modelo hacia tareas matemáticas y entrenando al modelo con conjuntos de datos específicos de problemas matemáticos.

En este contexto, el fine-tuning se presenta como una herramienta poderosa para entrenar a los LLMs para abordar problemas matemáticos de diversa complejidad, el modelo puede aprender a analizar enunciados, identificar las relaciones matemáticas involucradas y aplicar las estrategias correctas para llegar a una solución precisa. Puede usarse para entrenar a un LLM en la demostración formal de teoremas matemáticos, el modelo puede aprender a manipular símbolos y reglas lógicas, siguiendo pasos precisos para construir una demostración válida. El fine-tuning puede convertir a un LLM en una herramienta para explicar conceptos matemáticos de forma clara y comprensible. El modelo puede traducir información matemática compleja en lenguaje natural, adaptando su explicación al nivel de conocimiento del usuario.

Por ejemplo, Liu et al. (2023b) proporciona una visión de cómo el fine-tuning pueden mejorar el rendimiento de los LLMs en tareas complejas como la resolución de problemas matemáticos. Se propone tres estrategias para mejorar la capacidad de los LLMs en la resolución de problemas matemáticos: ajuste fino de soluciones, re-clasificación de grupos de soluciones y ajuste fino secuencial multitarea.

- **Ajuste Fino de Soluciones:** Esta estrategia implica ajustar un LLM utilizando soluciones específicas a problemas matemáticos como datos de entrenamiento. En lugar de entrenar desde cero, el modelo se ajusta a las características de las soluciones matemáticas existentes. Seleccionar soluciones de alta calidad y estilo coherente es crucial para el éxito del ajuste fino. Estas

soluciones paso a paso se utilizan para entrenar el modelo y mejorar su capacidad para resolver problemas matemáticos.

- **Re-Clasificación de Grupos de Soluciones:** En esta estrategia, se agrupan soluciones matemáticas similares en categorías o grupos. El modelo se entrena para clasificar estas categorías de soluciones. Esto permite al modelo generalizar mejor y comprender diferentes enfoques para resolver problemas similares.

- **Ajuste Fino Secuencial multitarea:** Esta estrategia combina la tarea de evaluación de soluciones con la tarea de generación de soluciones. El modelo se entrena secuencialmente en ambas tareas, lo que permite que la información de evaluación beneficie la generación de soluciones y viceversa. La integración eficiente de estas dos tareas mejora el rendimiento general del modelo en la resolución de problemas matemáticos.

Liu y sus colegas presentan su estudio utilizando modelos PaLM 2, destacando que la calidad y el estilo de las soluciones paso a paso utilizadas para el fine-tuning tienen un impacto significativo en el rendimiento del modelo. En relación con la mejora del rendimiento, se diseñó una forma de fine-tuning que logra aproximadamente un 58.8 % de precisión en el conjunto de datos MATH (conjunto de datos de problemas matemáticos de competencia con su solución completa paso a paso) con modelos PaLM 2-L, lo que representa una mejora del 11.2 % en precisión en comparación con el rendimiento previo al fine-tuning.

Con relación al mismo contexto, Wang et al. (2023) sugiere un método para mejorar la capacidad de razonamiento matemático de los LLMs de código abierto, permitiéndoles utilizar código para modelar y derivar ecuaciones matemáticas. Propone una técnica de fine-tuning supervisado y un enfoque de inferencia personalizado que permite a los modelos generar soluciones basadas en código para resolver problemas matemáticos complejos. Propone la creación de MathCoder, un LLM de código abierto diseñado para el razonamiento matemático, cerrando la brecha entre la comprensión del lenguaje natural y la resolución de problemas computacionales. MathCoder incorpora la construcción de conjuntos de datos matemáticos siguiendo instrucciones. Al utilizar los conjuntos de datos GSM8K (conjunto de datos de problemas matemáticos de escuela primaria) y MATH como datos de entrenamiento, aprovecha GPT-4 para generar problemas que abarcan el razonamiento, la generación de código y la

ejecución de programas. Además, se propone un método de interpretación de problemas para crear problemas de nivel intermedio. Lo destacable de este artículo es que se introduce un enfoque de fine-tuning supervisado personalizado, en el que la pérdida de entrenamiento solo se aplica al lenguaje natural y al código. El estudio demuestra que MathCoder logra un rendimiento mejorado en cinco conjuntos de datos matemáticos entre los LLM de código abierto, con puntuaciones del 83,9 % en el conjunto de datos GSM8K y del 45,2 % en el conjunto de datos MATH. Vale la pena señalar que MathCoder supera a 9 modelos de código cerrado como GPT-3.5 y PaLM 2 en los conjuntos de datos GSM8K y MATH.

Sin embargo, el trabajo tiene ciertas limitaciones. En primer lugar, se depende de GPT-4 para la generación de datos, las capacidades de MathCoder están inherentemente limitadas por las capacidades de este modelo y son incapaces de resolver problemas de demostración de teoremas. Además, como una serie de modelos unimodales, MathCoder aún enfrenta desafíos en la resolución de problemas de geometría complejos. Para abordar estas limitaciones, la presente Memoria de Título explorará el uso del fine-tuning en modelos de lenguaje grande, específicamente en el contexto de la enseñanza de matemáticas y física. Se investigará cómo el ajuste fino de modelos pre-entrenados puede mejorar su desempeño en la resolución de problemas complejos que involucran razonamiento matemático, considerando tanto la precisión de las respuestas como la calidad de las explicaciones proporcionadas. Esta investigación no solo pretende superar las limitaciones actuales, sino también contribuir al desarrollo de herramientas educativas más efectivas y accesibles, capaces de apoyar el aprendizaje en contextos académicos diversos.

Capítulo 3

Metodología

En esta sección se explica la metodología utilizada para llevar a cabo la investigación sobre el fine-tuning en modelos de lenguaje grande aplicados a la educación en física y matemáticas. Se detallan los procedimientos de cuantización de modelos, la preparación del entorno de desarrollo, la selección de modelos pre-entrenados, la recopilación y preparación de datos, la elección de técnicas de fine-tuning y la evaluación del rendimiento de los modelos entrenados. Este enfoque metodológico busca optimizar el desempeño de los LLMs en la resolución de problemas matemáticos y físicos, proporcionando una base sólida para su implementación en contextos educativos.

3.1 Cuantización de Modelos

Los LLMs han mostrado una notable habilidad en tareas como la generación de texto. Sin embargo, su uso doméstico presenta un desafío significativo debido a la enorme cantidad de recursos computacionales que requieren. El entrenamiento de estos modelos requiere acceso a infraestructuras de computación de alta gama, a menudo en la forma de clústeres de GPU (unidades de procesamiento gráfico) o TPU (unidades de procesamiento tensorial). Estos recursos no solo son costosos en términos de adquisición y mantenimiento, sino que también consumen cantidades significativas de energía eléctrica, lo que se traduce en altos costos operativos. La inferencia con LLMs, es decir, el uso de estos modelos para generar predicciones o producir texto en tiempo real, también es intensiva en recursos. La gran cantidad de parámetros que componen estos modelos (a menudo en el rango de miles de millones) requiere memoria considerable y poder de procesamiento.

La cuantización de modelos es una técnica para abordar estos desafíos. Consiste en reducir la precisión de los parámetros del modelo durante el entrenamiento y las inferencias, lo que puede

disminuir significativamente el tamaño del modelo y los requisitos de computación, sin una pérdida sustancial de rendimiento.

Para el uso de modelos cuantizados se hará uso de la plataforma Hugging Face, una plataforma que facilita el acceso y la utilización de LLMs (Wolf et al., 2020). Hugging Face alberga una colección vasta de modelos pre-entrenados y datasets. En esta plataforma se encuentran alojados modelos de código abierto de grandes empresas, como Meta AI y Mistral AI, así como versiones cuantizadas de estos modelos, que han sido optimizadas por la comunidad para un uso más eficiente.

3.2 Preparación del Entorno

Para efectos de esta investigación y las limitaciones de poder computacional, es necesario el uso de LLMs cuantizados. Se hará uso del servicio de Unsloth AI para el entrenamiento de LLMs, servicio que facilita y optimiza el proceso de fine-tuning de modelos de lenguaje pre-entrenados, el cual cuenta con su repositorio en GitHub. La plataforma se centra en mejorar la eficiencia del proceso, haciendo que el fine-tuning sea hasta 2.7 veces más rápido y reduzca el uso de memoria VRAM hasta en un 74 %. Unsloth AI soporta modelos avanzados como Llama y Mistral y se integra perfectamente con el ecosistema de Hugging Face, para el uso de modelos y datasets alojados en la plataforma. Utilizando técnicas avanzadas como QLoRA para el entrenamiento de LLMs.

La preparación del entorno para el proceso de fine-tuning se llevó a cabo en Google Colab, una plataforma que ofrece recursos de GPU (gratuitos y de pago), esenciales para manejar los requisitos computacionales de los LLMs. Se inicia creando un nuevo notebook en Google Colab. Se configuró el entorno para utilizar una GPU, esto es crucial para aprovechar la capacidad de procesamiento y acelerar el entrenamiento del modelo en la plataforma. Para trabajar con Unsloth AI y realizar el fine-tuning de modelos, se instalaron las bibliotecas necesarias: `transformers`, `datasets`, `accelerate` y `unsloth`, como se muestra en el Anexo 1. Estas bibliotecas proporcionan herramientas para manejar modelos de lenguaje, conjuntos de datos y optimización del entrenamiento.

3.3 Selección de Modelos Pre-Entrenados

En el contexto de esta investigación, la selección de modelos de lenguaje pre-entrenados constituye una fase crítica para asegurar no solo el éxito del proceso de fine-tuning, sino también la eficiencia operativa y el rendimiento del modelo ajustado. La elección adecuada de estos modelos es fundamental, ya que permite aprovechar al máximo las capacidades intrínsecas de los modelos base, optimizando su adaptación a tareas específicas en el ámbito educativo. Para la selección de los modelos utilizados en este estudio, se consideraron varios criterios importantes, cada uno de los cuales contribuye a garantizar que los modelos elegidos sean los más adecuados para cumplir con los objetivos de la investigación.

En primer lugar, se evaluó el rendimiento de los modelos en tareas similares, particularmente en contextos educativos y académicos relacionados con problemas de física y matemáticas. Este criterio asegura que los modelos tengan una base sólida de rendimiento en las áreas específicas de interés. Además, se consideró la capacidad de los modelos para ser ajustados mediante técnicas de fine-tuning, lo cual es esencial para mejorar su desempeño en tareas específicas. Este criterio incluye la facilidad de implementación y la compatibilidad con las técnicas de fine-tuning disponibles, como QLoRA y SFT, utilizadas en esta investigación.Otro aspecto crucial fue la disponibilidad de recursos pre-entrenados. Los modelos seleccionados necesitaban contar con una base robusta de entrenamiento previo en grandes cantidades de datos, lo que proporciona un punto de partida fuerte para el fine-tuning y reduce significativamente el tiempo y los recursos necesarios para el ajuste fino. Además, se realizó una comparación de modelos de código abierto para evaluar su accesibilidad y eficacia en contextos educativos. La elección de modelos de código abierto permite una mayor flexibilidad y adaptabilidad en la investigación, facilitando el acceso y la colaboración. Finalmente, se debe considerar la disponibilidad general de los modelos, incluyendo su accesibilidad para uso académico y la calidad de su documentación. La facilidad de acceso y la existencia de una documentación detallada son esenciales para asegurar que los modelos puedan ser utilizados y adaptados de manera eficiente durante el proceso de investigación.

Modelo	Descripción	Desarrollador	Parámetros
`babbage-002`	Basado en la arquitectura GPT (Generative Pre-trained Transformer). Diseñado para ser eficiente en términos de recursos computacionales mientras mantiene un buen rendimiento en tareas de NLP.	OpenAI	1.3 MM
`davinci-002`	Versión avanzada de la serie GPT-3. Conocido por su capacidad de generar textos coherentes y de alta calidad. Adecuado para tareas complejas de NLP, incluyendo comprensión de lenguaje, generación de contenido, etc.	OpenAI	175 MM
`Mistral-7B-Instruct-v0.3`	Enfocado en la eficiencia y el rendimiento en tareas específicas del NLP. Ideal para aplicaciones donde se requiere un procesamiento rápido y preciso del lenguaje.	Mistral AI	7 MM
`Meta-Llama-3-8B-Instruct`	Basado en la arquitectura de transformadores LLaMA, diseñado para ser accesible y eficiente, proporcionando un buen equilibrio entre rendimiento y requisitos computacionales. Adecuado para una variedad de aplicaciones de NLP.	Meta AI	8 MM

Tabla 3.1: Detalle de modelos seleccionados.

Para esta investigación, se seleccionaron versiones cuantizadas de los modelos Llama y Mistral, proporcionadas por Unsloth AI en la plataforma Hugging Face, `llama-3-8b-Instruct-bnb-4bit` y `mistral-7b-instruct-v0.3-bnb-4bit` respectivamente. Estos modelos cuantizados han sido

optimizados para un uso más eficiente, facilitando su implementación en entornos con limitaciones de recursos computacionales. Además, debido a su número de parámetros de entrenamiento, sus resultados son comparables.

3.4 Fuente de Datos

La recopilación de datos es una parte crucial del proceso de evaluación y entrenamiento de modelos de lenguaje grande. Para entrenar y evaluar estos modelos de manera efectiva, es necesario disponer de conjuntos de datos bien estructurados que incluyan preguntas y respuestas, especialmente en el ámbito de los problemas matemáticos. Estos conjuntos de datos deben estar divididos en secciones de entrenamiento y prueba para permitir tanto el fine-tuning como la evaluación del rendimiento de los modelos.

La selección de conjuntos de datos adecuados es fundamental para el éxito de cualquier modelo de lenguaje grande en tareas específicas, especialmente en el ámbito educativo. La calidad de los datos utilizados para el entrenamiento y evaluación del modelo determinan en gran medida su rendimiento para generalizar a nuevos problemas. Datos bien estructurados y representativos del dominio específico aseguran que el modelo pueda aprender patrones significativos y contextualmente apropiados. En el contexto de problemas matemáticos, por ejemplo, es crucial disponer de conjuntos de datos que no solo contengan problemas variados y de diferentes niveles de dificultad, sino que también proporcionen soluciones detalladas y explicaciones paso a paso. Esto no solo facilita un aprendizaje más profundo y efectivo del modelo, sino que también mejora su capacidad para proporcionar respuestas precisas y útiles en aplicaciones del mundo real.

Una fuente de datos utilizada en este estudio es GSM8K. Este conjunto de datos es ampliamente reconocido por su calidad y relevancia en la evaluación de capacidades de razonamiento matemático en modelos de lenguaje.

GSM8K

Conjunto de datos diseñado para evaluar y mejorar la capacidad de los modelos de lenguaje en resolver problemas matemáticos de nivel escolar. Este conjunto de datos incluye problemas matemáticos que cubren diversas áreas y niveles de dificultad (Cobbe et al., 2021). GSM8K contiene problemas típicos de matemáticas de primaria y secundaria. Es utilizado para el entrenamiento y evaluación de modelos en tareas de razonamiento matemático básico. Cada problema viene acompañado de una solución detallada, lo que facilita el entrenamiento supervisado de los modelos. GSM8K se encuentra alojado en la plataforma Hugging Face y el Anexo 2 presenta una vista parcial del dataset.

3.5 Elección de la Técnica de Fine-Tuning

SFT es el proceso de ajustar un modelo pre-entrenado utilizando un conjunto de datos etiquetado, donde las entradas están asociadas con las salidas deseadas. En este caso, el dataset GSM8K, que contiene problemas matemáticos de nivel escolar y sus respectivas respuestas. El entrenamiento supervisado garantiza que los modelos aprendan a mapear las entradas a las salidas correctas basándose en los ejemplos proporcionados.

Para llevar a cabo el fine-tuning de los modelos seleccionados (Llama y Mistral), se optó por la técnica QLoRA implementada en la plataforma Unsloth AI. Este método se seleccionó por su eficiencia y efectividad en el ajuste de modelos de lenguaje grande como Llama y Mistral, optimizando el uso de recursos computacionales. Esta elección permite un fine-tuning supervisado eficiente utilizando el dataset GSM8K, que contiene problemas matemáticos escolares y sus respectivas respuestas, asegurando que los modelos aprendan a mapear correctamente las entradas a las salidas deseadas basándose en los ejemplos proporcionados.

3.6 Preparación de Datos

El preprocesamiento de datos es un paso crítico para preparar los conjuntos de datos para el entrenamiento y la evaluación de los modelos. Este proceso incluye varias etapas fundamentales, entre las que destacan la limpieza de datos y la transformación de datos. Estos pasos son esenciales para garantizar que los datos sean consistentes y adecuados para los algoritmos de aprendizaje automático, lo que a su vez mejora la precisión y la eficiencia del modelo durante las fases de entrenamiento y evaluación.

3.6.1 Limpieza de Datos

Los procedimientos de limpieza de datos incluyen la eliminación de duplicados, corrección de errores y filtrado de contenido irrelevante o inapropiado. Esto asegura que los datos utilizados para el entrenamiento sean de alta calidad y relevantes para las tareas específicas. Entre las tareas, se destacan:

- **Eliminación de Duplicados:** Asegura que no haya entradas repetidas que puedan sesgar el entrenamiento del modelo.
- **Corrección de Errores:** Identificación y corrección de errores tipográficos y gramaticales en los datos.
- **Filtrado de Contenido Irrelevante:** Eliminación de datos que no sean pertinentes para la tarea de entrenamiento específica.

3.6.2 Transformación de Datos

Cada framework y modelo puede requerir un formato específico para realizar el fine-tuning. Por ejemplo, la biblioteca `transformers` de Hugging Face requiere que los datos se estructuren en un archivo `JSON`, donde cada entrada debe estar claramente etiquetada con la pregunta y la respuesta correspondiente.

3.6.3 Preparación del Dataset GSM8K

Para el fine-tuning de los modelos Llama y Mistral, se utilizó el dataset GSM8K, un conjunto de datos de problemas matemáticos de nivel escolar. Este dataset es ideal para evaluar la capacidad de los modelos en tareas de razonamiento y resolución de problemas.

Adaptación del Dataset

El dataset GSM8K fue adaptado al formato requerido por los frameworks de Hugging Face y Unsloth AI. En particular, se utilizó el formato ChatML para estructurar los datos en un estilo conversacional, adecuado para el fine-tuning de modelos de lenguaje. El dataset fue adaptado mediante un script en Python y fue alojado a la plataforma Hugging Face para su posterior uso.

El formato ShareGPT es un formato estructurado que facilita la representación de diálogos entre el modelo y el usuario. En este formato, cada entrada del dataset se estructura como una lista de diccionarios, donde cada diccionario representa un turno en la conversación. A continuación se muestra un ejemplo de su estructura:

```
[
 [{'from': 'human', 'value': 'Hello, how are you?'},
  {'from': 'gpt', 'value': 'I am fine, thank you. And you?'},
  {'from': 'human', 'value': 'I am fine too. What is 2+2?'}],
 [{'from': 'human', 'value': 'What is your name?'},
  {'from': 'gpt', 'value': 'I am a language model.'},
  {'from': 'human', 'value': 'Nice to meet you.'}]
]
```

Luego de aplicar de formato ShareGPT, se utilizó la función `get_chat_template` proporcionada por Unsloth AI para garantizar que los datos estuvieran listos para el entrenamiento, adaptando el

dataset a las diferentes plantillas de prompts que requieren los distintos modelos soportados Unsloth AI, el Anexo 3 muestra el uso de esta función.

Estos procedimientos aseguran que los conjuntos de datos utilizados para el entrenamiento y la evaluación sean de alta calidad, relevantes y seguros, permitiendo que los modelos de lenguaje grande sean entrenados y evaluados de manera efectiva.

3.7 Despliegue e Inferencias de Modelos

3.7.1 Infraestructura Utilizada

La infraestructura utilizada para el entrenamiento y despliegue de los LLMs se basa principalmente servicios en la nube para optimizar el rendimiento y minimizar los costos computacionales. A continuación, se detallan los componentes específicos utilizados para los modelos de código abierto y de código cerrado.

Modelos de Código Abierto

Para los modelos de código abierto, como Mistral y Llama, se utilizó el entorno de Hugging Face, que proporciona herramientas robustas para el manejo de modelos de lenguaje grande.

- **Hardware:** Se utilizó Google Colab para aprovechar las GPU gratuitas proporcionadas por la plataforma (NVIDIA T4), lo cual es esencial para el entrenamiento y despliegue de modelos de gran tamaño.
- **Software:** La biblioteca `transformers` de Hugging Face fue la principal herramienta utilizada. Esta biblioteca facilita la carga e inferencia de modelos de lenguaje grande.
- **Cuantización:** Para mejorar la eficiencia y reducir el uso de memoria, se aplicó la técnica de cuantización a los modelos para las inferencias.

Modelos de Código Cerrado

Para los modelos de código cerrado, como `babbage-002` y `davinci-002` de OpenAI, se utilizó la API de OpenAI, que proporciona una forma accesible de interactuar con estos modelos avanzados.

- **Hardware:** Los modelos de código cerrado no requieren infraestructura local ya que todo el procesamiento se realiza en los servidores de OpenAI.
- **Software:** La interacción con los modelos se realizó mediante la API de OpenAI. Esto permitió enviar solicitudes de inferencia y recibir respuestas generadas por los modelos.
- **Costos:** El uso de la API de OpenAI implica costos basados en el número de tokens procesados. Cada solicitud de inferencia y respuesta genera un costo proporcional a la cantidad de texto manejado.

3.7.2 Prompt Template para Modelos de Código Abierto

Para los modelos de código abierto como Llama y Mistral, es crucial diseñar plantillas de prompts que maximicen su rendimiento en tareas específicas. Los prompts son fundamentales para guiar al modelo en la generación de respuestas precisas y relevantes. A continuación, se explica detalladamente cómo se construyen y utilizan estas plantillas de prompts.

Modelo Llama

El modelo `Meta-Llama-3-8B-Instruct` es desarrollado por Meta y está diseñado para ser eficiente y accesible, permitiendo su uso en una variedad de aplicaciones de procesamiento del lenguaje natural.

- **Prompt Template:** Para Llama, la plantilla de prompts se diseña para proporcionar contexto claro y específico, esto ayuda al modelo a entender la tarea y generar respuestas coherentes.

- **Estructura:** La estructura típica de un prompt para Llama incluye un contexto que enmarca la situación o el tema, seguido de una pregunta o instrucción clara. Esto permite al modelo generar respuestas que son directamente relevantes y contextualmente apropiadas.
- **Uso de Instrucciones Claras:** Incluir instrucciones claras y específicas ayuda a Llama a enfocarse en la tarea requerida, mejorando la precisión y relevancia de las respuestas generadas.
- **Ejemplo de Prompt:**

```
<|begin_of_text|>
<|start_header_id|>system<|end_header_id|>
 You are a helpful AI assistant<|eot_id|>
<|start_header_id|>user<|end_header_id|>
 What is France's capital?<|eot_id|>
<|start_header_id|>assistant<|end_header_id|>
```

Modelo Mistral

El modelo `Mistral-7B-Instruct-v0.3` es desarrollado por Mistral AI y se destaca por su capacidad para adaptarse rápidamente a nuevas tareas, proporcionando un rendimiento eficiente en diversas aplicaciones de procesamiento del lenguaje natural.

- **Prompt Template:** La plantilla para los modelos Mistral está diseñada para aprovechar su capacidad de seguir instrucciones, lo que es útil en aplicaciones de generación de contenido.
- **Estructura:** Similar a Llama, las plantillas de prompts para Mistral incluyen un tema claro y una pregunta específica. Este enfoque ayuda al modelo a generar respuestas precisas y útiles.
- **Instrucciones Detalladas:** Proporcionar detalles específicos en el prompt asegura que el modelo entienda completamente la tarea y genere una respuesta adecuada. Esto incluye especificar el formato de la respuesta esperada.

- **Ejemplo de Prompt:**

```
<|user|>
I am going to Paris, what should I see?<|end|>
<|assistant|>
Respuesta generada."<|end|>
<|user|>
What is so great about #1?<|end|>
<|assistant|>
```

El diseño cuidadoso de los prompts es crucial para maximizar el rendimiento de los modelos de código abierto como Llama y Mistral. Estas plantillas no solo guían al modelo en la generación de respuestas, sino que también aseguran que las respuestas sean coherentes, relevantes y útiles en el contexto educativo y académico. Es importante señalar que, el objetivo de esta investigación es únicamente entender cómo el fine-tuning puede mejorar la calidad de las respuestas matemáticas correctas. Por esta misma razón, no se explora la ingeniería de prompts para mejorar las respuestas.

3.7.3 Proceso de Despliegue

El proceso de despliegue de los modelos en el entorno seleccionado se realizó en varios pasos, utilizando diversas herramientas y servicios para asegurar un rendimiento óptimo y una implementación eficiente.

Modelos de Código Abierto

Para suplir la falta de recursos locales y aprovechar las capacidades de GPU, se utilizó Google Colab. Esta plataforma proporciona acceso a recursos de computación potentes sin costo, lo cual es ideal para el entrenamiento y despliegue de modelos de lenguaje grande.

- **Configuración del Entorno:** Se configuró el entorno de Google Colab con las bibliotecas necesarias, incluyendo `transformers` de Hugging Face y otros paquetes de Python.
- **Carga de Modelos:** Los modelos se cargaron desde la biblioteca de Hugging Face, aprovechando las capacidades de almacenamiento y gestión de modelos de esta plataforma.
- **Cuantización:** Se aplicaron técnicas de cuantización a los modelos para mejorar la eficiencia y reducir el tiempo de inferencia.

Lo anterior se trabajó en notebooks de Jupyter, los cuales cuentan con su repositorio en GitHub, estos scripts se pueden utilizar localmente, pero por la necesidad de recursos de GPU se recomienda trabajarlos en Google Colab como se mencionó anteriormente.

Para desplegar modelos completos no cuantizados en la plataforma Hugging Face, es esencial considerar los costos de las GPU, ya que estos modelos requieren recursos significativos debido a su tamaño y complejidad. Hugging Face ofrece diversas opciones de hardware, cada una con diferentes capacidades y precios. Estas opciones permiten seleccionar la configuración adecuada según las necesidades específicas del modelo y el presupuesto disponible, facilitando el despliegue eficiente de modelos avanzados, a continuación se muestra la lista de precios entregada por Hugging Face por el uso de GPUs:

Arquitectura	GPUs	Memoria	Tarifa por hora
NVIDIA T4	1	14GB	$0.50
	4	56GB	$3.00
NVIDIA L4	1	24GB	$0.80
	4	96GB	$3.80
NVIDIA A10G	1	24GB	$1.00
	4	96GB	$5.00
NVIDIA A100	1	80GB	$4.00
	2	160GB	$8.00
	4	320GB	$16.00
	8	640GB	$32.00
NVIDIA T4	1	16GB	$0.50
NVIDIA L4	1	24GB	$1.00
	4	96GB	$5.00
NVIDIA A100	1	80GB	$6.00
	2	160GB	$12.00
	4	320GB	$24.00
	8	640GB	$48.00
NVIDIA H100	1	80GB	$12.50
	2	160GB	$25.00
	4	320GB	$50.00
	8	640GB	$100.00

Tabla 3.2: Precios por uso de GPUs de Hugging Face.

Modelos de Código Cerrado

Para los modelos de código cerrado de OpenAI, se siguieron los siguientes pasos:

- **Registro y Configuración:** Se creó una cuenta en OpenAI y se configuraron las credenciales de API necesarias para acceder a los modelos.

- **Envío de Solicitudes:** Se utilizaron scripts en Python para enviar solicitudes a la API de OpenAI, especificando las instrucciones y contextos necesarios para la inferencia.

3.7.4 Entrenamiento de Modelos

Se entrenaron los modelos `babbage-002` y `davinci-002` mediante la API, lo cual permitió aprovechar las capacidades avanzadas de procesamiento de lenguaje natural disponibles en estos modelos preentrenados. Posteriormente, se entrenaron los modelos Mistral y Llama en sus versiones Instruct (preparadas para el chat) durante 2 epochs, con el objetivo de ajustar y mejorar su rendimiento en tareas específicas.

El parámetro epoch se refiere a una iteración completa sobre todo el conjunto de datos de entrenamiento. Durante cada época, el modelo ajusta sus pesos y sesgos en función del error observado en los datos de entrenamiento. En el contexto del fine-tuning, un número adecuado de épocas es crucial para que el modelo aprenda patrones específicos sin sobre ajustarse (overfitting, en inglés) a los datos de entrenamiento.

Para evaluar la influencia del tamaño del dataset en el rendimiento del modelo, se utilizó el conjunto de datos GSM8K. Se realizaron experimentos de entrenamiento utilizando el 100 % de los datos disponibles, así como con un 50 %, permitiendo observar cómo la variabilidad en el tamaño del dataset afecta el rendimiento final de los modelos. Estos experimentos ayudan a comprender mejor la relación entre la cantidad de datos de entrenamiento y la capacidad del modelo para generalizar y desempeñarse de manera efectiva en diversas tareas.

Los experimentos de fine-tuning para los modelos de código abierto se trabajaron en notebooks de Jupyter, los cuales cuentan con su repositorio en GitHub.

3.7.5 Despliegue de Modelos Entrenados

Se presenta un enfoque para desplegar modelos entrenados utilizando el método QLoRA. QLoRA es una técnica innovadora de fine-tuning que permite la cuantización de modelos grandes en solo 4 bits,

lo cual reduce significativamente el uso de memoria mientras se mantiene un rendimiento alto. Esto se logra a través de dos técnicas principales, la cuantización NF4 y la doble cuantización. NF4 optimiza la compresión de datos normalmente distribuidos con mínima pérdida de información, mientras que la doble cuantización reduce aún más el tamaño de los parámetros, permitiendo un entrenamiento eficiente en hardware limitado.

Para desplegar estos modelos, se utilizó un script de Python que integra las librerías transformers y bitsandbytes. Estas librerías facilitan el despliegue de modelos cuantizados. El script comienza por cargar el modelo pre-entrenado y configurar los parámetros de cuantización.

Primero, se debe asegurar que el modelo pre-entrenado original esté correctamente cargado y disponible. Este modelo servirá como la base sobre la cual se aplicarán las modificaciones de QLoRa. Posteriormente, se deben integrar las capas adaptativas de QLoRA en el modelo pre-entrenado. Esto implica reemplazar ciertas capas, como las capas de atención, con sus versiones modificadas por QLoRA. Una vez que las capas de QLoRA se han integrado en el modelo, se deben cargar los parámetros entrenados en estas capas. Estos parámetros son los que se ajustaron durante la fase de entrenamiento.

Con el modelo pre-entrenado y las modificaciones de QLoRA integradas y cargadas, el modelo está listo para realizar inferencias. Durante la inferencia, las actualizaciones de QLoRA se aplican automáticamente, lo que permite al modelo generar predicciones utilizando tanto los pesos originales como las actualizaciones específicas de QLoRA.

Los despliegues para los modelos entrenados de código abierto se trabajaron en notebooks de Jupyter, los cuales cuentan con se cuentan repositorio en GitHub, en recomendable trabajarlos en Google Colab.

3.8 Evaluación

La evaluación del rendimiento de los modelos es un paso crucial para determinar la efectividad del proceso de fine-tuning. En este estudio, se llevó a cabo una comparación exhaustiva entre los modelos

base (pre-entrenados) y los modelos ajustados (mediante fine-tuning) para medir las mejoras en su desempeño en tareas específicas de matemáticas y física.

Para evaluar el rendimiento de los modelos, se emplearon las siguientes métricas estándar en el procesamiento del lenguaje natural:

- **Precisión (Accuracy):** La proporción de respuestas correctas generadas por el modelo en comparación con las respuestas esperadas.

El proceso de evaluación se realizó en varias etapas, utilizando conjuntos de datos de prueba que no se emplearon durante el entrenamiento. Las etapas del proceso de evaluación incluyen:

- **Inferencia:** Se realizaron inferencias tanto con los modelos base como con los modelos ajustados. Durante esta etapa, los modelos generaron respuestas para primeras 100 preguntas del subconjunto de evaluación en el conjunto de datos GSM8K.
- **Comparación de Resultados:** Las respuestas generadas por los modelos se compararon con las respuestas correctas utilizando las métricas de evaluación mencionadas anteriormente. Esta comparación permitió medir la precisión y las mejoras obtenidas con el fine-tuning.

Capítulo 4

Resultados

4.1 Evaluación

La evaluación del rendimiento de los modelos es un paso crucial para determinar la efectividad del proceso de fine-tuning. En este estudio, se llevó a cabo una comparación exhaustiva entre los modelos base (pre-entrenados) y los modelos entrenados (ajustados mediante fine-tuning) para medir las mejoras en su desempeño en tareas específicas.

Los resultados obtenidos de los modelos base y los modelos entrenados se resumen en la Tabla 4.1, en la cual se muestran los porcentajes de acierto en el sub-conjunto de ejemplos de prueba en el conjunto de datos GSM8K (los cuales no se utilizaron para el entrenamiento), en la tabla se muestran los resultados para los modelos base escogidos en esta investigación, además de sus versiones entrenadas con el 50 % y 100 % del sub-conjunto mencionado, 3736 y 7473 ejemplos respectivamente.

Modelo	**Modelo base**	**FT 50 % GSM8K**	**FT 100 % GSM8K**
`babbage-002`	0 %	3 %	12 %
`davinci-002`	5 %	13 %	– %
`Mistral-7B-Instruct-v0.3`	33 %	36 %	52 %
`Meta-Llama-3-8B-Instruct`	45 %	64 %	70 %

Tabla 4.1: Resultados de evaluación de modelos en GSM8K.

El modelo `babbage-002` mostró una mejora significativa después del fine-tuning. Inicialmente, el modelo base no tuvo ningún acierto, pero con el entrenamiento en GSM8K, el porcentaje de acierto aumentó al 3 % y luego al 12 % con un dataset más grande (7473 ejemplos). Con aproximadamente 1.3 mil millones de parámetros, esta baja cantidad de parámetros y la falta de fine-tuning inicial explican

el bajo rendimiento del modelo base. El aumento del tamaño del dataset de entrenamiento mejora considerablemente su rendimiento.

El modelo `davinci-002` también mostró mejoras notables con el fine-tuning. El modelo base tenía un 5 % de acierto, que aumentó al 13 % con 3736 ejemplos. Modelo con aproximadamente 175 mil millones de parámetros, debido al mayor número de parámetros respecto al modelo `babbage-002`, `davinci-002` demostró ser más efectivo inicialmente. El fine-tuning mejoró significativamente su rendimiento, indicando una mayor capacidad de adaptación del modelo.

El modelo `Mistral-7B-Instruct-v0.3` mostró un rendimiento superior en todas las etapas de la evaluación con respecto a los modelos de OpenAI. El modelo base tuvo un 33 % de acierto, que aumentó al 36 % con 3736 ejemplos y al 52 % con 7473 ejemplos. Este modelo tiene aproximadamente 7 mil millones de parámetros y demostró ser altamente eficaz desde el principio debido a su diseño orientado a seguir instrucciones (Instruct). La técnica de fine-tuning y el tamaño del dataset contribuyeron a mejoras adicionales en su rendimiento.

El modelo `Meta-Llama-3-8B-Instruct` tuvo el mejor rendimiento en todas las evaluaciones. El modelo base tenía un 45 % de acierto, que aumentó al 64 % con 3736 ejemplos y al 70 % con 7473 ejemplos. Modelo con aproximadamente 8 mil millones de parámetros, mayor número de parámetros que `Mistral-7B-Instruct-v0.3`, lo que le permite captar mejor los patrones en los datos de entrenamiento. El fine-tuning y el aumento del tamaño del dataset mejoraron significativamente su rendimiento.

Al evaluar las distintas inferencias, se encontraron varios casos recurrentes en las respuestas proporcionadas por los modelos, base y entrenados. Uno de los casos más recurrente fue aquel que se presenta, a manera de ejemplo, en las respuestas entregadas por el modelo `Mistral-7B-Instruct-v0.3` para la pregunta 1 del subconjunto de prueba del dataset GSM8K (Anexo 4), en donde se compararon las respuestas con la respuesta esperada (Anexo 5), del dataset mencionado. Se pudo observar que el modelo base lograba hacer el desarrollo paso a paso y llegar a la respuesta correcta (Anexo 6), al igual que el modelo entrenado con el dataset completo (Anexo 7), que además respetaba el formato y sintaxis del dataset GSM8K. Sin embargo, el modelo entrenado con el 50 % del dataset, aunque respetaba el formato y trataba de seguir un desarrollo lógico, proporcionaba una respuesta incorrecta (Anexo 8).

Otro caso común fue aquel que se presenta, a manera de ejemplo, en las respuestas entregadas por el modelo `Meta-Llama-3-8B-Instruct` para la pregunta 74 del subconjunto de prueba del dataset GSM8K (Anexo 9), también se comparó con la respuesta del dataset (Anexo 10). En este caso el modelo base no lograba llegar a la respuesta correcta (Anexo 11). El modelo entrenado con el 50 % del dataset también presentaba errores (Anexo 12), aunque mostraba una mejora en cuanto al formato y la sintaxis de las respuestas en comparación con el dataset de entrenamiento. Por último, el modelo entrenado con el 100 % del dataset no solo respetaba el formato de respuesta (Anexo 13), sino que también llegaba a la respuesta correcta.

El tamaño del dataset de entrenamiento desempeñó un papel crucial en la mejora del rendimiento de los modelos. Los datos de alta calidad y en grandes volúmenes son esenciales para entrenar modelos de lenguaje grande de manera efectiva. El aumento del tamaño del dataset, como se observó con el conjunto de datos GSM8K, que pasó de 3736 a 7473 ejemplos, resultó en mejoras sustanciales en el rendimiento de los modelos. Un mayor volumen de datos proporciona una mayor diversidad de ejemplos, lo que ayuda al modelo a aprender una variedad más amplia de patrones y contextos. La calidad de los datos es igualmente importante. Datos bien estructurados y relevantes permiten que el modelo aprenda de manera más efectiva. En el caso de problemas matemáticos, disponer de soluciones detalladas y explicaciones paso a paso facilita el aprendizaje profundo del modelo. Un dataset más grande ayuda a reducir el sesgo y el overfitting. Con más datos, el modelo puede generalizar mejor y no se limita a aprender únicamente los ejemplos específicos del conjunto de entrenamiento. Esto se traduce en un mejor rendimiento en datos de prueba y en aplicaciones del mundo real. Datos adicionales proporcionan un contexto más rico y variado, lo cual es crucial para modelos de lenguaje que dependen del contexto para generar respuestas precisas y coherentes. En el ámbito educativo, esto significa que los modelos pueden comprender y responder mejor a una amplia gama de preguntas y problemas.

El proceso de fine-tuning demostró ser una estrategia altamente efectiva para mejorar el rendimiento de todos los modelos evaluados en este estudio. Los resultados indicaron que el ajuste fino permite que los modelos de lenguaje grande se adapten mejor a tareas específicas, aprovechando al máximo los datos de entrenamiento adicionales. Se muestra que los modelos de código abierto, como `Meta-Llama-3-8B-Instruct` y `Mistral-7B-Instruct-v0.3`, mostraron mejoras significativas debido a su capacidad superior para capturar y aprender patrones más complejos en los datos. Esta

capacidad permite que los modelos comprendan mejor las sutilezas de los problemas matemáticos y las preguntas de tipo educativo. El fine-tuning permite que los modelos se adapten a tareas específicas, especializándose en resolver problemas concretos. Este ajuste mejora la relevancia y precisión de las respuestas generadas, ya que el modelo se entrena directamente en el tipo de datos que encontrará en la práctica. Las mejoras se reflejaron en la tasa de acierto. Los modelos fine-tuned mostraron un desempeño significativamente mejor en estas métricas en comparación con los modelos base, lo que subraya la efectividad del proceso de ajuste fino. Así como este entrenamiento ayudó en la mejora del razonamiento matemático de los modelos, el fine-tuning se puede aplicar a otras tareas específicas, como en feedback matemático, contribuyendo significativamente al aprendizaje personalizado y efectivo de los estudiantes. Al ajustar el feedback según las fortalezas y debilidades individuales, se facilita la corrección específica de errores comunes y se promueve un aprendizaje activo y autónomo. Esta personalización no solo mejora la comprensión inmediata de conceptos matemáticos, sino que también fortalece la capacidad de los estudiantes para autoevaluarse y corregir errores, creando así un ambiente de aprendizaje más enriquecedor y eficiente.

4.2 Análisis Económico

Finalmente, para evaluar estos modelos es esencial realizar un análisis económico detallado de los costos asociados con el entrenamiento y la inferencia de modelos de lenguaje grande. Este análisis considera los costos de los servicios utilizados, tanto para los modelos de código cerrado de OpenAI como para los modelos de código abierto utilizando Google Colab.

4.2.1 Costos de OpenAI

OpenAI ofrece servicios de API para el uso de sus modelos pre-entrenados, como `davinci-002` y `babbage-002`, y también permite realizar fine-tuning en estos modelos. Los costos asociados incluyen tanto el uso de la API para inferencia como el entrenamiento personalizado. El costo de uso de la API de OpenAI se basa en el número de tokens procesados y el entrenamiento de modelos en OpenAI implica costos adicionales que dependen del tamaño del modelo y la cantidad de datos utilizados.

Modelo	Costo de inferencia (modelo base)	Costo de entrenamiento	Costo de inferencia (modelo entrenado)
`babbage-002`	0,40 US$ / 1 M tokens	0,40 US$ / 1 M tokens	1,60 US$ / 1 M tokens
`davinci-002`	2,00 US$ / 1 M tokens	6,00 US$ / 1 M tokens	12,00 US$ / 1 M tokens

Tabla 4.2: Costos de la API de OpenAI por token.

4.2.2 Costos de Modelos Open Source

Para los modelos de código abierto, se utilizó Google Colab para las inferencias, en esta plataforma se utilizaron las librerías `trainer` de Hugging Face y la versión ajustada de Unsloth AI para modelos cuantizados para el entrenamiento. Google Colab ofrece recursos de GPU que son esenciales para la inferencia y entrenamiento de modelos grandes. Existen versiones gratuitas y de pago (Colab Pro y Colab Pro+).

Versión de Google Colab	Descripción	Costo
Colab Free	Ofrece acceso a máquinas virtuales con recursos limitados (T4). Permite ejecutar notebooks de Jupyter y realizar cómputo en la nube con GPU y TPU de forma intermitente y con restricciones de tiempo.	Sin costo
Colab Pro	Proporciona acceso prioritario a máquinas virtuales más potentes y estables (T4, A100 y L4). Permite ejecutar notebooks por períodos más prolongados (hasta 24 horas continuas). Ofrece almacenamiento persistente en Google Drive (hasta 100 GB) y 100 unidades de procesamiento al mes (se consumen por tiempo de uso durante la ejecución).	$9.99 por mes
Colab Pro+	Ofrece acceso a las máquinas virtuales más poderosas disponibles (T4, A100 y L4 con memoria ampliada). Incluye todas las características de Colab Pro, como ejecución prolongada de notebooks y almacenamiento persistente en Drive, además de 500 unidades de procesamiento y ejecución en segundo plano.	$49.99 por mes

Tabla 4.3: Detalle de versiones de Google Colab.

4.2.3 Comparación de Costos

Los costos de uso de la API y el entrenamiento son significativamente más altos, especialmente para modelos grandes como `davinci-002`. Para los modelos de código abierto utilizando Google Colab, los costos son mucho más manejables, especialmente para proyectos con recursos limitados. El análisis económico revela que, los modelos de OpenAI ofrecen una infraestructura robusta y de alta calidad, pero a un costo considerablemente mayor. Los modelos de código abierto, aunque requieren una configuración inicial más compleja, son más económicos y accesibles para proyectos con presupuestos limitados. La elección entre OpenAI y modelos de código abierto depende del balance entre el presupuesto disponible y la necesidad de rendimiento y facilidad de uso.

Los resultados de la comparación entre los modelos base y los modelos entrenados se analizaron para identificar mejoras en el rendimiento debido al fine-tuning. Los modelos pre-entrenados (versión base) se utilizaron como referencia para establecer un punto de comparación inicial. Estos modelos no habían sido ajustados con los conjuntos de datos específicos utilizados en este estudio. Los modelos ajustados mediante fine-tuning mostraron mejoras significativas en la métrica de evaluación, lo que indica que el proceso de ajuste fino fue efectivo para adaptar los modelos a las tareas específicas. Se observaron aumentos de precisión en los modelos entrenados en comparación con los modelos base.

La Tabla 4.4 presenta una comparación de los costos asociados al entrenamiento de modelos de lenguaje grande de código cerrado, específicamente `babbage-002` y `davinci-002`. Los costos mostrados son reales, excepto el costo de entrenamiento de `davinci-002` con el dataset completo, que es un valor teórico debido a las limitaciones económicas de la investigación y el tamaño significativo de los tokens involucrados. Es importante destacar que, para los modelos de código abierto, no se incurrió en gastos monetarios, ya que se utilizaron recursos gratuitos de Google Colab y bibliotecas de código abierto de Hugging Face. Esta comparación busca resaltar las diferencias económicas en el entrenamiento de modelos de código cerrado versus modelos de código abierto, subrayando la viabilidad económica de utilizar herramientas y recursos disponibles de forma gratuita en el ámbito educativo y de investigación.

Modelo	Tamaño dataset	Tokens de entrenamiento	Costo total de entrenamiento
`babbage-002`	50 % GSM8K	1.749.798	$0.70
	100 % GSM8K	3.534.408	$1.41
`davinci-002`	50 % GSM8K	1.749.798	$10.50
	100 % GSM8K	3.534.408	$21.21*

Tabla 4.4: Costos de entrenamiento para modelos de código cerrado.

Capítulo 5

Conclusiones

El análisis de los resultados confirma que el fine-tuning y el tamaño del dataset son factores críticos para mejorar el rendimiento de los modelos de lenguaje grande en el contexto educativo. Los modelos con más parámetros y una capacidad avanzada para el aprendizaje de patrones complejos se benefician más significativamente de estos procesos, permitiendo la entrega de retroalimentación de mejor calidad.

El análisis detallado de los resultados proporcionó información valiosa sobre el impacto del fine-tuning. El proceso de fine-tuning mejoró significativamente el rendimiento de los modelos en tareas específicas, demostrando la eficacia de esta técnica para adaptar modelos pre-entrenados a nuevos contextos. Los modelos entrenados mostraron una mayor capacidad para resolver problemas matemáticos, proporcionando respuestas más precisas y contextualmente relevantes. Los resultados sugieren que los modelos entrenados mediante fine-tuning pueden ser herramientas efectivas en el ámbito educativo, ayudando a los estudiantes a comprender y resolver problemas complejos de manera más eficiente. El fine-tuning demostró ser una técnica efectiva para mejorar el rendimiento de los modelos de lenguaje grande en tareas específicas, como el razonamiento matemático. Los modelos ajustados mediante fine-tuning mostraron una mayor capacidad para resolver problemas matemáticos, proporcionando respuestas más precisas y contextualmente relevantes. Para la eficacia del Proceso, el proceso de fine-tuning permitió adaptar los modelos pre-entrenados a nuevos contextos, mejorando significativamente su desempeño en tareas específicas.

La relación entre tamaño del dataset y rendimiento, se observó que el aumento del tamaño del dataset de entrenamiento mejora considerablemente el rendimiento de los modelos. Por ejemplo, el modelo `babbage-002` mostró un incremento en la precisión del 3 % al 12 % al aumentar el tamaño del dataset de 3736 a 7473 ejemplos. Concluyendo que el aumento del tamaño del dataset de entrenamiento

permite alcanzar mejoras sustanciales en el rendimiento de los modelos, haciéndolos más precisos y relevantes para tareas específicas.

Los modelos de OpenAI, como `davinci-002` y `babbage-002`, tienen costos de uso significativos. Por ejemplo, el costo de inferencia para `davinci-002` es de $2,00 USD por millón de tokens y el costo de entrenamiento es de $6,00 USD por millón de tokens. Los modelos de código abierto utilizando plataformas como Google Colab son mucho más manejables en términos de costos. Google Colab ofrece opciones gratuitas y de pago (Colab Pro y Colab Pro+), con costos que van desde $9,99 USD a $49,99 USD por mes. El análisis económico reveló que los modelos de OpenAI, aunque ofrecen una infraestructura robusta y de alta calidad, son considerablemente más costosos que los modelos de código abierto. Los modelos open source, aunque requieren una configuración inicial más compleja, son más económicos y accesibles para proyectos con presupuestos limitados.

Los modelos `babbage-002` y `davinci-002` de OpenAI, mostraron mejoras significativas con el fine-tuning. Aunque `babbage-002` tiene menos parámetros y comenzó con un rendimiento muy bajo, el fine-tuning mejoró su precisión considerablemente. Por otro lado, `davinci-002`, con una cantidad significativamente mayor de parámetros, mostró mejoras notables después del fine-tuning.

Los modelos de código abierto `Mistral-7B-Instruct-v0.3` y `Meta-Llama-3-8B-Instruct` mostraron un rendimiento sobresaliente desde el inicio y continuaron mejorando con el fine-tuning. Estos modelos demostraron una mayor capacidad para capturar y aprender patrones complejos en los datos, lo que les permitió aprovechar al máximo los datos adicionales.

El presente estudio enfrenta varias limitaciones que impactan su aplicabilidad y generalización. Una de las principales limitaciones es la dependencia de los modelos pre-entrenados de datasets de gran tamaño y calidad, lo cual puede restringir la capacidad de los modelos ajustados. Los modelos pre-entrenados, aunque poderosos, están intrínsecamente ligados a la calidad y diversidad de los datos con los que fueron entrenados inicialmente. Si estos datos presentan sesgos o carecen de representatividad, el rendimiento del modelo ajustado se verá afectado negativamente en contextos reales. Además, la naturaleza unimodal de estos modelos presenta desafíos significativos en la resolución de problemas que requieren la integración de información visual y textual, limitando su efectividad en contextos educativos más amplios. Los modelos unimodales, diseñados para procesar exclusivamente texto o

imágenes, no pueden abordar eficientemente tareas que involucran múltiples modalidades de datos, como la combinación de gráficos con explicaciones textuales en problemas de física o matemáticas. Otra limitación importante es el alto costo computacional del proceso de fine-tuning, que requiere acceso a infraestructuras tecnológicas avanzadas. El fine-tuning de modelos de lenguaje grande demanda recursos computacionales significativos, como GPU de alto rendimiento o clústeres de TPU, los cuales no están disponibles en todas las instituciones educativas. Este requisito de hardware especializado no solo eleva los costos operativos, sino que también limita la accesibilidad de estas técnicas a instituciones con menor capacidad financiera. La calidad de los resultados está directamente ligada a la calidad y diversidad de los datos utilizados para el entrenamiento. Cualquier sesgo en estos datos puede afectar la precisión y utilidad del modelo en situaciones del mundo real. Es crucial que los datos de entrenamiento sean variados y representativos de los diferentes escenarios en los que se espera que el modelo opere. La falta de diversidad en los datos de entrenamiento puede conducir a un rendimiento sub-óptimo y a la perpetuación de sesgos existentes. Además, el uso de datos sesgados o de baja calidad durante el fine-tuning puede amplificar los errores y limitaciones inherentes del modelo pre-entrenado, haciendo que sus aplicaciones en el entorno educativo sean menos fiables. Por lo tanto, es fundamental implementar estrategias robustas para la recolección, limpieza y preparación de datos antes de llevar a cabo el fine-tuning. Es importante mencionar que la evaluación de la efectividad del fine-tuning en contextos educativos requiere un enfoque multidimensional que no solo considere el rendimiento del modelo en términos de precisión, sino también su capacidad para proporcionar explicaciones comprensibles y útiles para los estudiantes. Este enfoque debe incluir métricas que evalúen la calidad de las explicaciones generadas por el modelo y su impacto en el aprendizaje de los estudiantes.

Futuros trabajos podrían centrarse en desarrollar modelos multimodales que integren de manera efectiva información visual y textual, mejorando así la capacidad de los modelos para abordar problemas complejos en educación. Además, la optimización de los algoritmos de fine-tuning para reducir su demanda computacional es una área prometedora, permitiendo una adopción más amplia en instituciones con recursos limitados. Asimismo, la implementación de enfoques colaborativos, donde varios modelos trabajen en conjunto para resolver problemas, podría aumentar la precisión y adaptabilidad de los modelos en diferentes contextos educativos. La evaluación continua y la actualización de los modelos con nuevos datos serán esenciales para mantener su relevancia y efectividad en la enseñanza de matemáticas y física. Un enfoque innovador sería integrar la técnica de Retrieval-Augmented

Generation (RAG, por su sigla en inglés), que permite a los modelos de lenguaje grande acceder a información externa y relevante (proporcionada por el usuario) durante la generación de respuestas, permitiendo respuestas más precisas con respecto a un dominio específico (Lewis et al., 2021). Al hacer fine-tuning a un LLM y utilizar RAG, se puede mejorar la capacidad del modelo para proporcionar respuestas más precisas y contextualizadas. Esto es especialmente útil en el ámbito educativo, donde la exactitud y la relevancia de la información son cruciales. Para implementar RAG de manera efectiva, se debe comenzar por el proceso de Indexing, donde se cargan, dividen y almacenan los datos relevantes en una base de datos. Luego, durante la generación de respuestas, el modelo recupera información pertinente de esta base de datos y la utiliza para enriquecer sus respuestas. Esta técnica no solo mejora la precisión del modelo, sino que también amplía su capacidad para manejar consultas complejas que requieren conocimientos actualizados y específicos.

La combinación de un proceso de fine-tuning efectivo, datasets grandes y de alta calidad permite alcanzar mejoras sustanciales en el rendimiento de los modelos, haciéndolos más precisos y relevantes para tareas específicas. Estos hallazgos subrayan la importancia de invertir en datos de alta calidad y en técnicas avanzadas de entrenamiento para desarrollar modelos de lenguaje grande que puedan satisfacer las demandas de aplicaciones complejas y específicas, como las educativas.

Referencias

Angels Balaguer, Vinamra Benara, Renato Luiz de Freitas Cunha, Roberto de M. Estevão Filho, Todd Hendry, Daniel Holstein, Jennifer Marsman, Nick Mecklenburg, Sara Malvar, Leonardo O. Nunes, Rafael Padilha, Morris Sharp, Bruno Silva, Swati Sharma, Vijay Aski, and Ranveer Chandra. Rag vs fine-tuning: Pipelines, tradeoffs, and a case study on agriculture, 2024.

Samyadeep Basu, Martin Grayson, Cecily Morrison, Besmira Nushi, Soheil Feizi, and Daniela Massiceti. Understanding information storage and transfer in multi-modal large language models, 2024.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, and Adam Roberts. Palm: Scaling language modeling with pathways, 2022.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023.

Katja Grace, Harlan Stewart, Julia Fabienne Sandkühler, Stephen Thomas, Ben Weinstein-Raun, and Jan Brauner. Thousands of ai authors on the future of ai, 2024.

Arto Hellas, Juho Leinonen, and Leo Leppänen. Experiences from integrating large language model chatbots into the classroom, 2024. URL https://arxiv.org/abs/2406.04817.

Jiwoo Hong, Noah Lee, and James Thorne. Orpo: Monolithic preference optimization without reference model, 2024. URL https://arxiv.org/abs/2403.07691.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

Hanxu Hu, Pinzhen Chen, and Edoardo M. Ponti. Fine-tuning large language models with sequential instructions, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. ISSN 1476-4687. doi: 10.1038/nature14539.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

Qingyao Li, Lingyue Fu, Weiming Zhang, Xianyu Chen, Jingwei Yu, Wei Xia, Weinan Zhang, Ruiming Tang, and Yong Yu. Adapting large language models for education: Foundational capabilities, potentials, and challenges, 2024.

Zihao Li, Zhuoran Yang, and Mengdi Wang. Reinforcement learning with human feedback: Learning dynamic choices via pessimism, 2023.

Naiming Liu, Shashank Sonkar, Zichao Wang, Simon Woodhead, and Richard G. Baraniuk. Novice learner and expert tutor: Evaluating math reasoning abilities of large language models with misconceptions, 2023a.

Yixin Liu, Avi Singh, C. Daniel Freeman, John D. Co-Reyes, and Peter J. Liu. Improving large language model fine-tuning for solving math problems, 2023b.

Joy Mahapatra and Utpal Garain. Impact of model size on fine-tuned llm performance in data-to-text generation: A state-of-the-art investigation, 2024. URL https://arxiv.org/abs/2407.14088.

Kamil Malinka, Martin Peresíni, Anton Firc, Ondrej Hujnák, and Filip Janus. On the educational impact of chatgpt: Is artificial intelligence ready to obtain a university degree? In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*, ITiCSE 2023. ACM, June 2023. doi: 10.1145/3587102.3588827. URL http://dx.doi.org/10.1145/3587102.3588827.

Nick Mecklenburg, Yiyou Lin, Xiaoxiao Li, Daniel Holstein, Leonardo Nunes, Sara Malvar, Bruno Silva, Ranveer Chandra, Vijay Aski, Pavan Kumar Reddy Yannam, Tolga Aktas, and Todd Hendry. Injecting new knowledge into large language models via supervised fine-tuning, 2024.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, and Florencia Leoni Aleman. Gpt-4 technical report, 2024.

Teo Susnjak. Chatgpt: The end of online exam integrity?, 2022.

Gemini Team, Rohan Anil, Sebastian Borgeaud, and Jean-Baptiste Alayrac. Gemini: A family of highly capable multimodal models, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

Ke Wang, Houxing Ren, Aojun Zhou, Zimu Lu, Sichun Luo, Weikang Shi, Renrui Zhang, Linqi Song, Mingjie Zhan, and Hongsheng Li. Mathcoder: Seamless code integration in llms for enhanced mathematical reasoning, 2023.

Shen Wang, Tianlong Xu, Hang Li, Chaoli Zhang, Joleen Liang, Jiliang Tang, Philip S. Yu, and Qingsong Wen. Large language models for education: A survey and outlook, 2024.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

Gokul Yenduri, Ramalingam M, Chemmalar Selvi G, Supriya Y, Gautam Srivastava, Praveen Kumar Reddy Maddikunta, Deepti Raj G, Rutvij H Jhaveri, Prabadevi B, Weizheng Wang, Athanasios V.

Vasilakos, and Thippa Reddy Gadekallu. Generative pre-trained transformer: A comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions, 2023.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models, 2023.

Anexo 1

```
%%capture
%pip install "unsloth[colab-new] @ git+https://github.com/unslothai/unsloth.git"
%pip install --no-deps xformers trl peft accelerate bitsandbytes
```

Anexo 2

question string · lengths 137↔232 46%	answer string · lengths 50↔168 19.9%
Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?	Natalia sold 48/2 = <<48/2=24>>24 clips in May. Natalia sold 48+24 = <<48+24=72>>72 clips altogether in April and May. #### 72
Weng earns $12 an hour for babysitting. Yesterday, she just did 50 minutes of babysitting. How much did she earn?	Weng earns 12/60 = $<<12/60=0.2>>0.2 per minute. Working 50 minutes, she earned 0.2 x 50 = $<<0.2*50=10>>10. #### 10
Betty is saving money for a new wallet which costs $100. Betty has only half of the money she needs. Her parents decided to giv…	In the beginning, Betty has only 100 / 2 = $<<100/2=50>>50. Betty's grandparents gave her 15 * 2 = $<<15*2=30>>30. This…

Anexo 3

```
tokenizer = get_chat_template(
    tokenizer,
    chat_template = "chatml",
    mapping = {"role" : "from", "content" : "value", "user" : "human", "assistant" : "gpt"},
    map_eos_token = True
)
```

Anexo 4

```
Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning

and bakes muffins for her friends every day with four.

She sells the remainder at the farmers' market daily for $2 per freshduck egg.

How much in dollars does she make every day at the farmer's market?
```

Anexo 5

```
Janet sells 16 - 3 - 4 = <<16-3-4=9>>9 duck eggs a day.
She makes 9 * 2 = $<<9*2=18>>18 every day at the farmer's market.
#### 18
```

Anexo 6

```
First, let's find out how many eggs Janet has left after
consuming three for breakfast and baking muffins with four.
Eggs consumed: 3 (for breakfast) + 4 (for muffins) = 7 eggs
Now, let's find out how many eggs are left for sale each day.
We know that Janet's ducks lay 16 eggs per day.
Eggs for sale: 16 (total eggs laid) - 7 (consumed) = 9 eggs
To find out how much money Janet makes each day selling these
eggs at the farmers' market, we need to multiply the number of
eggs sold by the price per egg.
Money earned: 9 (eggs for sale) × $2 (price per egg) = $18
So, Janet makes $18 in dollars every day at the farmer's market.
```

Anexo 7

```
Janet eats 3 + 4 = <<3+4=7>>7 eggs per day.

The ducks lay 16 - 7 = <<16-7=9>>9 eggs per day.

She sells 9 x $2 = $<<9*2=18>>18 worth of eggs per day.

#### 18
```

Anexo 8

```
Janet lays 16 eggs per day. She eats 3 eggs for breakfast, so

she has 13 eggs left. She bakes muffins with 4 eggs, so she

has 9 eggs left. She sells the remaining 9 eggs at the farmers

market for $2 per egg, so she makes $18 per day.

$18/day = $<<18=1800>>1800/year

#### 1800
```

Anexo 9

```
Shiela bought five cell phones for $150 each for a 3-month installment.

A 2% interest will be charged for each unit. How much will Shiela pay

each month for 3 months?
```

Anexo 10

```
A $150 x 2/100 = $<<150*2/100=3>>3 interest will be charged for each unit.

So, each cell phone costs $150 + $3 = $<<150+3=153>>153.

The total amount to be paid for 5 cell phones is $153 x 5 = $<<153*5=765>>765.

Thus, Shiela will pay $765/3 = $<<765/3=255>>255 each month.

#### 255
```

Anexo 11

```
Let's break down the problem:

We have 5 cell phones, each costing $150. So, the total cost is:

5 x $150 = $7500

Since it's a 3-month installment, we need to divide the total cost by 3:

$7500 ÷ 3 = $2500

Now, we need to calculate the interest for each month. Since it's 2% interest,

we need to calculate 2% of $2500:

2% of $2500 = $2500 x 0.02 = $50

So, the interest for each month is $50. To find the total amount Shiela needs

to pay each month, we add the interest to the installment:

$2500 (installment) + $50 (interest) = $2550

Therefore, Shiela will pay $2550 each month for 3 months.
```

Anexo 12

```
Five cell phones cost $150 x 5 = $<<150*5=750>>750.

The interest for each cell phone is $750 x 2/100 = $<<750*2/100=15>>15.

So, Shiela will pay $150 + $15 = $<<150+15=165>>165 each for 3 months.

#### 165
```

Anexo 13

```
Five cell phones cost $150 x 5 = $<<150*5=750>>750.

The interest for five cell phones is $750 x 2/100 = $<<750*2/100=15>>15.

So, the total amount that Shiela will pay for the 5 cell phones is

$750 + $15 = $<<750+15=765>>765.\nTherefore, Shiela will pay

$765/3 = $<<765/3=255>>255 each month for 3 months.

#### 255
```

UNIVERSIDAD DE CONCEPCIÓN – FACULTAD DE INGENIERÍA

RESUMEN DE MEMORIA DE TÍTULO

Departamento: Departamento de Ingeniería

Carrera: Ingeniería Civil Industrial

Nombre del memorista: Luis Andrés Melita Cruces

Título de la memoria: EVALUACIÓN DEL USO DE FINE-TUNING EN MODELOS DE LENGUAJE GRANDE COMO HERRAMIENTA DE APRENDIZAJE AJUSTADA A LAS ÁREAS DE FÍSICA Y MATEMÁTICAS EN LA EDUCACIÓN

Fecha de la presentación oral: 21-08-2024

Profesor Guía: Carlos Camilo Navarrete Lizama

Profesora Co-Guía: Alejandra Marcela Maldonado Trapp

Profesor Revisor : Jorge Ignacio Maluenda Albornoz

Concepto: Investigación

Resumen:

El presente trabajo tiene como objetivo principal explorar estrategias de fine-tuning para optimizar el desempeño de modelos de lenguaje grande en la resolución de problemas en educación. La metodología aplicada abarca la cuantización de modelos, la preparación del entorno de desarrollo, la selección de modelos pre-entrenados y la recopilación de datos. Se llevaron a cabo experimentos de fine-tuning para mejorar el rendimiento en tareas educativas específicas de física y matemáticas. Los resultados muestran que el fine-tuning de LLMs es eficaz para adaptar modelos pre-entrenados a tareas educativas, mejorando su precisión y utilidad en la enseñanza de conceptos complejos. Esto sugiere que los LLMs ajustados no solo proporcionan respuestas precisas, sino que también contribuyen al aprendizaje profundo y a la mejora de las habilidades de resolución de problemas de los estudiantes.