GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

Análisis de modelos de lenguaje preentrenados: Evaluación de Rendimiento, calidad de respuesta y Aplicación en ingeniería

Autor: Marco Rodriguez Segura

Director: Alfonso Vázquez Requejo

24 enero 2024, Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Análisis de Modelos de Lenguaje Preentrenados: Evaluación de Rendimiento, calidad de respuesta y Aplicación en la ingeniería en la ETS de Ingeniería – ICAI de la Universidad Pontificia Comillas en el curso académico 2023/24 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Marco Rodríguez Segura Fecha: 09/07/2024

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Alfonso Vázquez Requejo Fecha: 10 / 07 / 2024

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

Análisis de modelos de lenguaje preentrenados: Evaluación de Rendimiento, calidad de respuesta y Aplicación en ingeniería

Autor: Marco Rodriguez Segura

Director: Alfonso Vázquez Requejo

24 enero 2024, Madrid

Agradecimientos

A mi abuelo Ángel,

Por ser la inspiración para estudiar ingeniería,

Por sembrar en mi la curiosidad con aquellas réplicas de caterpilar,

Por enseñarme con tu ejemplo las virtudes de la paciencia y la calma,

Por seguirme en cada paso del camino,

Y, sobre todo, por seguir disfrutándote mucho más.

ANALYSIS OF PRE-TRAINED LANGUAGE MODELS: PERFORMANCE EVALUATION, RESPONSE QUALITY, AND APPLICATION IN ENGINEERING

Author: Rodriguez Segura, Marco.
Supervisor: Vázquez Requejo, Alfonso.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

PROJECT SUMMARY

This study examines the advances in pre-trained language models and their application in engineering, evaluating the effectiveness of three main AIs (ChatGPT-3.5, ChatGPT-4, and Gemini) through comparative testing. The results show that ChatGPT-4 is the most capable AI, with a performance close to 80%. The conclusion is that AIs are extremely capable of solving technical problems but are not yet ready for autonomous applications in engineering. Their use is recommended as support tools.

Keywords: Pre-trained Language Models, Artificial Intelligence, transformers

1. Introduction

Artificial intelligence (AI), defined by John McCarthy as "the science and engineering of making intelligent machines," aims to emulate human intelligence through technology. This work examines technological advances in Natural Language Processing (NLP) models and their application in engineering. It explores the theory that enables this technology to function and the practical application of AI in engineering, reshaping the future of the discipline.

2. Proyect definition

First, a generalizable theoretical framework is established for engineers interested in systematizing processes with AI. Second, the capability of AIs to solve engineering problems is evaluated through comparative testing. For the analysis, test bases composed of series of questions with their correct answers are used. The AIs will answer these questions, and their responses will be compared to those in the test base. To obtain these answers, Python programming will be used to call the artificial intelligence from the code.

3. Model/sistem/tool description

Pre-trained language models are artificial intelligence algorithms that use transformer architecture and large amounts of data to emulate human knowledge. In this project, these systems will be accessed through an API, which allows for connecting with the artificial intelligence via code. This facilitates and enables the systematization of interactions with the artificial intelligence.

4. Results

The results of the artificial intelligence on the test bases are as follows:

Colección de datos	ChatGPT4	ChatGPT3.5	Gemini
STEM-AI	82,65%	77,56%	74,81%
GHOSTS	79,00%	73,00%	71,00%
Análisis de sentimiento	86,63%	86,09%	85,48%
Promedio	**83,23%**	**79,39%**	**78,48%**

Table 15: Datasets Results

5. Conclusions

In general, the AIs have a performance close to 80%, a notable result given the intensive development since November 2022. However, this level is still not sufficient for autonomous application in engineering, where the margins for error are narrower. The ideal way to use AI is under the concept of a copilot, understanding its advantages and strengths to lighten the workload of the pilot, in this case, the engineer.

6. References

[1] Vlassis, N.A.; Papakonstantinou, G.; Tsanakas, P. Dynamic sensory probabilistic maps for mobile robot localization. Source: Proceedings. 1998 IEEE / RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190) New York, NY, USA: IEEE, 1998.p.718-23 vol.2 of 3 vol. xliv+2010 pp. 11. Último acceso 08/05/2024

[2] Loeffler, B. "Cloud Computing: What is Infrastructure as a Service", Microsoft Technet Magazine, October 211. https://technet.microsoft.com/en-us/magazine/hh509051.aspx Último acceso 08/05/2024

[3] Herrero Alcántara, T. "Big Data: ¿Moda u oportunidad de negocio para el emprendedor?", Think Big, Octubre 2014. http://blogthinkbig.com/big-data-emprendedor/. Último acceso 08/05/2024

ANÁLISIS DE MODELOS DE LENGUAJE PREENTRENADOS: EVALUACIÓN DE RENDIMIENTO, CALIDAD DE RESPUESTA Y APLICACIÓN EN LA INGENIERÍA

Autor: Rodriguez Segura, Marco.
Director: Vázquez Requejo, Alfonso.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas)

RESUMEN DEL PROYECTO

Este trabajo examina los avances de los modelos de lenguaje preentrenados y su aplicación en la ingeniería, evaluando la eficacia de tres IA principales (ChatGPT-3.5, ChatGPT-4 y Gemini) mediante pruebas comparativas. Los resultados muestran que ChatGPT-4 es la IA más capaz, con un rendimiento cercano al 80%. Se concluye que las IA son extremadamente capaces de resolver problemas de índole técnica pero aún no están listas para aplicaciones autónomas en ingeniería y se recomienda su uso como herramientas de apoyo.

Palabras clave: Modelos de lenguaje preentrenados, Inteligencia Artificial, transformers

1. Introducción

La inteligencia artificial (IA), definida por John McCarthy como "la ciencia e ingeniería de hacer máquinas inteligentes", busca emular la inteligencia humana mediante tecnología. Este trabajo examina los avances tecnológicos en modelos de Procesamiento del Lenguaje Natural (NLP) y su aplicación en ingeniería. Explorando la teoría que permite que esta tecnología funcione y la aplicación práctica de la IA en la ingeniería, rediseñando el futuro de la disciplina.

2. Definición del proyecto

Primero, se establece un marco teórico generalizable para ingenieros interesados en sistematizar procesos con IA. Segundo, se evalúa la capacidad de las IA para resolver problemas de ingeniería mediante pruebas comparativas. Para el análisis, se usan bases de prueba compuestas por series de preguntas con su respuesta correcta. Las IA responderán estas preguntas, y sus respuestas serán comparadas con la de la base de pruebas. Para conseguir estas respuestas se hará uso de la programación en Python llamando a la inteligencia artificial desde el código.

3. Descripción del modelo/sistema/herramienta

Los modelos de lenguaje preentrenados son algoritmos de inteligencia artificial que utilizan la arquitectura transformers y grandes cantidades de datos para emular el conocimiento humano. En este proyecto se accederán a estos sistemas a través de una API; que permite conectar con la inteligencia artificial a través de un código, esto facilita y permite sistematizar las interacciones con la inteligencia artificial.

4. Resultados

Los resultados de la inteligencia artificial a las bases de pruebas son los siguientes:

Colección de datos	ChatGPT4	ChatGPT3.5	Gemini
STEM-AI	82,65%	77,56%	74,81%
GHOSTS	79,00%	73,00%	71,00%
Análisis de sentimiento	86,63%	86,09%	85,48%
Promedio	**83,23%**	**79,39%**	**78,48%**

Tabla 15: Resultados a las bases de pruebas

5. Conclusiones

En general, las IA tienen un rendimiento cercano al 80%, un resultado notable dado el reciente desarrollo intensivo desde noviembre de 2022. Sin embargo, este nivel aún no es suficiente para la aplicación autónoma en ingeniería, donde los márgenes de error son más estrechos. La forma ideal de utilizar la IA, es bajo el concepto de copiloto, conocer las bondades y las fortalezas para aligerar el trabajo del piloto, en este caso el ingeniero.

6. Referencias

[1] Vlassis, N.A.; Papakonstantinou, G.; Tsanakas, P. Dynamic sensory probabilistic maps for mobile robot localization. Source: Proceedings. 1998 IEEE / RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190) New York, NY, USA: IEEE, 1998.p.718-23 vol.2 of 3 vol. xliv+2010 pp. 11. Último acceso 08/05/2024

[2] Loeffler, B. “Cloud Computing: What is Infrastructure as a Service”, Microsoft Technet Magazine, October 211. https://technet.microsoft.com/en-us/magazine/hh509051.aspx Último acceso 08/05/2024

[3] Herrero Alcántara, T. “Big Data: ¿Moda u oportunidad de negocio para el emprendedor?”, Think Big, Octubre 2014. http://blogthinkbig.com/big-data-emprendedor/. Último acceso 08/05/2024

Índice de la memoria

Índice de Figuras

Índice de Tablas

GLOSARIO

RNN (Redes Neuronales Recurrentes): Algoritmo de procesamiento de texto que utiliza retroalimentación de información anterior para evaluar datos actuales. Limita a una entrada y una salida y sufre de problemas de memoria a corto plazo y desvanecimiento del gradiente.

LSTM (Long Short Term Memory): Redes de memoria a corto y largo plazo diseñadas para resolver los problemas de memoria a corto plazo de las RNN. Utilizan una estructura de cadena y añaden complejidad en el proceso de retroalimentación de información.

NLP (Natural Language Processing): Procesamiento de Lenguaje Natural, modelos que comprenden y procesan el lenguaje humano, utilizados para generar respuestas coherentes y similares a las de un humano en interacciones tecnológicas.

Machine Learning: Campo de la inteligencia artificial que permite a las máquinas aprender de los datos y mejorar su rendimiento con el tiempo sin ser explícitamente programadas para cada tarea específica.

Embedding: Método de clasificación de palabras en un espacio de n-dimensiones, asignando vectores a cada palabra y permitiendo operaciones matemáticas para detectar relaciones y similitudes entre ellas.

Lematización: Proceso de simplificación de frases para reducir el número de parámetros almacenados en cada token, mejorando la relación entre ellos y facilitando la identificación de similitudes.

Token: Unidad básica de datos en el procesamiento de lenguaje natural que puede ser una palabra, sub-palabra o carácter. La tokenización reduce el número de datos con los que trabaja el algoritmo al asignar un número identificador a cada conjunto de datos repetitivo.

Transformers: Algoritmo que revolucionó la inteligencia artificial y el procesamiento de lenguaje natural, capaz de procesar texto sin perder memoria y en paralelo, superando las limitaciones de los algoritmos de redes recurrentes.

Attention: Concepto en los Transformers que permite al modelo comprender el significado de una oración mediante la comparación de cada palabra con las demás y ponderando su importancia relativa.

Positional Encoding: Método utilizado en los Transformers para incluir información posicional en el vector codificado de cada palabra, permitiendo al algoritmo procesar frases completas y mantener el orden y el sentido de las palabras.

TF-IDF (Term Frequency-Inverse Document Frequency): Técnica utilizada en recuperación de información y minería de texto para evaluar la importancia de una palabra en un documento respecto a una colección o corpus de documentos.

Cosine Similarity: Métrica utilizada para medir la similitud entre dos vectores en un espacio de múltiples dimensiones, comúnmente usada en procesamiento de texto basado en TF-IDF.

Supervised Fine-Tuning (SFT): Fase de ajuste fino supervisado en el entrenamiento de modelos de IA, donde se enseña al modelo a dar respuestas ideales mediante ejemplos específicos y retroalimentación directa.

Prompting: Acto de proporcionar información al algoritmo de IA para obtener una mejor respuesta.

1

INTRODUCCIÓN

La inteligencia artificial (IA), como definió John McCarthy, uno de los pioneros en esta disciplina, se describe como "la ciencia e ingeniería de hacer máquinas inteligentes, especialmente programas informáticos inteligentes". Esta ciencia se alinea con la meta de comprender el conocimiento humano mediante el uso de la tecnología, pero se distancia de otras ramas científicas, como la biología que se centra en el estudio de los seres vivos o la física que explora las propiedades y el comportamiento de la energía. Mientras, la inteligencia artificial que indaga las rutas computacionales para emular y potenciar la inteligencia humana. A diferencia de los seres vivos, la IA puede procesar y analizar datos a

una escala y velocidad que desafían la capacidad humana. (*¿Qué es la inteligencia artificial (IA)?*)

Hoy en día, las contribuciones de la IA se han vuelto omnipresentes, infiltrándose en actividades cotidianas y transformando prácticas rutinarias en procesos automatizados y eficientes. Desde el análisis de imágenes de alta calidad, que ahora agrupa el diagnóstico médico y la verificación de identidad, la conducción autónoma, que promete revolucionar el transporte, son ejemplos de cómo la IA ha materializado una evolución en numerosas industrias.

La semilla de la inteligencia artificial fue plantada en 1956 durante la conferencia de Dartmouth, un evento que marcó el reconocimiento formal de la disciplina, y que también consolidó la colaboración interdisciplinaria como su piedra angular. Con el objetivo ambicioso de imitar la inteligencia humana, los primeros años de la IA existía un optimismo protagonizado por una serie de desarrollos significativos. Sin embargo, los investigadores pronto encontraron limitaciones; la tecnología disponible resultó insuficiente para alcanzar las metas propuestas, dando lugar a un fenómeno conocido como "invierno de la IA". (*Conferencia de Dartmouth: Orígenes de la Inteligencia artificial* 2022)

A pesar de los desafíos, la resiliencia de los científicos y su dedicación a la causa no decayó. El campo del machine learning, que engloba la inteligencia artificial, experimentó una evolución, donde el aprendizaje de las máquinas se convirtió en el foco, superando la simple imitación de tareas humanas. Con el surgimiento de algoritmos más sofisticados y el acceso a poder computacional sin precedentes, la IA ha comenzado a cumplir algunas de sus promesas originales, redefiniendo lo que es posible y estableciendo un nuevo paradigma en la relación entre los seres humanos y las máquinas.

A medida que esta disciplina continúa madurando, la sinergia entre la IA y campos como la ingeniería se hace cada vez más cercana. La integración de la inteligencia artificial en la

ingeniería no solo ha optimizado los procesos existentes, sino que también ha abierto horizontes para la innovación en diseño, fabricación, y solución de problemas complejos.

Este campo ha evolucionado hasta dejar al mundo asombrado por la capacidad de comprensión y ejecución de la IA. Este trabajo de fin de grado aborda dicha integración, explorando cómo la IA va a influenciar y transformar la ingeniería, sirviendo como punto de referencia para ingenieros cuando busquen conocer las limitaciones y fortalezas de la IA en un campo tan amplio y profundo.

Hay numerosas formas de nombrar a los modelos que coloquialmente se conocen como modelos de inteligencia artificial. NLP (Natural Language Processing) es el nombre técnico que reciben, ya que, como su nombre indica son capaces de descifrar y comprender el lenguaje natural humano. Todos estos modelos se engloban dentro de la disciplina del machine learning que abarca el entrenamiento autómata de los ordenadores. Estos modelos de estudio destacan por su amplia gama de aplicaciones y versatilidad en la resolución de problemas de ingeniería. Si bien se han desarrollado distintos tipos de IA con enfoques similares, en este caso el interés recae en aquellos que procesan información de tipo texto, aunque otras que son capaces de procesar imágenes y videos. Estos últimos, por ejemplo, operan bajo un principio similar al del lenguaje: interpretan una secuencia, ya sea de píxeles en imágenes o de cuadros en videos, donde cada elemento tiene una distribución y orden específicos que la IA aprende a analizar y comprender.

La inteligencia artificial (IA) es el nombre que se le confiere a los modelos que tienen la capacidad de generar contenido escrito o gráfico único en respuesta a un input. Estos, conocidos como modelos NLP (Natural Language Processing) son aquellos que son capaces de procesar lenguaje natural humano tal y como su nombre indica. Han sido capaces de generar una disrupción en la forma que se interactúa con los ordenadores y internet. Antes, los buscadores ostentaban el monopolio de la información, situando a Google como una de las empresas más valiosas del mundo. Ahora, estos modelos son capaces de llegar un paso

más allá, y generar respuestas a peticiones, esquivando el paso de la búsqueda de información. La primera empresa en sacar al mercado un modelo de inteligencia artificial capaz de generar respuestas a peticiones fue OpenAI con su modelo ChatGPT (Chat Generative Pre-trained Transformer) que sorprendió al mundo por su gran capacidad de emular las respuestas de un humano.

La impresionante historia del ascenso de ChatGPT prueba la relevancia de estos modelos en el futuro de la historia de la humanidad. En 2020, OpenAI lanzó ChatGPT,y consiguió batir el récord de llegar a 100 millones de usuarios en 60 días, superando la marca anterior establecida por Google+ en 720 días.

Rápidamente, al detectar el potencial en este sector, el mercado, con las grandes empresas tecnológicas a la cabeza, han sacado su propia versión de inteligencia artificial. Google tiene a Gemini, Facebook tiene a LLaMA, OpenAI tiene ChatGPT y por último Anthropic con ClaudeAI.

Gracias a este rápido surgimiento de la tecnología nace el proyecto de este TFG, que consiste en servir de base para todos los ingenieros en descubrir cuales son las capacidades actuales de la inteligencia artificial en las diferentes ramas de la ingeniería.

El propósito central es realizar un exhaustivo análisis del estado actual y el avance tecnológico en el campo de los modelos de Procesamiento del Lenguaje Natural (NLP), focalizando específicamente en su aplicación dentro del ámbito de la ingeniería. Este estudio tiene como objetivo proporcionar una visión detallada y actualizada de cómo operan estos modelos, así como identificar sus puntos fuertes y débiles, enfocándose exclusivamente en su aplicación práctica en el contexto ingenieril. Este trabajo busca, en esencia, ofrecer una comprensión profunda y completa de la tecnología NLP y su relevancia para los desafíos y demandas específicas del campo de la ingeniería.

Dado que utilizar plenamente una tecnología implica comprender en detalle su arquitectura y funcionamiento, resulta fundamental examinar cómo se han desarrollado estos modelos. En este proyecto, se abordará este desafío al poner estas herramientas de las diferentes potencias tecnológicas en comparación directa, lo que permitirá identificar y analizar las fortalezas individuales de cada una. Con este proyecto, no solo se comprenderá la amplitud del espectro tecnológico disponible, sino también diferenciar cómo estas herramientas pueden aplicarse de manera óptima en el campo de la ingeniería y si existe alguna herramienta especializada en ámbito concreto.

La comunicación entre el ser humano y los ordenadores ha sido un campo de estudio y desarrollo que ha evolucionado de manera significativa desde los inicios de la informática. Tradicionalmente, esta comunicación se ha realizado a través de la programación: un conjunto de instrucciones precisas y bien definidas que le indican a la terminal cómo realizar tareas específicas. Este método ha demostrado ser extremadamente eficaz para operaciones rutinarias y repetitivas, su aplicación se encuentra limitada cuando se enfrenta a tareas que requieren una comprensión contextual que imitan las capacidades humanas, como comprender el lenguaje humano.

La inteligencia artificial (IA) surge como un concepto revolucionario que promete superar las barreras de la programación convencional. A diferencia de la programación tradicional, que depende de la codificación manual de cada posible acción y reacción, la IA se centra en el desarrollo de sistemas capaces de aprender y adaptarse a nuevas situaciones sin intervención directa, además es capaz de comprender el lenguaje natural evitando la barrera de conocer el lenguaje de programación para tener una comunicación efectiva con el ordenador. Esta capacidad para procesar información de manera autónoma y generar respuestas apropiadas es lo que distingue a la IA y abre nuevas fronteras en la comunicación con los ordenadores y la red.

La IA ha encontrado aplicaciones particularmente valiosas en áreas donde la programación tradicional se queda corta. En el ámbito de la comunicación escrita, por ejemplo, las técnicas de procesamiento del lenguaje natural han permitido que las máquinas entiendan y generen texto de manera coherente y única. En la detección y reconocimiento de imágenes, los sistemas de IA pueden identificar y clasificar componentes visuales en una variedad de contextos y condiciones. En todas estas aplicaciones la ingeniería está detrás del desarrollo.

En la ingeniería, la aplicación de la inteligencia artificial presenta un posible avance sin precedentes. La capacidad de la IA para analizar grandes volúmenes de datos y modelar sistemas complejos ha llevado a mejoras notables en diseño, automatización, diagnósticos y solución de problemas en diversas especialidades. Desde la ingeniería civil que emplea algoritmos predictivos para el mantenimiento de infraestructuras hasta la ingeniería biomédica que utiliza la IA para el diagnóstico médico y personalización de tratamientos, la flexibilidad y eficacia de la inteligencia artificial están rediseñando el futuro del mundo que hoy se conoce.

Este trabajo explora la intersección de la inteligencia artificial con la ingeniería, diseñando el camino desde su concepción teórica hasta su implementación práctica en la ingeniería.

2

MARCO TEORICO

En este apartado se van a estudiar teóricamente la construcción y arquitectura de los modelos NLP. El objetivo principal de conocer la arquitectura que sustenta a estos modelos es conocer las limitaciones y fortalezas intrínsecas por su arquitectura y diseño.

2.1 COMPRENSIÓN DEL LENGUAJE NATURAL

La principal conjetura que se encuentran los algoritmos es comprender el lenguaje natural de los seres humanos. Al vocabulario, los humanos le han otorgado un sentido a cada una de las palabras para ser capaces de comunicar de manera efectiva. La palabra “ingeniería” tiene un significado en el lenguaje castellano que es posible traducir a otros idiomas, por lo tanto, los humanos han llegado al consenso de otorgarle ese significado a ese conjunto de letras.

Por desgracia, para un ordenador "ingeniería" se lee como un conjunto números en código ASCII, en este caso sería: 105 110 103 101 110 105 101 114 105 97.

Esto no guarda ningún significado específico, convirtiendo cualquier tipo de texto en un conjunto de números. Somos nosotros, los humanos los que hemos de entrenar a la inteligencia artificial para que sea capaz de comprenderlo tal y como se explica en este trabajo.

Para conseguir que el algoritmo sea capaz de comprender la información que se está aportando a través de palabras primero se lleva a cabo un proceso de tokenización.

2.2 EMBEDDING Y TOKENIZACIÓN

El proceso de embedding y tokenización consiste en dar al algoritmo un gran conjunto de datos (palabras) para que este sea capaz de identificar las similitudes. Siguiendo con el ejemplo de la palabra ingeniería, si el algoritmo encontrase el conjunto de números que representa la palabra "ingeniería" en repetidas ocasiones guardará ese conjunto como un token al que le asignará un número identitario. Estos tokens no tienen por qué ser una palabra específicamente, existiendo tres tipos de tokens, palabras, sub-palabras o carácter.

Con la tokenización se consigue reducir el número de datos con los que tiene que trabajar al algoritmo, se pasa de 105 110 103 101 110 105 101 114 105 97 (ingeniería) se convierte en 1294 (token para ingeniería) Los tokens son creados por cada una de las herramientas de inteligencia artificial y no tienen por qué guardar relación entre ellos. (GAMCO (n.d.) Clasification of tokens.)

A continuación, se lleva a cabo el proceso de clasificación de tokens. Este ejercicio consiste en etiquetar a los tokens según diferentes categorías. Un ejemplo sencillo puede ser el

siguiente, después de analizar muchos datos el algoritmo detecta que al token de ingeniería normalmente viene acompañado por el token "la" por lo que crea un marcador común para estas palabras. El token "la" también se relaciona con niña o con silla, se podría decir que el marcador que crea el algoritmo es a lo que el ser humano lo identificaría como femenino. Otro marcador relacionado con ingeniería puede ser el de los sustantivos. Identificará ingeniería como un sustantivo en base a la repetición de la misma estructura. El algoritmo crea numerosos marcadores distintos para cada uno de sus tokens, que le permite ser capaz de clasificar de alguna manera conjunto de números que antes de tokenizar carecían de sentido.

Al completar este proceso, se obtiene una "matriz" con los marcadores para todos los tokens, haciendo capaz al algoritmo de agrupar las palabras según similitudes. Estas palabras se pueden agrupar de maneras diferentes, y es posible crear una matriz n-dimensional con todos los tokens que el algoritmo ha generado. *(Word Embedding, 2022)*

A este método de clasificación se lo conoce cómo embedding. A través de esta agrupación y sistema, es posible realizar incluso operaciones matemáticas. Se puede calcular la distancia (similitud) entre palabras o incluso hacer sumas y restas. Un ejemplo sería, sumar al token de príncipe el token femenino "la" y se obtendría princesa.

El siguiente nivel de embedding que usan los algoritmos más avanzados de modelos NLP y que ayuda a la comprensión del mensaje natural consiste en llegar al siguiente nivel y tokenizar frases, repitiendo el mismo proceso que se llevaba a cabo con las palabras. Con el objetivo de seguir reduciendo en número de datos con los que trabajar en el futuro. En la siguiente figura se puede observar la clasificación n-dimensional que hacen los algoritmos de sus tokens.

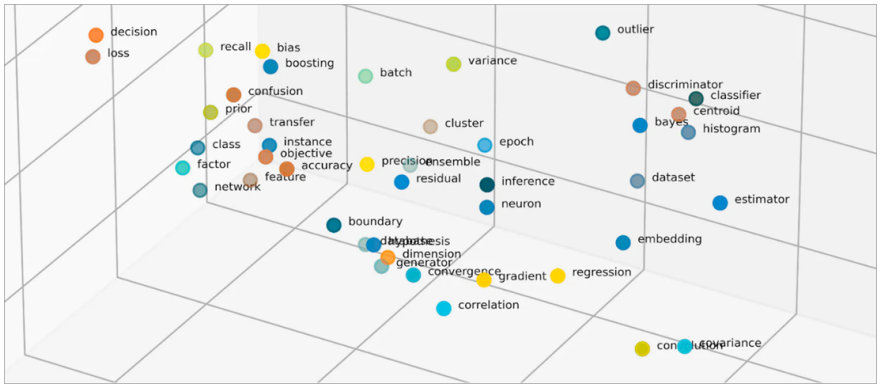

Figura 1: Ejemplo de embedding en 3D

2.3 LEMATIZACIÓN

En este momento, el algoritmo NLP tiene grandes cantidades de información sobre sus tókenes y cómo se relacionan entre sí. Sin embargo, en estos tókenes hay mucha información que no aporta información sustancial. Es decir, en el token "la ingeniería se enseña en ICAI" las palabras "la, se, en" no aportan información relevante a la oración. Es posible entender el mensaje de la oración sin estas palabras "ingeniería enseña ICAI" mantiene el significado. Y "ingeniería enseñar ICAI" es la forma más elemental de esta frase.
A este proceso de simplificar las frases se lo conoce como lematización, y permite reducir el número de parámetros que se almacenan en cada token. Esta técnica mejora la relación entre cada uno de los tokens, hace más sencillo encontrar similitudes entre los mismos.

Una vez se ha conseguido generar la base de pruebas, llevar a cabo el proceso de embedding y lematización. Es posible generar texto único con esta información de manera sencilla, ya que la frase está en su forma más simplificada, al reescribirla con las normas gramaticales de cada idioma es poco probable que la oración sea igual que la oración principal, partir de "ingeniería enseña ICAI" el algoritmo generaría una frase nueva que sería parecida a "la ingeniería es enseñada por ICAI" y diferente a su frase inicial. Este método es muy positivo y emula el proceso que llevan a cabo los humanos. Si el modelo buscase seguir aportando

información o completando las frases, se utiliza una función probabilística que pasa del token en el que se encuentra a uno "cercano" (con información parecida).

Una vez es posible generar textos únicos, el siguiente problema al que se enfrentan los modelos NLP es comprender que se le está pidiendo que generen. Para que los modelos sean útiles tienen que comprender cual es el objetivo de lo que se está requiriendo. Si son capaces de tokenizar la información y comparar con la base de pruebas, pero es posible comprender el significado anterior en una conversación en lenguaje natural, cuando el contexto de la información puede ser cualquier palabra. Esta solución se creó gracias a los algoritmos de inteligencia artificial, que buscaban retener el máximo significado de la oración posible. A lo largo del máximo número de oraciones posibles, idealmente infinitas.

2.4 EVOLUCIÓN ALGORTIMOS DE IA

El desarrollo de la inteligencia artifical ha evolucionado hasta el punto en el que se encuentra actualmente. Se detallara a continuación los algoritmos que han evolucionado hasta el desarrollo evolutivo que se ha conseguido en el presente.

2.4.1 REDES NEURONALES RECURRENTES (RNN)

(Theisen & Pekaric, 2021) El primer método que se inventó fueron las redes neuronales recurrentes. Se inventaron en 1986 gracias a David Rumehalt. Estas son conocidas por su capacidad de obtener información de datos secuenciales. Es posible entender una frase como un conjunto de datos secuenciales con distinta relevancia en cada punto de la secuencia.

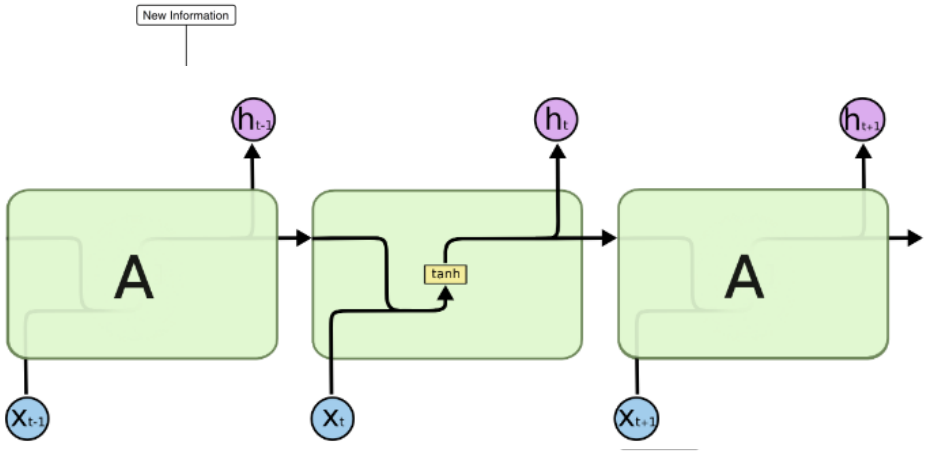

Figura 2: Diagrama funcionamiento redes neuronales recurrentes

El concepto de su funcionamiento es una retroalimentación de la información anterior a la palabra siguiente. Se guarda parte del significado y se utiliza para evaluar las propiedades del dato actual con respecto a los datos de pasado inmediato. Es importante tener en cuenta que estas redes neuronales están limitadas a una entrada y una salida. El mayor problema de este tipo de algoritmos de procesamiento de texto es el concepto conocido como ·short-term memory". A medida que la red procesa información tiene problemas para recordar los términos más lejanos y es un proceso que decae exponencialmente.

Además, no es posible procesar información en paralelo, ya que cada uno de los procesos de RNN tendría distinta información almacenada y el resultado de cada uno de los procesos sería diferente. Para solucionar este problema nacen las redes neuronales de memoria a corto y largo plazo (LSTM).

2.4.2 REDES DE MEMORIA A CORTO Y LARGO PLAZO (LSTM) (LONG SHORT TERM MEMORY)

(Olah (2015)) Las redes LSTM fueron introducidas por Hochreiter & Schmidhuber en 1997. Fueron especialmente diseñadas para acabar con el problema de la "short term memory" que tenían los algoritmos de RNN.

Las cadenas LTSM también cuentan con una estructura de cadena tal y como lo hacen las RNN, aumentan la complejidad del proceso de retroalimentación de información.

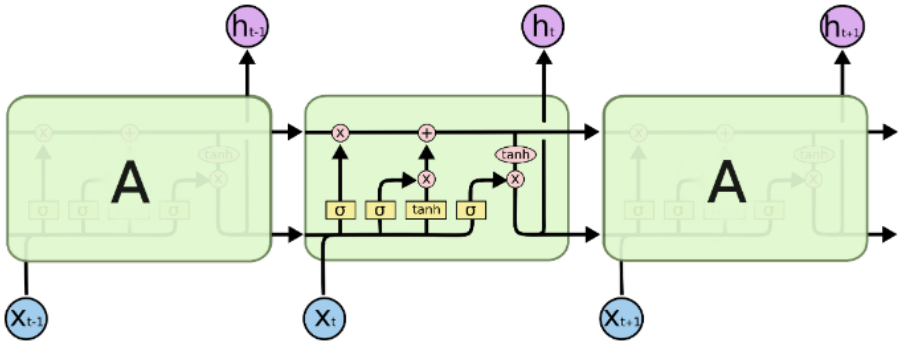

Figura 3: Diagrama funcionamiento LTSM

La clave del modelo es la línea horizontal en la parte superior, esta es la que transporta la información previa y va a través del modelo transportando la información prácticamente inalterada. Este modelo tiene la capacidad de añadir información a esta "cinta transportadora" en puntos concretos, esto sucede en los círculos rosas.

En la parte de debajo del diagrama hay varias operaciones ocurriendo en cadena. La primera de las operaciones se denomina "forget gate layer" y se encarga de decidir qué información eliminar.

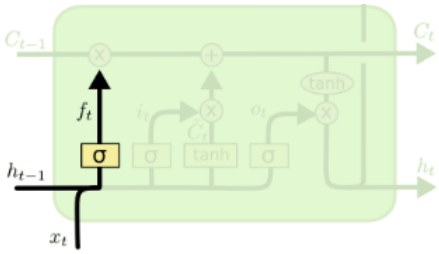

Figura 4: Diagrama "forget gate layer"

El siguiente paso consiste en decidir cuál es la información que se va a almacenar en la célula y que se transportará en la "cinta transportadora". A este paso se le conoce como "input gate layer".

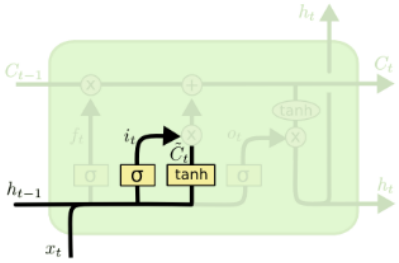

Figura 5: Diagrama "input gate layer"

Por último, se actualiza la información en la "cinta transportadora". En este paso ya se había decidido cuál es la información que se va a añadir y simplemente se hace el input de la misma. Simultáneamente, se decide cuáles van a ser los outputs del proceso.

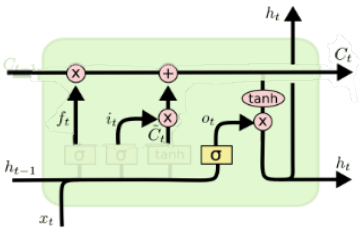

Figura 6: Diagrama "decisión" de añadir información

Este algoritmo, era capaz de solucionar el problema de la memoria a corto plazo, pero mantenía la limitación de una única entrada de texto, lo que limitaba absolutamente el ratio de aprendizaje de las herramientas de inteligencia artificial, haciendo que en ningún caso fuesen suficientemente buenas como para conseguir un hueco en el mercado.

2.4.3 TRANSFORMERS

Transformers es el algoritmo que ha servido como punto de inflexión en la capacidad de comprender el lenguaje natural. El algoritmo fue publicado en 2017 por unos desarrolladores de Google en un "paper" denominado "Attention is all you need", en un principio tenía el objetivo mejorar la traducción texto. Rápidamente se descubrieron las fortalezas y la capacidad de inferencia hacía los modelos NLP. Este algoritmo resuelve todas las limitaciones que tenían los algoritmos de redes recurrentes; es capaz de procesar el texto sin perder memoria y procesa el texto en paralelo. Este segundo punto ha sido lo que ha permitido que explote el desarrollo de algoritmos NLP, procesar texto en paralelo resuelve el cuello de botella que tenían estos algoritmos previamente, permitiendo maximizar la capacidad de procesamiento del texto introducido y moviendo el cuello de botella a la capacidad computacional máxima del ordenador. Permitiendo que el problema sea solucionable a base de aumentar la capacidad computacional disponible.

Este algoritmo es lo que da vida a todos los modelos de inteligencia artificial que han surgido en los últimos años. Desde ChatGPT, algoritmos para comprender el genoma humano y otros muchos. Gracias a su capacidad de contextualizar y no perder información sin importar la longitud de la cadena. Es importante comprender desde un punto de vista de ingeniería cómo está construido para explorar las limitaciones de base que puede tener en cuanto a sus futuras aplicaciones y desarrollos.

El algoritmo consiste en una capa de codificación y una de decodificación. La información entra por la capa de codificación, pasa a la capa de decodificación y sale. Ambas capas son idénticas, excepto que a la capa de decodificación a la que se le añade otra capa para codificar parte de la salida de la capa de codificación.

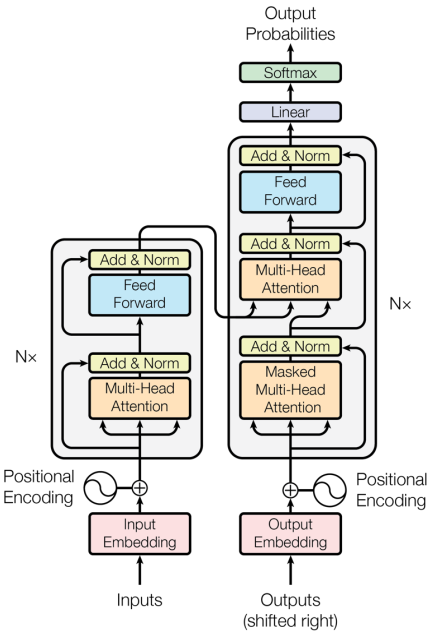

Figura 7: Esquema arquitectura algoritmo transformers

Esta es la estructura que tiene el algoritmo, se puede ver que ya no es simplemente una cadena, se ha duplicado. En el lado izquierdo la parte encargada de la codificación y en el lado derecho la parte encargada de la decodificación. Este aumento de complejidad respecto a los modelos anteriores habilita al algoritmo a procesar y ser capaz de comprender inputs de diverso tipo eficientemente. Con un entrenamiento especifico puede realizar tareas similares a las de un humano. Por este motivo, el enfoque que le dan las inteligencias artificiales genéricas como las que se van a estudiar en este proyecto es entrenarlos con el mayor número de inputs al alcance para que sea capaz de hacer eficientemente el máximo

número de tareas que el usuario pueda requerir. El cambio de concepto que supone este método, está compuesto a su vez por reinvenciones en la forma de plantear los algoritmos de RNN.

2.4.4 POSITIONAL ENCODING

(Transformer architecture: The positional encoding, 2019) En el lenguaje humano la posición de las palabras es lo que otorga sentido a una oración, lo que convierte el orden en algo primordial para comprender lo que a través del lenguaje se está intentando expresar. Las redes RNN para mantener el significado se veían obligadas a procesar palabra por palabra en el orden que se introduce la frase. Sin embargo, con este sistema el algoritmo es capaz de procesar la frase al completo. Dado que la posición que ocupa una palabra es altamente relevante, lo que hace este algoritmo es introducir esa información añadiéndosela al vector codificado de la palabra que ha procesado. Introducir la información posicional se basa en numerar la posición de una palabra dentro de una oración en número binario, el resultado de la operación no es un número, si no un vector d-dimensional; que se le añade al embedding de cada una de las palabras procesadas por el algoritmo.
Los números binarios tienen una propiedad de alternancia entre las columnas de 1 y 0. Si se contase de 1 a 20, la primera columna alterna 1 y 0 recurrentemente, la segunda columna alterna dos ceros consecutivos con dos unos consecutivos, la tercera columna alterna cuatro unos y ceros. El patrón es una alternancia de 2^{n-1} de unos y ceros.

Decimal	Binario			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Tabla 1: Ejemplo notación binaria

La interpretación de un número binario, rápidamente se puede sustituir por una función senoidal ya que consiste esencialmente en una onda cuadrada. Los creadores tomaron esta idea y crearon el método. Para el algoritmo Transformers la onda que se utiliza es esta:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Fórmula 1: Formula onda algoritmo transformers

PE se refiere a "positional encoding", pos se refiere a la posición de la palabra en la frase, d_{model} a la dimensión de la frase que se esté estudiando, sirve para adimensionalizar la función. El vector resultaría en una columna y n filas.

$$\overrightarrow{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

Fórmula 2: Vector resultante transformers

Según los autores. *(traducido al español)* "Se ha elegido esta función porque suponíamos que permitiría al modelo aprender fácilmente a atender a las posiciones relativas, ya que, para cualquier desplazamiento fijo en k, se puede representar como una función lineal de " Es decir, que el uso de las funciones senoidales permite además calcular la posición relativa entre las palabras dentro de una oración.

2.4.5 ATTENTION

"Attention" es el nombre del concepto que engloba el proceso del algoritmo de Transformers. Este algoritmo está compuesto por numerosos engranajes y conceptos que forman el algoritmo general. Profundizando en "Attention", se centra en hacer que el algoritmo comprenda el significado de la oración que se le ha introducido, puesto que se no se introducen las frases en orden, este es el sistema para comprender a que se refieren las palabras dentro de la oración. Un ejemplo sería "El alumno no completó el examen por falta

de tiempo", es relevante saber quién no completo, que no completo, por qué no completo el examen; todo esto se logra gracias a la solución de "attention".

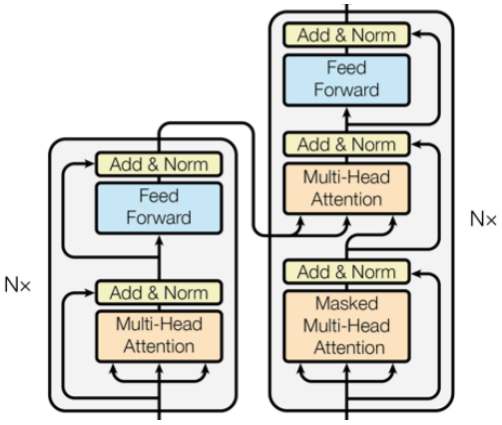

Figura 8: Capa de atención en la arquitectura Transformers

En primer lugar, consiste en seis componentes de codificador y seis componentes de decodificador, el primero convierte el texto a embedding y el segundo lo vuelve a convertir a texto con las consecuentes tareas que se le exijan al programa. En el caso de la publicación original, se utiliza este algoritmo para la traducción de texto, para los ejemplos de esta explicación se van a usar ejemplos de traducción de texto dado que son muy explicativos.

Cada uno de los componentes del codificador, esta subdividido en dos partes, una capa de "atención propia" y una red neuronal de avance que transmite la información al siguiente codificador, esta capa es siempre la misma en todas las etapas. El decodificador funciona de la misma forma con la diferencia que entre las dos capas tiene incluida otra capa de atención para centrarse en las partes relevantes de la oración de entrada. El esquema es el siguiente:

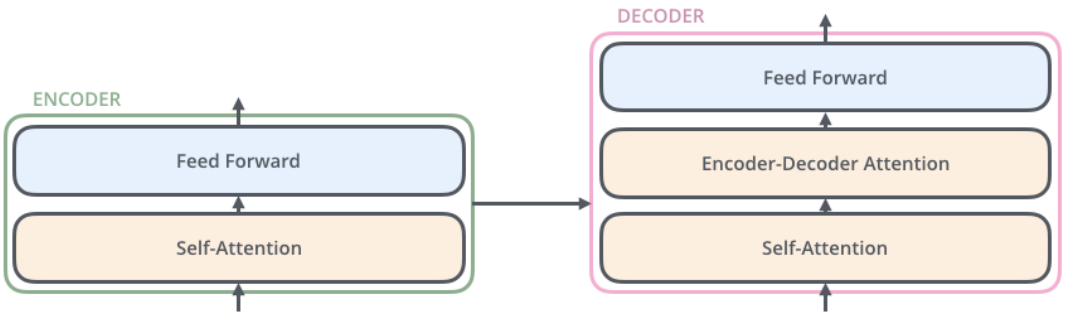

Figura 9: Esquema de la capa de atención

Para que este algoritmo funcione es necesario convertir el texto en *embeddings,* este proceso se realiza en la primera capa de codificación, la característica principal de los modelos Transformers, es que cada palabra fluye por su propio camino en el codificador, ya vienen con su posición marcada de la capa de *positional encoding*.

2.4.6 CAPA DE ATENCIÓN PROPIA

Esta es la capa que permite que el modelo Transformers sea tan potente, es un concepto que fue introducido por primera vez en el paper de "attention is all you need". Consiste en comparar cada una de las palabras y su importancia relativa respecto a las del resto de la oración.

Es importante recordar que en el proceso de embedding se guarda la información de cada una de las palabras en un mapa de n-dimensiones, de esta forma se le otorga un vector de n-dimensiones a cada una de las palabras, esto permite la posibilidad de hacer operaciones matemáticas con las palabras y permite ver las relaciones entre las palabras por la distancia que tienen en el espacio vectorial que se encuentran.

La función de esta capa es comparar cada una de las palabras de una oración con respecto a las demás para detectar relaciones y ponderar la importancia relativa de cada palabra respecto a la oración.

Para conseguir esta comparación sigue un proceso algo parecido al de la fuerza bruta. Lo hace a base de medir la distancia (producto escalar) entre la palabra elegida y todas las demás, y le otorga un valor a esa relación.

Concretamente, el primer paso es crear tres vectores a partir de cada uno de los embeddings de cada palaba. Lo vectores se llaman **Query, Key** y **Value.** Estos términos son comunes, una analogía es que el vector Query actúa como una "cerradura" y el vector key como "llave" y se van a realizar comparaciones basadas en la similitud de ambos.

En segundo lugar, se le va a otorgar una puntuación a cada una de las palabras con respecto a las demás, palabra a palabra se va comparando el vector Key contra los Query del resto de palabras de la oración. Este valor se calcula con el producto escalar entre el vector Query y el Key, resultando en un valor que representa la similitud o importancia relativa de las palabras y se convierte a módulo (dividir entre el módulo del tamaño del vector Key). Es importante recordar que la información de los tokens esta almacenada en una matriz n-dimensional, por lo que a través del producto escalar se puede medir la distancia dentro de esa matriz y obtener la similitud relativa como una distancia entre dos tokens.

Por último, a ese valor se le aplica la función ***softmax*** cuya función es convertir un vector de K variables en una función de probabilidad. A continuación, se multiplica el valor que devuelve la función softmax por el valor del módulo; el concepto detrás de esta idea es mantener la relevancia en las palabras más importantes y dejar de lado las menos relevantes para poder mantener el significado a lo largo de grandes cantidades de contenido.

El paper refina aún más la idea de la atención con un refuerzo que denominan "multi-headed attention". Esta técnica se utiliza para mejorar el rendimiento de dos maneras:

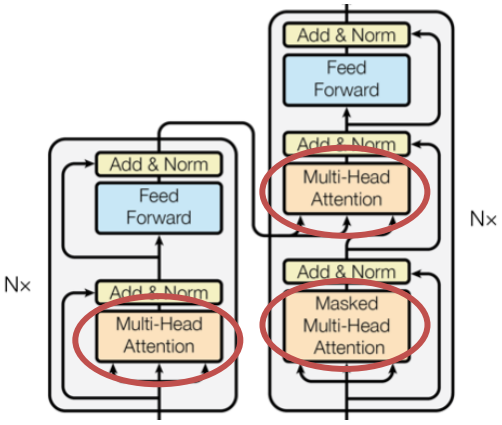

Figura 10: Esquema capa atención con foco en la capa de atención propia

Amplia la habilidad del modelo para concentrarse en múltiples posiciones. Es decir, en una frase larga como "El ingeniero no fue capaz de terminar su trabajo porque se encontraba lejos de su puesto de trabajo." Permite rápidamente encontrar las relaciones de "se" y "su" con "el ingeniero", mejorando la comprensión general del algoritmo.

Le permite al algoritmo mantener múltiples "subespacios de representación". Esto se refiere a que el modelo es capaz de interpretar la oración de maneras diferentes. En el modelo Transformers, existen 8 subespacios desde los cuales se recaba información respecto de la oración, permitiendo obtener una comprensión más rica y completa.

Recapitulando sobre este algoritmo Transformer, es relevante comprender como está programado y cuáles son los procesos lógicos que sigue el algoritmo para poder comprender y preveer el alcance futuro de esta tecnología. El siguiente gráfico proporciona un resumen sencillo del proceso del modelo Transformers. *(Alammar, 2018)*

1. Se recibe una frase de entrada en el modelo.

2. Se incorpora cada palabra de la frase como un elemento individual para su procesamiento.

3. El sistema divide estos elementos en 8 grupos o "cabezas". Cada grupo se multiplica por matrices de peso específicas, identificadas como X o R.

4. Para calcular la atención, se utilizan las matrices resultantes, denominadas Q (Query), K (Key) y V (Value).

5. Las matrices Z obtenidas de este proceso se concatenan y, posteriormente, se multiplican por una matriz de peso final, W^0, para generar la salida de la capa.

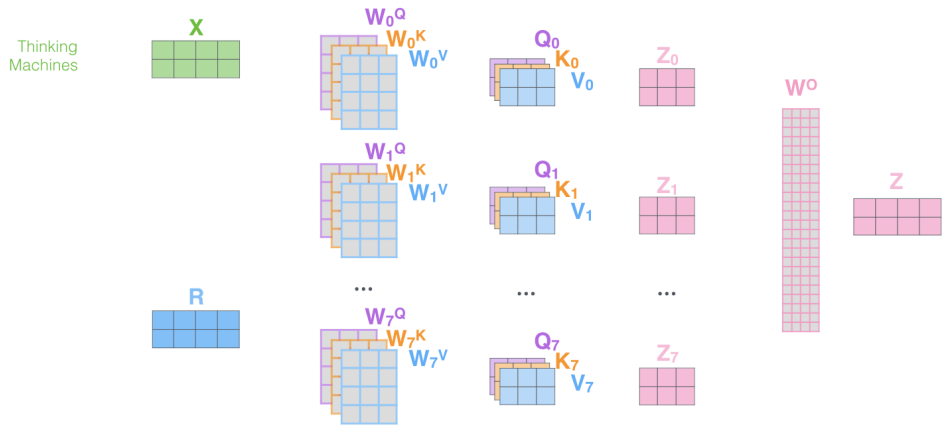

Figura 11: Esquema proceso lematización algoritmo Transformers

2.5 ENTRENAMIENTO DE IA

2.5.1 TAMAÑO DEL CONJUNTO DE DATOS

El entrenamiento de modelos de inteligencia artificial se fundamenta en la absorción y procesamiento de extensos conjuntos de datos. Esta práctica busca construir una amplia base de conocimiento que permita al sistema inferir y responder de manera competente a las solicitudes de los usuarios. Tradicionalmente, este proceso resultaba excesivamente costoso en términos de tiempo y recursos computacionales, limitando significativamente la complejidad y la eficacia de los modelos resultantes.

Con la concepción de los modelos Transformers, esta situación ha experimentado un giro importante. Los modelos actuales se benefician de una reducción considerable en los tiempos de ejecución y de una mayor capacidad de procesamiento, gracias a la eficiencia inherente al nuevo diseño de la arquitectura de los Transformers.

Los datos son el pilar fundamental en el desarrollo de estos modelos, alimentan de información y mejoran la calidad de la respuesta. Siendo uno de los mayores desafíos encontrar bases de pruebas de calidad y representativos. En el caso de los modelos genéricos que se tratan en este trabajo requieren de una cantidad de datos inmensa.
En esta figura se compara la exactitud de la respuesta según la cantidad de datos de entrenamiento del modelo. Se observa la influencia directa del tamaño de la muestra en la exactitud de la respuesta del modelo NLP. *(Mei et al., 2020)*

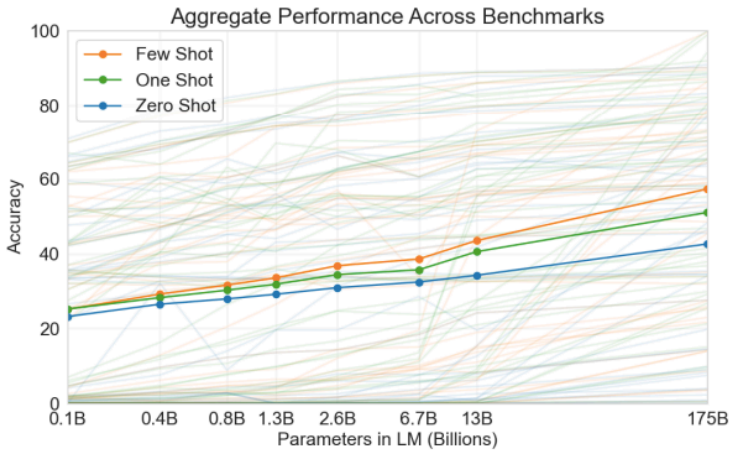

Figura 12: Rendimiento IA respecto al tamaño de la base de pruebas de entrenamiento

El tamaño de la base de pruebas para entrenar el modelo afecta positivamente a la exactitud de las respuestas. sin embargo, tiene un efecto inversamente proporcional en el tiempo de procesamiento de los datos, en la siguiente tabla se muestra el efecto del tamaño de la base de pruebas respecto al tiempo de procesamiento. Aunque, gracias al modelo Transformers que permite que se procesen datos en paralelo, el tiempo de procesamiento se puede reducir aumentando la capacidad de procesamiento (aumentando la potencia computacional). De hecho, ChatGPT fue entrenado utilizando la nube Azure de Microsoft.

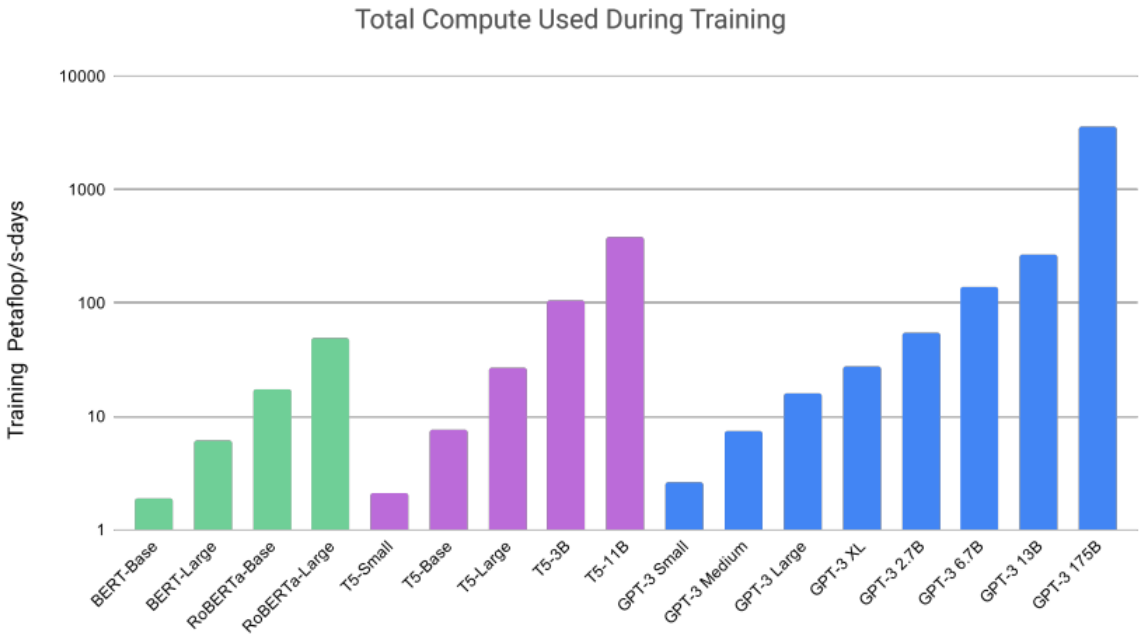

Figura 13: Tiempo de entrenamiento de los modelos de IA respecto al algoritmo de IA

(Mei et al., 2020) En el caso concreto de ChatGPT, el tamaño de la muestra de datos de entrenamiento ha ido incrementando con las actualizaciones. Pasando de 117 millones de parámetros utilizados en ChatGPT1 a un estimado de un 1 trillón (10^{12}) de parámetros para el modelo más reciente ChatGPT4. No obstante, la cantidad de parámetros no es el único parámetro relevante en el entrenamiento de los modelos de inteligencia artificial, la fuente de los datos representa un gran porcentaje en relevancia respecto a la respuesta. Es importante que los datos sean variados en temas, exactos y abundantes. La empresa OpenAI ha entrenado a su modelo mayormente en estas fuentes de información.

2.5.2 FUENTE DE LOS DATOS DE ENTRENAMIENTO

(Mei et al., 2020) Common crawl es un repositorio de datos que rastrea la web y trata de combinar toda la información de la WWW (world wide web) en un mismo lugar para democratizar el acceso a la información. Esta es la principal fuente de la que extrae conocimiento ChatGPT y sirve para mostrar el espíritu del entrenamiento de una inteligencia artificial. En vista de que tiene que emular la capacidad de un humano, es necesario que acceda a la mayor variedad de datos en abundancia, y esta solución es la web.

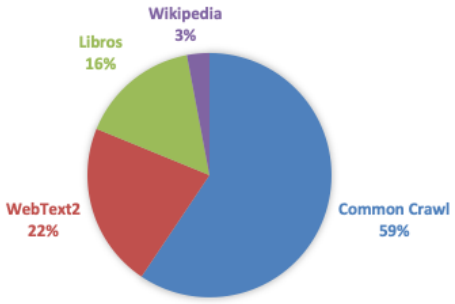

Figura 14: Fuente de los datos de entrenamiento del ChatGPT

El segundo gran porcentaje de los datos sobre los que se ha entrenado ChatGPT es WebText2, esta fuente de datos recopila todos los posts de Reddit de 2005 a 2020. Reddit es un foro dónde usuarios interactúan y opinan sobre prácticamente todos los temas que existen.

Por último, las librerías Books 1 y 2, contenidas dentro del apartado libros en la figura, están formadas por una gran muestra de los libros de dominio público publicados en el mundo, junto con Wikipedia en inglés forman el conocimiento completo de la inteligencia artificial de OpenAI, ChatGPT.

El conjunto de datos expuesto es de lo que se nutre ChatGPT para generar todas las respuestas, es un conjunto de datos extenso y con el espíritu de recoger el conocimiento de la humanidad en una gran medida. No obstante, es relevante destacar que una gran parte de este conocimiento no es producido de manera supervisada y rigurosa, si no que esta producido por usuarios de la web. Supone una amenaza respecto a la relevancia y exactitud de las respuestas que concede esta inteligencia artificial.

2.5.3 ENTRENAMIENTO Y PROCESAMIENTO DE LOS DATOS

Sobre el proceso de entrenamiento OpenAI desarrolla en primer lugar un proceso de filtrado y mejora de los datos que obtiene de las fuentes anteriores. A la base de pruebas commonCrawl se le aplican distintas técnicas para depurar los datos. En primer lugar, se filtran los datos basándose en un "corpus de referencia", esto son documentos e información validada de alta calidad como artículos académicos, literatura bien escrita o textos de calidad. A posteriori, llevaron a cabo una de-duplicación (técnica que consiste en optimizar los datos para eliminar duplicidades de los datos) difusa de los datos, que consiste en eliminar duplicados en conjuntos de datos que son similares entre sí. Se utiliza para evitar el sobreajuste. Por último, se añadieron "corpus de referencia" conocidos por su alta calidad.

En el momento que el conjunto de datos es el indicado, comienza el proceso de entrenamiento. La primera fase, denominada preentrenamiento generativo, se enfoca en la asimilación de todo tipo de información en la mayor cantidad de temas posible. Seguidamente, la etapa de ajuste fino supervisado (Supervised Fine-Tuning o SFT) refina las capacidades del modelo mediante la exposición a ejemplos específicos y retroalimentación directa. La fase final del entrenamiento involucra técnicas de aprendizaje

reforzado, donde el modelo optimiza sus respuestas a través de un sistema de recompensas basado en la calidad de las interacciones y el cumplimiento de objetivos predeterminados. El entrenamiento del corpus de SFT se lleva a cabo utilizando el modelo GPT base y un algoritmo conocido como Descenso de Gradiente Estocástico (SGD), este podría ser comparado con un entrenador que instruye a un ordenador a mejorar en un juego mediante consejos y correcciones iterativas. Esta optimización se efectúa ajustando continuamente los parámetros del modelo hasta minimizar la función de costo. Durante este proceso, en cada paso, el algoritmo selecciona aleatoriamente un subconjunto del conjunto de datos de entrenamiento para calcular el gradiente de la función de costo en relación con los parámetros, y con base en ello, realiza la actualización correspondiente.

A lo largo de la etapa de SFT, los parámetros del modelo base de ChatGPT se actualizan para incorporar información específica de la tarea que no estaba presente antes del SFT. Sin embargo, incluso después de este intenso proceso de SFT, el modelo ChatGPT todavía enfrenta un fenómeno conocido como "cambio distribucional", el cual se refiere a las diferencias entre los datos de entrenamiento y las situaciones del mundo real con las que el modelo podría encontrarse una vez salga a la luz.

Para mejorar la capacidad reactiva del modelo en conversaciones reales, se implementa la tercera etapa de aprendizaje con Reforzamiento a través de Retroalimentación Humana (RLHF), donde el modelo se entrena para actuar proactivamente, ajustándose a nuevas entradas y situaciones que no estaban presentes en los datos de entrenamiento originales. Este enfoque permite que el modelo no solo reaccione pasivamente con respuestas aprendidas, sino que también desarrolle estrategias para manejar solicitudes inéditas de manera efectiva. Este tipo de entrenamiento es el más costoso, ya que requiere de personas físicas enseñando al modelo a cómo interpretar y responder correctamente a las peticiones humanas.

La forma de comunicación con el modelo y de entrenamiento es parecida en su última fase, las respuestas dependen del input que recibe. De modo que existe una disciplina alrededor del arte de comunicarte con el modelo, a esta disciplina se la conoce como prompting.

Existe un riesgo con los lenguajes de procesamiento natural que se conoce como alucinación de los sistemas. Esta consiste en información devuelta por el sistema escrita de manera coherente, está compuesta de datos incorrectos, erróneos o sesgados. Esto sucede de base por la arquitectura transformers de la que están compuestos puede tener errores en la fuente de extracción de los datos y que a pesar de mantenerse impecable en la redacción y simular que la respuesta es correcta tienen un error de base alto y puede engañar a muchos usuarios. Supone un riesgo dado que a medida que la confianza general aumenta en estos lenguajes, pueden suponer una gran fuente de desinformación. Además, a medida que avanza y se desarrolla la tecnología las respuestas que devuelve pueden parecer más fiables y aumentar aún más el riesgo de tomar como verdaderas respuestas erróneas de la inteligencia artificial.

2.6 PROMPTING

La definición de prompting según el diccionario de Cambridge es "el acto de intentar que alguien diga algo". *(Cambridge University Press, n.d.)* En el caso de la inteligencia artificial el prompting es la información que se le introduce al algoritmo para conseguir la mejor respuesta.

Se aprecian grandes cambios en resultados dependiendo del input que recibe el algoritmo. En el proceso de prueba de OpenAI, crearon esta tabla que compara la exactitud de la respuesta con el prompt.

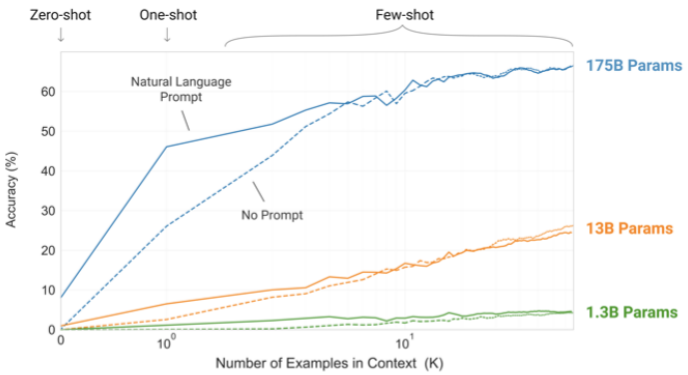

Figura 15: Diferencias de rendimiento según el tamaño de la base de pruebas de entrenamiento y la cantidad de prompts

De este gráfico se pueden deducir dos cosas, en primer lugar, siempre es mejor utilizar un prompt. Sin importar el número de parámetros con el que se ha entrenado el modelo, en todos los casos el prompt funciona mejor y devuelve una respuesta más exacta. El incremento marginal en la exactitud de la respuesta decrece hasta llegar a ser prácticamente indiferente cuando K tiende a infinito.

En segundo lugar, la exactitud mejora notablemente con el número de ejemplos (K) que se le introducen al modelo y el número de interacciones que se tienen con el mismo para obtener mayor exactitud de respuesta. Ahora bien, en el uso diario y de aplicación para la ingeniería, es impráctico introducir más de 10 ejemplos para cada una de las interacciones que se mantengan con el algoritmo. El incremento marginal de utilizar la herramienta del prompting correctamente supone un incremento en la eficiencia en el comportamiento con el modelo.

2.6.2 TIPOS DE PROMPTING

OpenAI diferencia entre tres tipos de prompting Zero-Shot, One-Shot y Few-Shot. Se toma como referencia papers de esta empresa ya que ha sido pionera en el las herramientas de inteligencia artificial y tenían una filosofía de apertura de la tecnología al público por lo que compartían mucha información relevante a este proyecto

El "Few-Shot Learning" (FS) es un enfoque donde al modelo se le proporcionan unas pocas demostraciones de la tarea durante el tiempo de inferencia sin permitir actualizaciones en sus parámetros (K). Se le entregan al modelo K ejemplos entre 10 y 100 de un contexto y la conclusión correspondiente, y luego se le presenta un nuevo contexto para que el modelo genere la conclusión esperada. Las ventajas de este enfoque incluyen una reducción significativa en la necesidad de datos específicos para la tarea. Sin embargo, los resultados obtenidos hasta ahora con few-shot suelen ser inferiores a los de modelos ajustados específicamente para tareas concretas.

El "One-Shot Learning" (1S) es similar al few-shot pero solo se permite una demostración única, acompañada de una descripción en lenguaje natural de la tarea. Esta distinción es relevante porque coincide con la forma en que algunas tareas se comunican a los humanos, dónde se les entrega una petición y un ejemplo para trabajar a modo de base.

El "Zero-Shot Learning" (0S) es similar al "One-Shot Learning", con la diferencia de que no se permiten demostraciones; en su lugar, el modelo recibe únicamente instrucciones en

lenguaje natural que describen la tarea. Representa el escenario más complejo para el modelo.

Para este proyecto es extremadamente relevante estandarizar el proceso del prompting, ya que puede variar la exactitud de las respuestas excesivamente según la calidad del input. Además de las diferentes pruebas, el prompt son una serie de instrucciones en lenguaje natural humano, existen numerosas variables a tener en cuenta para elaborar el mensaje. OpenAI expone las claves que consideran que conforman el mejor prompt, en base a eso se elaboraran unas instrucciones estándares para las diferentes inteligencias artificiales.

Para generar un buen prompt hay que tener en cuenta numerosos factores. El concepto general detrás de esta ciencia es expresar muy concretamente lo que quieres para obtener la respuesta más exacta a tus necesidades. Según OpenAI, estos son los factores a tener en cuenta: *(Chen & Li, 2023)*

2.6.3 CLAVES PARA GENERAR UN BUEN PROMPT

- **Entender el objetivo y resultado deseado**: Definir claramente el objetivo y el resultado que se espera obtener y formular preguntas que se alineen con las expectativas.
- **Conocer las fortalezas y limitaciones de ChatGPT**: Familiarizarse con lo que ChatGPT puede y no puede hacer asiste en el diseño de preguntas.
- **Especificar la extensión de la respuesta**: Si se van a tratar temas de especialización, usar vocabulario o contextos específicos del tema puede guiar al modelo hacia una respuestas más precisa y relevante. Proporcionar contexto adicional y ejemplos es útil.
- **Claridad y especificidad en las solicitudes**: Asegurar que las preguntas sean claras y específicas para evitar ambigüedades o confusiones que puedan resultar en

respuestas no óptimas. La ambigüedad puede surgir de instrucciones poco claras, preguntas vagas, o un contexto insuficiente.

- **Establecer restricciones**: Determinar si existen restricciones necesarias, como la longitud de la respuesta o su formato, para alcanzar el resultado deseado. Especificar estas restricciones explícitamente orienta al modelo hacia la generación de respuestas que cumplan con requisitos específicos, como límites de caracteres o formatos estructurados.

3

METODOLOGÍA

La metodología de este trabajo se estructura en varias fases clave para llevar a cabo un análisis y comparativa de las diferentes inteligencias artificiales en su aplicación a los roles de la ingeniería.

Primero, se ha establecido el marco teórico que sienta las bases para el análisis. La ingeniería es una disciplina muy amplia, por lo que se optará por un enfoque superficial y generalizable a cualquier ingeniero que busque sistematizar procesos con inteligencia artificial. A continuación, se profundizará en la capacidad de las inteligencias artificiales para resolver

problemas de ingeniería, sometiéndolas a diferentes pruebas y creando una comparativa de su desempeño.

El estudio comparativo incluye las tres inteligencias artificiales más relevantes de la actualidad, mencionadas en la introducción: OpenAI (ChatGPT-3.5 y ChatGPT-4) y Google (Gemini). Se han elegido estas inteligencias porque disponen de la posibilidad del uso de API, que se explica más adelante y es vital para la realización de las pruebas. La metodología sigue un enfoque investigativo, comenzando con una revisión de los conceptos teóricos y herramientas del proyecto. Una información más detallada sobre estas herramientasa:

ChatGPT-3.5: La versión gratuita de OpenAI, ideal para tareas de conocimiento y razonamiento general, asistencia en redacción y respuesta a preguntas complejas sin costos adicionales.

ChatGPT-4: El modelo más avanzado de OpenAI, destacado por su capacidad en tareas de conocimiento general, razonamiento especializado y social y ético, ideal para aplicaciones que requieren alta precisión y sofisticación.

Gemini: Desarrollado por Google, con un rendimiento sólido en tareas de conocimiento y razonamiento, ofreciendo un equilibrio entre rendimiento y costo para aplicaciones con comprensión detallada y contextual.

Para el análisis, se implementarán pruebas estandarizadas en distintas ramas de la ingeniería para evaluar el comportamiento de estos modelos en escenarios similares a los que enfrentan los ingenieros en su vida cotidiana.

Estas pruebas consisten en utilizar bases de pruebas que contienen miles de preguntas con respuestas “correctas” sobre un tema. Estas bases de pruebas, normalmente se utilizan para entrenar a la inteligencia artificial, en este trabajo se van a utilizar estas mismas colecciones para a través de código hacer que las herramientas del estudio respondan a las preguntas

contenidas en las colecciones y comparar las respuestas, buscando un porcentaje de similitud.

Los resultados de estas pruebas serán registrados y analizados, finalizando con la elaboración de tablas de rendimiento que destacarán las fortalezas y debilidades de cada modelo, ofreciendo una visión clara de su eficiencia, precisión y adaptabilidad en contextos de problemas de ingeniería reales.

Con esta metodología, el trabajo busca proporcionar una comparativa rigurosa y detallada de las capacidades de las inteligencias artificiales en el ámbito de la ingeniería, facilitando una evaluación de su utilidad y rentabilidad para los profesionales del sector.

3.1 BASES DE PRUEBA

Para poner a prueba estos programas, existen numerosas bases de pruebas que se pueden utilizar para medir la capacidad de las inteligencias artificiales. Las bases de pruebas difieren en estilo y tipo; estas consisten en una larga serie de funciones de entrada que ponen a prueba a la inteligencia artificial en una tarea concreta. Además, incluyen la respuesta ideal para poder comparar y obtener un porcentaje de acierto. De esta forma, se compara la respuesta de la IA con la respuesta correcta y en un número alto de sucesos, es posible otorgar un porcentaje de acierto a la IA y en base a estos aciertos valorar y comparar las diferentes modelos.

Se han seleccionado específicamente bases de pruebas concretas para probar las capacidades en casos específicos de la ingeniería. Estas bases de pruebas han sido seleccionadas con el objetivo de tratar el mayor espectro de ramas posible. Se componen de las siguientes bases de pruebas:

- **STEM-AI** - Esta base de pruebas la proporciona, y contiene más de mil prompts de preguntas sobre la ingeniería eléctrica. Estas tienen un formato de función de entrada, con una instrucción clara seguido de contexto de la pregunta.

- **Python-codes –** Consiste en una base de pruebas que pone a prueba la capacidad de programación de las diferentes inteligencias artificiales. Siendo la programación y la ingeniería del software una de las ramas más presentes en el día a día y dónde las inteligencias artificiales funcionan cómo verdaderos expertos, ya que han sido entrenadas en una altísima cantidad de código. Es relevante para la investigación conocer el nivel de conocimiento en programación que estas tienen. Para ello, se va a utilizar una base de pruebas que consiste en tareas de programación. La base de pruebas sigue el mismo concepto que la anterior, en base a un prompt, la inteligencia artificial debe resolver y se cuenta con la respuesta correcta. Un ejemplo de Esta base de pruebas: *Give me a Python function that'll download a CSV file from a URL."* Para lo que el algoritmo de inteligencia artificial tiene que producir un código que resuelva este problema. En este caso, el prompt necesita que se le especifique que tiene que crear un código para resolver este problema. Esto se añade manualmente desde la programación, para intentar mantener un rigor en la introducción de datos a la herramienta

- **GHOSTS** - Consiste en una base de pruebas que está específicamente creada con preguntas matemáticas de nivel superior al grado. Las matemáticas son un concepto relevante dentro de la ingeniería y es extremadamente importante comparar las respuestas en este apartado. Un ejemplo del tipo de preguntas que se pueden encontrar dentro de esta base de pruebas es: *How many vertical asymptotes does the graph of $y=\\frac{2}{x^2+x-6}$ have?"*

- **Test de sentimiento –** Este es un test muy común que consiste en comprobar si la IA realmente entiende el input que se le está introduciendo. Consiste en hacer que la inteligencia artificial detecte el sentimiento (positivo/negativo/neutro) de una serie de frases. Este test se lleva a cabo ya que un ingeniero debe leer y comprender grandes cantidades de información para posteriormente poder aplicarlas en su día a día. Un ejemplo de esta base de datos es: *that`s great!! weee!! visitors!* Que tiene un sentimiento positivo.

3.2 INTRODUCCIÓN DE DATOS

Introducir las bases de pruebas en cualquier lenguaje de inteligencia artificial no se lleva a cabo a través de la barra de chat que se habilita para poder conversar. Si se hace de esta forma, la IA es incapaz de procesar la alta cantidad de datos que se le están introduciendo y ejecuta la tarea para un número reducido de datos. Están adaptadas a tareas sencillas en cuanto a procesamiento de datos, conservando de esta forma espacio computacional para ejecutar el resto de las peticiones de los otros usuarios. En el hipotético caso de que pudiesen llevar a cabo estas tareas, es probable que se saturase la red por las peticiones exigentes de todos los usuarios combinados.

Es decir, para poder comprobar las bases de pruebas se va a utilizar código en Python para "forzar" a estas herramientas a procesar todas las operaciones de la base de pruebas. A través de la programación es posible que las inteligencias artificiales se enfrenten a las bases de pruebas y obtener respuestas fidedignas con un estudio completo de las respuestas.

Por suerte, una de las fortalezas de los modelos de lenguaje natural es la programación. Los modelos han sido entrenados en inmensas cantidades de programación, siendo la programación una de las disciplinas de las que se disponen mayores cantidades de datos y ejemplos. Gracias a este entrenamiento son capaces de proporcionar código preciso basado en instrucciones en lenguaje natural. Se va a apalancar sobre esta propiedad para crear código

en Python, probar que funciona de manera correcta y utilizarlo posteriormente en el entorno local de Python y realizar las pruebas necesarias.

En primer lugar, para acceder desde un ordenador local o desde fuera de la red de las inteligencias se necesita de una API (Interfaz de Programación de Aplicaciones, las siglas vienen del inglés). Esta es un conjunto de reglas que permiten a los diferentes softwares comunicarse entre sí (Amazon Web Services(n.d)) Estas definen las formas en las que es posible comunicarse con otro software/aplicación, y proporcionan un modo estandarizado de solicitar y enviar datos entre los sistemas.

En el caso de las IA, permite que desde la red local de cualquier ordenador conectar con el software y desarrollar sin necesidad de interactuar con la interfaz de ninguna aplicación web.

Para instalar la API de cada herramienta es necesario un código similar pero diferente en algunos aspectos. Para acceder a las herramientas de ChatGPT el código que se necesita es el siguiente:

```
pip install openai
import openai
openai.api_key = 'Se mantiene privada'
```

Para Gemini el Código es parecido, con diferencia de las bibliotecas que se importan:

```
pip install google-cloud
export GOOGLE_APPLICATION_CREDENTIALS="Se mantiene privado”
import os
os.environ["API"] = "se mantiene privado"
```

```
from google.cloud import
```

Por privacidad las claves API no se muestran en este trabajo, ya que están vinculadas a una cuenta personal. Esta clave se consigue de manera gratuita al registrarte en las herramientas de inteligencia artificial, sin embargo, el servicio de su uso suele de ser de pago. Siguiendo una metodología de pago por uso de tokens. Según el número de tokens que devuelva el programa se calcula el precio. El número de tokens se puede ajustar a través de la programación.

Con este código ya es posible comunicarse efectivamente con las herramientas de inteligencia artificial.

3.3 CÓDIGO PARA LAS PRUEBAS

Para realizar el análisis exhaustivo del rendimiento de las inteligencias artificiales en tareas específicas aplicadas a la ingeniería, es básico implementar un conjunto de pruebas utilizando bases de pruebas especializadas en distintas herramientas de inteligencia artificial. Cada una de las bases de pruebas son diferentes en cuanto a cómo introducen los datos en el código, necesitan ciertos ajustes en la programación general.

3.3.1 STEM-AI

Esta base de pruebas, explicada previamente consiste en un repositorio de preguntas sobre la ingeniería eléctrica. Ya se ha indicado el tipo de pregunta que incluye la base de pruebas y a modo ejemplo, la respuesta que ChatGPT4 da a la pregunta dentro de la base *Eres un ingeniero eléctrico y vas a responder a preguntas relacionadas con la ingeniería eléctrica. ¿Qué es la ley de ohm?* Es la siguiente:

La Ley de Ohm es un principio fundamental en la ingeniería eléctrica que describe la relación entre el voltaje, la corriente y la resistencia en un circuito eléctrico. Generalmente se expresa con la fórmula: $V=I\cdot R$V=I·R Donde:

VV es el voltaje (diferencia de potencial) a través del circuito o componente, medido en voltios (V).

II es la corriente que fluye a través del circuito o componente, medida en amperios (A).

RR es la resistencia del circuito o componente, medida en ohmios (Ω).

La Ley de Ohm establece que la corriente a través de un conductor entre dos puntos es directamente proporcional al voltaje a través de los dos puntos e inversamente proporcional a la resistencia entre ellos. Esta relación permite el cálculo de cualquiera de las tres magnitudes (voltaje, corriente, resistencia) si se conocen las otras dos.

Claramente, ambas respuestas explican la ley de ohm y son similares en los conceptos. El problema radica en el porcentaje de similitud de las respuestas, para este tipo de respuestas que son amplias, es la propia herramienta de inteligencia artificial la que determina la similitud. En este caso, ChatGPT4 le ha dado un 87% de similitud.

Para determinar la similitud en casos de texto se va a utilizar el método TF-IDF (Term Frequency-Inverse Document Frequency). Este es el método que se va a utilizar para calcular la similitud entre la respuesta entendida cómo correcta por parte de la base de pruebas y la respuesta que devuelve la IA.

Método de comparación de texto (TF-IDF)

El método TF-IDF (Term Frequency-Inverse Document Frequency) es una técnica utilizada en recuperación de información y minería de texto para evaluar la importancia de una palabra en un documento respecto a una colección o corpus de documentos. Este método se utiliza exclusivamente para esta base de pruebas por la tipología de las respuestas. Ya que son respuestas de texto y cada una puede variar en longitud. En las futuras bases de datos no se adapta tan bien a las características específicas de las respuestas para comparar. Esta técnica combina dos medidas:

TF (Term Frequency): Mide la frecuencia de una palabra en un documento, en este caso en un párrafo. Específicamente, es el número de veces que una palabra aparece en un párrafo dividido por el número total de palabras en ese párrafo.

Dónde:

- **t** es el término sobre el que se realiza la comparación.
- $\boldsymbol{d}$ es el documento/párrafo.
- $\boldsymbol{f(t,d)}$ es la frecuencia del término tt en el documento dd.
- $\boldsymbol{Nd}$ es el número total de palabras en el documento dd.

IDF (Inverse Document Frequency): Mide la importancia de una palabra en el párrafo. Es decir, compara la relevancia de la palabra respecto del documento completo. La idea es, si una palabra se repite mucho, la relevancia de esta será menor. Como, por ejemplo, determinantes o conectores. Por lo que el IDF tenderá a 1 o menor y el IDF tenderá a 0. En cambio, palabras relevantes se repiten poco por lo que el IDF le otorgará un valor mayor.

Donde:

- $|D|$ es el número total de documentos en el corpus. En este caso siempre son 2, el generado por la inteligencia artificial y el de referencia

- $|\{d \in D : t \in d\}|$es el número de documentos que contienen el término t. Es decir, cuantas veces aparece la palabra de referencia en ambos documentos.

TF-IDF: Es el producto de TF e IDF y proporciona una medida de la importancia de una palabra en un documento considerando su frecuencia en el corpus.

$$TF\text{-}IDF(t,d,D) = TF(t,d) \times IDF(t,D)\ \text{TF-IDF(t,d,D)} = \text{TF(t,d)} \times \text{IDF(t,D)}$$

Por último, se utiliza el método de **similitud del coseno**. Es una métrica utilizada para medir la similitud entre dos vectores en un espacio de múltiples dimensiones. En el contexto del procesamiento de texto, los vectores son representaciones numéricas de documentos o frases basadas en TF-IDF. La similitud del coseno se calcula como el coseno del ángulo entre dos vectores, y su valor varía entre -1 y 1, donde:

- 1 indica que los vectores son idénticos (es decir, están en la misma dirección).

- 0 indica que los vectores son ortogonales (sin relación).

- -1 indica que los vectores son opuestos (están en direcciones opuestas).

La fórmula es la siguiente:

Siendo A y B los vectores TF-IDF resultantes de las operaciones anteriores.

Toda esta información se aplica en Python con la siguiente formulación y el resultado es un porcentaje final similitud.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
respuestas_correctas = [
]
respuestas_generadas = [
]
vectorizer = TfidfVectorizer()
all_responses = respuestas_correctas + respuestas_generadas
tfidf_matrix = vectorizer.fit_transform(all_responses)
tfidf_correctas = tfidf_matrix[:len(respuestas_correctas)]
tfidf_generadas = tfidf_matrix[len(respuestas_correctas):]
similaridades = cosine_similarity(tfidf_correctas, tfidf_generadas)
similitudes_diagonales = np.diag(similaridades)
porcentajes_similitud = similitudes_diagonales * 100
print(porcentajes_similitud)
```

El código de Python que se ha implementado para ejecutar Esta base de pruebas es: Primero, es necesario instalar las bibliotecas necesarias para interactuar con la API de OpenAI y para el procesamiento de texto. Esto se logra con el siguiente comando en la terminal:

```
pip install openai scikit-learn nltk
```

Este comando sirve para que todas las herramientas necesarias estén disponibles para el código. A continuación, se importan las bibliotecas esenciales en el código:

```
import openai

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine_similarity
```

La biblioteca **openai** permite interactuar con la API de OpenAI, mientras que **TfidfVectorizer** y **cosine_similarity** de **sklearn** se utilizan para calcular la similitud entre textos usando el método TF-IDF y la similitud coseno, respectivamente. Posteriormente, se configura la clave de la API de OpenAI:

```
openai.api_key = 'Se mantiene privada'
```

Aquí, openai.api_key se establece con tu clave API válida para que puedas acceder al modelo de ChatGPT. Para obtener una respuesta de ChatGPT, se define la siguiente función: Se define el motor de inteligencia artificial que se va a utilizar, en el caso del ejemplo "gpt-3.5-turbo", en el caso de ChatGPT4, el nombre que recibe el motor es "gpt-4-turbo" y para Gemini "Gemini-1.5-flash"

```
def obtener_respuesta_chatgpt(pregunta):
    response = openai.Completion.create(
        engine="gpt-3.5-turbo",
        prompt=pregunta,
        max_tokens=num_tokens_deseados
    )
    return response.choices[0].text.strip()
```

Esta función envía una pregunta a ChatGPT utilizando el modelo **gpt-3.5-turbo** y limita la respuesta a 150 tokens. La respuesta se extrae y limpia usando **response.choices[0].text.strip().** Para calcular la similitud entre la respuesta de ChatGPT y una respuesta correcta, se define la siguiente función:

```
def calcular_similitud(respuesta_chatgpt, respuesta_correcta):

    vectorizer = TfidfVectorizer()

    tfidf_matrix = vectorizer.fit_transform([respuesta_chatgpt, respuesta_correcta])

    similitud = cosine_similarity(tfidf_matrix[0:1], tfidf_matrix[1:2])

    return similitud[0][0]
```

Además de usar el método TF-IDF, se va a utilizar de nuevo la inteligencia artificial para que mida el si se cumple el objetivo del prompt, ya que el método de comparación solo mide los resultados en texto, sin embargo, la similitud, a pesar de ser diferente el texto puede cumplir el mismo objetivo, para ello, utilizamos la IA, que coge el resultado de la comparación y le suma la capa de comprensión del texto en respuesta al objetivo.

```
def confirmar_similitud_con_chatgpt(pregunta, respuesta_correcta, respuesta_chatgpt,
tfidf_similitud, num_responses=3):
    similarity_scores = []
    confirmation_prompt = f"""
    Given the problem: "{pregunta}"
    The correct solution is: "{respuesta_correcta}"
    The ChatGPT solution is: "{respuesta_chatgpt}"
    The TF-IDF similarity score is: {tfidf_similitud:.2f}
    Determine if the ChatGPT solution achieves the same objective as the correct solution.
Provide ONLY the similarity percentage from 1-100 being 100 equal answers in content
and effectiveness of answer to the prompt.
    """
```

```
    for _ in range(num_responses):
        response = openai.ChatCompletion.create(
            model="gpt-3.5-turbo",
            messages=[
                {"role": "system", "content": "You are an assistant that evaluates the similarity between your answer and the correct answer, taking into account not only the similarity in text, but the similarity of the objective. Provide ONLY the percentage of similarity answer from 1-100."},
                {"role": "user", "content": confirmation_prompt}
            ],
            max_tokens=10
        )
        similarity_score = response.choices[0]['message']['content'].strip()
        similarity_scores.append(float(similarity_score))

    return sum(similarity_scores) / len(similarity_scores)
```

En esta función, **TfidfVectorizer** convierte los textos en vectores TF-IDF y **cosine_similarity** calcula la similitud coseno entre estos vectores. Para procesar un archivo que contiene preguntas y respuestas correctas, se utiliza la siguiente función:

```
def procesar_archivo(nombre_archivo):
    if not os.path.isfile(nombre_archivo):
        print(f"Error: El archivo '{nombre_archivo}' no existe.")
        return
```

```
    preguntas = []
    respuestas_correctas = []
    with open(nombre_archivo, 'r') as archivo:
        lineas = archivo.readlines()
        for i in range(0, len(lineas), 2):
            if lineas[i].startswith("input:"):
                pregunta = lineas[i].strip().replace("input:", "").strip()
                respuesta_correcta = lineas[i+1].strip().replace("answer:", "").strip()
                preguntas.append(pregunta)
                respuestas_correctas.append(respuesta_correcta)

    resultados = []
    for pregunta, respuesta_correcta in zip(preguntas, respuestas_correctas):
        respuesta_chatgpt = obtener_respuesta_chatgpt(pregunta)
        print(f"Pregunta: {pregunta}")
        print(f"Respuesta de ChatGPT: {respuesta_chatgpt}\n")
        similitud = calcular_similitud(respuesta_chatgpt, respuesta_correcta)
        confirmacion = confirmar_similitud_con_chatgpt(pregunta, respuesta_correcta,
respuesta_chatgpt)
        print(f"Confirmación de ChatGPT: {confirmacion}%\n")
        resultados.append({
            'pregunta': pregunta,
            'respuesta_chatgpt': respuesta_chatgpt,
            'respuesta_correcta': respuesta_correcta,
            'similitud': similitud,
            'confirmacion_chatgpt': confirmacion })
    calcular_media_por_bloques(resultados)
```

Esta función abre el archivo en modo lectura y extrae las preguntas y respuestas correctas, asumiendo que están en líneas alternas. Luego, obtiene las respuestas de ChatGPT y calcula la similitud entre las respuestas de ChatGPT y las correctas, almacenando los resultados en una lista.

La siguiente función busca imprimir el porcentaje medio en bloques de 100, para que se pueda hacer una comparación más precisa dentro de la base de pruebas. De esta forma, se obtiene una visión detallada de cómo varía la precisión del modelo en diferentes partes del conjunto de datos, lo que es útil para identificar patrones de rendimiento y áreas de mejora.

```
def calcular_media_por_bloques(resultados, bloque_tamano=100):
    num_bloques = (len(resultados) + bloque_tamano - 1) // bloque_tamano


    for i in range(num_bloques):
        inicio = i * bloque_tamano
        fin = inicio + bloque_tamano
        bloque = resultados[inicio:fin]
        if not bloque:
            break
        similitud_media = sum([resultado['similitud'] for resultado in bloque]) / len(bloque)
        print(f"{inicio}-{fin}: {similitud_media * 100:.2f}%")
```

Finalmente, se define la función principal para ejecutar el script:

```
def main():
    nombre_archivo = 'ruta del archivo'
    resultados = procesar_archivo(nombre_archivo)
```

La función iterará sobre los resultados obtenidos y los imprimirá, mostrando la pregunta, la respuesta generada por ChatGPT, la respuesta correcta y la similitud calculada.

3.3.2 PYTHON-CODES

La ingeniería informática es una rama que ha tenido un crecimiento exponencial desde el nacimiento de la programación. A medida que la tecnología cobra protagonismo en las vidas humanas, sigue aumentando su presencia. Antes, era un sector con unas barreras de entrada elevadas, ya que requería del conocimiento del lenguaje de programación en demanda para poder acceder a él Con el surgimiento de las inteligencias artificiales ha sufrido una disrupción en el sector, ya que estas son capaces de convertir lenguaje natural escrito a código.

La base de pruebas para este análisis consiste en una serie de instrucciones para crear programaciones sencillas. El input que introduce la base de pruebas es una sencilla tarea y el programa debe programar en Python la tarea con éxito. En la propia base de pruebas se encuentra la solución correcta al programa requerido.

De nuevo, el ejemplo del input de la base de pruebas es: *"Give me a Python function that'll download a CSV file from a URL."* (Dame una función de Python que descargue un CSV desde una URL)

Esta base de pruebas exige una metodología parecida a la que se ha llevado a cabo con la base STEM-AI, siendo un prompt que deja libre a la inteligencia artificial para la respuesta y posteriormente se compara con la respuesta que la base de pruebas da como correcta.

La diferencia radica en el tipo de pregunta, mientras que STEM-AI busca una respuesta teórica, Python-codes requiere de una respuesta que involucra la programación. Una

característica de esta disciplina es la posibilidad de conseguir el mismo objetivo por diferentes caminos. Normalmente esta propiedad es positiva, permitiendo la creatividad y la optimización, sin embargo, para este trabajo supone un problema para comparar las respuestas con el método TF-IDF o cualquier otro, ya que requieren que la respuesta en texto sea similar para tener un mayor grado de similitud. Puede llegar a darse el caso que la programación que devuelve la inteligencia artificial y la base de pruebas sea completamente diferente y que el resultado sea igualmente correcto, resultando en un 0% de similitud. Más aún el método TF-IDF no distingue entre texto y código, bajando el porcentaje de similitud también.

Además, los datos de las bases de datos perteneces a bases de prueba de entrenamiento para inteligencias artificiales, orientadas a entrenar al modelo en datos específicos para aumentar su capacidad de precisión en una tarea específica. Esto sugiere que la base de pruebas utiliza un estilo de programación concreto que no tiene que coincidir por el utilizado por las inteligencias artificiales.

Estos motivos hacen que la comparación con el estilo de base de datos como Python-codes, se obtengan valores muy inferiores a los del estudio y a los valores reales. Incluir la programación y los resultados distorsionaría las conclusiones del estudio de manera no real ya que los resultados que devuelve el código son muy inferiores a la similitud real de similitud que tienen las respuestas en el objetivo en si.

3.3.3 GHOSTS

Medir el rendimiento en matemáticas es una tarea monolítica, ya que esta disciplina se extiende en complejidad y profundidad de manera infinita. La base de pruebas GHOSTS (*Alhashim et al. (2023)*), fue creada con el propósito de comprobar las capacidades de la inteligencia artificial frente a preguntas matemáticas de todos los niveles y en distinto tipo de formato. La propia base de pruebas está dividida en sub-bases de pruebas para aumentar la precisión del análisis de las respuestas y cada subcolección de datos está compuesta por preguntas de distinto nivel de dificultad.

El tipo de preguntas del que está compuesta la base de pruebas sigue este patrón alfanumérico:

Código	Descripción
Dificultad matemática (creciente)	
M1	Problemas aritméticos elementales
M2	Problemas simbólicos (integración de funciones).
M3	Ejercicios de nivel (pre)universitario de libros de texto referenciados en el estudio.
M4	Ejercicios en el estilo de problemas de olimpiadas matemáticas.
Tipo de prompt	
Q1	Preguntas de repaso, que piden declarar o nombrar ciertos hechos matemáticos correctamente.
Q2	Preguntas de repaso de tipo general, que abarcan todo un campo de las matemáticas.
Q3	Preguntas computacionales.
Q4	Preguntas basadas en pruebas, que piden una demostración de un teorema o la solución de un acertijo.
Q5	Preguntas de finalización de pruebas, donde una prueba que tiene lagunas o está incompleta necesita ser completada.

Tabla 2: Descripción del tipo de preguntas en la base de pruebas Ghosts

Este código es el que rige el tipo de pregunta de la base de pruebas, no obstante, esta también está dividida según la fuente de la información y el tipo de pregunta de la siguiente forma:

Nombre del Subconjunto	# preguntas	Fuente de información	Tipo de pregunta
Grad-Text	28	W. Rudin, Functional Analysis (ch. 1)	M3 Q4
	15	W. Rudin, Functional Analysis (ch. 2)	M3 Q4
	37	J. Munkres, Topology (ch. 1)	M3 Q4
	29	J. Munkres, Topology (ch. 2)	M3 Q4
	21	R. Durrett, Probability Theory	M3 Q4
Holes-in-Proofs	60	Proofs Collection A	M3 Q1 Q2 Q5
	52	Proofs Collection B Prealgebra	M1 Q5
	50	Proofs Collection B Precalculus	M1 Q5
Olympiad-Problem-Solving	101	Olympiad Problem Solving	M4 Q4 D2
Symbolic-Integration	100	Symbolic Integration	M2 Q3 D1
MATH	50	MATH Algebra	M1 M2 M3 Q3 Q4
	50	MATH Counting and Probability	M1 M2 M3 Q3 Q4
	18	MATH Prealgebra	M1 Q3 Q4
	20	MATH Precalculus	M1 Q3 Q4
Search-Engine-Aspects	30	Definition Retrieval	M3 Q2 D3
	30	Reverse Definition Retrieval	M3 Q1 Q2
	18	Named Theorem Proof Completion	M3 Q2 Q5

Tabla 3: Fuente de contenido de las preguntas de la base de pruebas Ghosts

Esta tabla muestra el esqueleto de la base de pruebas, se puede observar que es completa en cuanto a información y fuente de las preguntas. Sin embargo, es pequeña en cuanto al volumen de las preguntas para hacer un estudio estadístico.

Cómo existen tantos tipos de preguntas y formas de responder al prompt, no tiene sentido crear un código para cada una, por lo que esta vez, el estudio se llevará a cabo de manera manual sin realizar una automatización mediante Python. Se probarán 10 prompts por cada uno de los tipos, y en base a esa muestra se hará el conteo estadístico.

Las circunstancias obligan a un método de valoración manual, para ello se utilizará el mismo que se usa en el estudio, dónde se le da una puntuación a cada una de las preguntas del 1- 5, según lo completa que esté la respuesta respecto a la solución ideal. En la siguiente tabla se muestra el sistema de puntuación claramente.

Puntuación	Descripción
1	fallo en comprender el prompt (Ejemplo:, el prompt pregunta sobre teoría de números, y responde con información sobre ecuaciones diferenciales)
2	el prompt fue entendido, pero la respuesta fue completamente incorrecta (Ejemplo:, el prompt pregunta cuales son los divisores primos de 111, y responde con 8 y 6)
3	el prompt fue entendido, pero la respuesta fue solo parcialmente correcta (Ejemplo:, el prompt pregunta cuales son los divisores primos de 111, y responde con 3 y 6)
4	el prompt fue entendido, y la respuesta fue mayormente correcta (Ejemplo:, el prompt pregunta cuales son los divisores primos de 222, y responde con 3 y 37)
5	el prompt fue entendido y la respuesta fue completamente correcta

Tabla 4: Método de puntuación base de pruebas Ghosts

Para calcular el porcentaje total, cada uno de la puntuación resta un 25%, empezando por una puntuación de 5 que corresponde a un 100%, 4 al 75% y bajando hasta la puntuación de 1 que corresponde a un 0%.

Un ejemplo de una respuesta con una puntuación de 5 es este. Al prompt "*How many vertical asymptotes does the graph of $y=\\frac{2}{x^2+x-6}$ have?"*

la respuesta de ChatGPT4 es:

Para determinar la cantidad de asíntotas verticales de la gráfica de la función \(y = \frac{2}{x^2 + x - 6} \), empezamos analizando el denominador, \(x^2 + x - 6 \). Las asíntotas verticales ocurren donde el denominador es cero y el numerador es distinto de cero.
Primero, resolvemos para encontrar las raíces de la ecuación cuadrática en el denominador:
\[
x^2 + x - 6 = 0
\]
Para factorizar la ecuación cuadrática, buscamos dos números que multiplicados den \(-6\) y sumados den \(1\). Estos números son \(3\) y \(-2\). Así, podemos factorizar la cuadrática de la siguiente manera:
\[
x^2 + x - 6 = (x + 3)(x - 2)
\]
Igualando cada factor a cero obtenemos las soluciones:
\[
x + 3 = 0 \quad \Rightarrow \quad x = -3
\]
\[
x - 2 = 0 \quad \Rightarrow \quad x = 2
\]
Por lo tanto, el denominador \(x^2 + x - 6 \) es cero en \(x = -3 \) y \(x = 2 \). Como el numerador \(2 \) es distinto de cero en estos puntos, tanto \(x = -3 \) como \(x = 2 \) son asíntotas verticales de la función.

Concluimos que la gráfica de \(y = \frac{2}{x^2 + x - 6} \) tiene dos asíntotas verticales.

Así, el número de asíntotas verticales es:

2

Esta respuesta es perfecta, entiende el prompt, aporta contexto y devuelve la respuesta correcta.

3.3 ANÁLISIS DE SENTIMIENTO

El análisis de sentimiento es una práctica común en el campo de la inteligencia artificial y el machine learning. Consiste en determinar las emociones que inspira un texto de cualquier tipo, desde opiniones, comentarios, reseñas, libros… Se evalúa el sentimiento que inspira ese contexto, varía entre positivo, negativo o neutral. En algunos análisis se puede llegar a identificar emociones cómo tristeza o alegría, pero queda fuera del objetivo de este proyecto.

El uso de estos análisis es para analizar la capacidad de la inteligencia artificial en la comprensión del lenguaje natural y sus matices. El sentimiento de una frase es de las cosas más complejas de comprender, por lo que supone una medida fidedigna de la capacidad de estas herramientas de comprender el lenguaje natural.

Este estudio se va a realizar usando una base de pruebas que prueba simplemente si es sentimiento de la frase es positivo, negativo o neutral, de esta forma, el análisis es sencillo y la obtención de datos es clara para posteriormente realizar un análisis completo. La base de pruebas consta de frases de diferentes tipos y la solución sobre sí el sentimiento es positivo, neutro o negativo. El programa tiene que responder el sentimiento de la frase y comparar con la respuesta correcta.

Este código es sencillo y el resultado porcentual es exacto, ya que la comparación es binaria, ya sea correcto o incorrecto, la respuesta siempre se puede comprobar de manera directa. A cada uno de los sentimientos posibles (positivo, negativo, neutral) se le asigna un número del 0-2 para facilitar la comparación futura. 0 – negativo, 1 – neutral, 2 – positivo.

Sucede algo concreto con esta base de pruebas, que se analizará debidamente en el apartado de conclusiones. El tipo de input que introduce la base de pruebas a la inteligencia artificial proviene de comentarios de internet que están redactados en formato informal o coloquial que para la inteligencia artificial supone un reto elevado descifrar el sentimiento de la frase.

Por lo tanto, se ha hecho el mismo análisis con otra base de pruebas que está formada por texto formal.

El código de importar librerías, importar el documento y llamar a la API de las inteligencias artificiales ya se ha explicado previamente, tiene esta forma:

```
import pandas as pd

import openai

file_path = '/Users/Marco/Desktop/Sentiment2_DS1.xlsx'

data = pd.read_excel(file_path)

openai.api_key = 'PRIVADA'
```

La base de pruebas tiene las frases del análisis de sentimiento en la segunda columna, y en la tercera columna se encuentra el código para la respuesta correcta. Se extrae del documento esa información y se convierte en listas.

```
phrases = data.iloc[:, 1].tolist()

correct_labels = data.iloc[:, 2].tolist()
```

Ahora son las listas de **phrases,** y **correct_labels** las que tienen la información de la base de pruebas.

Posteriormente, se define la función para obtener el sentimiento de la inteligencia artificial. Esta función ya se ha utilizado en otras ocasiones y hace una llamada a la inteligencia artificial y le pide que responda a la frase en este caso. El número máximo de tokens en este caso es 10, ya que la respuesta es corta. (Positivo, Negativo, Neutro).

```
def get_sentiment(phrase):
    response = openai.Completion.create(
        engine="gpt-3.5-turbo",
        prompt=f"Analiza el sentimiento de la siguiente frase y devuelve ÚNICAMENTE 'positivo', 'neutral' o 'negativo': {phrase}",
        max_tokens=10
    )
    sentiment = response.choices[0].text.strip().lower()
    return sentiment
```

El uso de la palabra únicamente, que esta subrayada en el apartado de código anterior es lo que permite que el código funcione. La inteligencia artificial requiere de mucha precisión. Sin la palabra que especifica que solo devuelva los valores que se van a utilizar en el código se obtendrían frases largas imposibles de convertir a número posteriormente y comparar con la respuesta correcta. Además, se limita el uso de tokens para no incentivar a que de repuestas largas.

Posteriormente, se convierte la respuesta de la IA en el código numérico para poder comparar con el resultado correcto. Se hace de esta forma ya que se trabaja con la certeza que la IA solo va a devolver una de esas tres palabras.

```
def sentiment_to_score(sentiment):
    if sentiment == 'positivo':
        return 2
    elif sentiment == 'neutral':
        return 1
    elif sentiment == 'negativo':
        return 0
    else:
        return -1
```

Esta información queda guardada en la información **sentiment_to_score** el valor -1 es en caso de que no coincida con ninguna de las respuestas predeterminadas, que el programa no falle. Con la siguiente función se calcula la precisión, coge la información de las listas

predicted_scores y **correct_labels.**

```
predicted_scores = [sentiment_to_score(get_sentiment(phrase)) for phrase in phrases]

def calculate_accuracy(predicted, correct):
    correct_predictions = sum(p == c for p, c in zip(predicted, correct))
    return correct_predictions / len(correct)
```

Una vez se obtiene esta función, ya se ha conseguido unir la lista de los resultados que devuelve la inteligencia artificial con los resultados correctos proporcionados por la base de pruebas. Lo que resta es convertir los datos obtenidos al formato que se busca para poder hacer los análisis correspondientes. Esta función se encarga de dividir por intervalos los resultados.

```
accuracies = []
for i in range(0, len(predicted_scores), 100):
    interval_scores = predicted_scores[i:i + 100]
    interval_correct = correct_labels[i:i + 100]
    interval_accuracy = calculate_accuracy(interval_scores, interval_correct)
    accuracies.append(interval_accuracy)
```

Y la siguiente función calcula el resultado total.

```
overall_accuracy = calculate_accuracy(predicted_scores, correct_labels)
```

Por último, se genera un informe para extraer la información.

```
report = f"Informe de Precisión del Análisis de Sentimiento:\n\n"
for i, accuracy in enumerate(accuracies):
    report += f"Intervalo {i*100} - {(i+1)*100}: {accuracy * 100:.2f}%\n"
print(report)
```

4

RESULTADOS

A continuación, se muestran los resultados del proyecto, estos se muestran presentados en tablas divididas en intervalos de diferente amplitud dependiendo del tamaño de la base de pruebas introducida. Posteriormente, se realiza un análisis de los resultados obtenidos individual y colectivamente. Es relevante destacar que es un sector en auge, estos resultados son una "fotografía" de la situación actual, no son perpetuos. Tanto es así que durante la realización del estudio OpenAI ha lanzado una herramienta superior a ChatGPT4 llamada GPT4o.

4.1 STEM-AI

Una vez se ha ejecutado el script de Python para las tres inteligencias artificiales, resaltar que la función de llamada a la API de cada una cambia para referenciar directamente al programa en cuestión y poder acceder a las respuestas de la misma. Se obtiene la siguiente tabla de resultados. Por conveniencia y eficiencia, los resultados extraídos de Python quedan recogidos en esta tabla.

Intervalo	ChatGPT4	ChatGPT3,5	Gemini
0-100	82,35%	79,25%	78,92%
100-200	84,86%	82,64%	80,24%
200-300	75,92%	75,00%	74,23%
300-400	78,22%	73,85%	72,31%
400-500	85,97%	76,14%	71,67%
500-600	78,62%	80,43%	70,23%
600-700	79,34%	75,26%	73,88%
700-800	84,56%	81,09%	75,61%
800-900	77,45%	70,11%	74,21%
900-1000	87,24%	76,25%	75,38%
1000-1100	80,45%	75,39%	80,78%
1100-1200	85,97%	77,83%	78,98%
Promedio	**81,75%**	**76,94%**	**75,54%**

Tabla 5: Resultados STEM-AI

Los resultados obtenidos al procesar los datos de las tres inteligencias artificiales, ChatGPT4, ChatGPT3.5 y Gemini, revelan promedios de rendimiento cercanos al 80%, solo siendo ChatGPT4 capaz de superar este número. ChatGPT4, es la inteligencia artificial que mejor rendimiento ha tenido en esta prueba, superando en un 5,09% a ChatGPT3.5 y en un 7.84% al rendimiento de Gemini.

Este gráfico muestra los promedios más claramente y sirve como referencia para situar los rendimientos de las inteligencias artificiales en la base de pruebas de ingeniería eléctrica.

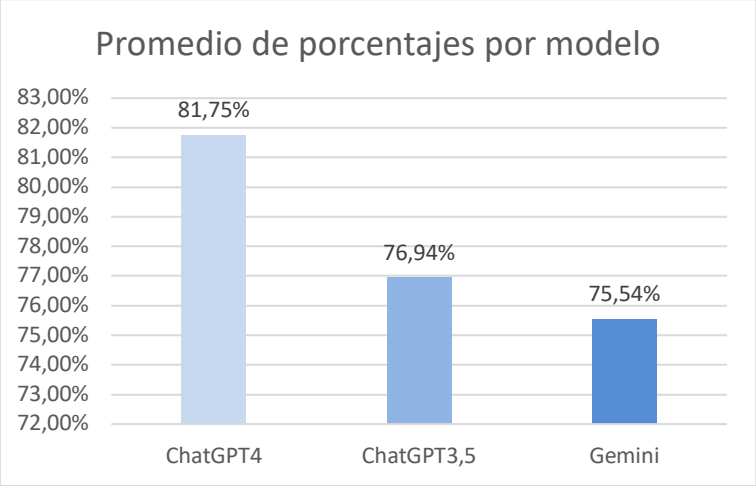

Tabla 6: Promedio de porcentaje STEM_AI por modelo

Es destacable que ninguno de los resultados de los subgrupos de 100 supera el 90%, esto se puede deber a dos motivos. En primer lugar, la limitación intrínseca de las inteligencias artificiales para resolver problemas a los que se exponen los humanos. Este es el objeto de estudio de este trabajo y es importante definir. En este caso, la tipología de la base de pruebas estando compuesta de preguntas teóricas debería de atacar una de las fortalezas de la inteligencia artificial. Ya que, en la arquitectura transformers de la que están compuestas, esa información debería estar lemmatizada y ser capaces de dar respuestas teóricas más precisas. Por lo que, el segundo motivo que puede haber afectado en cierta medida al rendimiento es el método de comparación TF-IDF que tiene limitaciones intrínsecas, y que por la diferencia en la longitud de la respuesta y en las diferentes formulaciones de las oraciones puede dar lugar a cierto margen de error. Esta metodología puede haber afectado en el porcentaje final de similitud haciendo que sea más bajo de la capacidad real de la inteligencia artificial.

Aunque, el hecho de que los promedios sean inferiores a lo esperado puede sugerir que, aunque las inteligencias artificiales son competentes en una amplia gama de temas, su precisión disminuye cuando se trata de conocimiento especializado o contextualizado. Esta observación es importante para entender las capacidades y limitaciones de las IA actuales, y plantea la necesidad de desarrollar técnicas más avanzadas que puedan manejar con mayor precisión tareas específicas sin sacrificar su desempeño en áreas generales. En resumen, las herramientas de IA evaluadas muestran una clara tendencia a rendir mejor en contextos de conocimiento general, pero enfrentan desafíos significativos cuando se requiere un conocimiento más detallado y específico.

En el siguiente gráfico se muestran los puntos porcentuales en cada uno de los intervalos que se han creado y cómo se ha comportado cada una de las herramientas del estudio.

En un cálculo rápido de la correlación siguiendo la fórmula de la correlación de Pearson (r):

$$r = \frac{n \cdot \sum x_i \cdot y_i - \sum x_i \cdot \sum y_i}{\sqrt{[n \cdot \sum x_i^2 - (\sum x_i)^2] \cdot [n \sum y_i^2 - (\sum y_i)^2]}}$$

Figura 16: Fórmula correlación Pearson

Esta fórmula es básica en matemáticas y el contenido de la misma se puede encontrar fácilmente en internet. Por lo que no se realizará una explicación en detalle si no que directamente se utilizará para obtener las correlaciones entre las tres inteligencias artificiales. Se obtiene una tabla de correlaciones con la siguiente forma:

3	ChatGPT4	ChatGPT3,5	Gemini
ChatGPT4	1	0,5	0,36
ChatGPT3,5	0,5	1	0,29
Gemini	0,36	0,29	1

Tabla 7: Correlación entre las inteligencias artificiales en STEM-AI

En esta prueba hay una correlación fuerte entre las tres inteligencias artificiales, las más correlacionadas son ChatGPT4 y ChatGPT3.5 que tienen una correlación fuerte con 0.5. Gemini no se queda atrás con una correlación superior a 0.25 con las otras dos herramientas de inteligencia artificial. La correlación se puede deber a ya que son preguntas teóricas y las tres inteligencias artificiales hayan extraído sus datos de fuentes parecidas. Intuitivamente, las dos inteligencias propiedad de OpenAI deben estar correlacionadas, ya que beben de la misma fuente y uno es la evolución del otro.

4.2 PYTHON-CODES

Los resultados de esta base de datos como se ha explicado en el apartado anterior no se van a incluir en el estudio por las limitaciones intrínsecas del código y la base de datos. Causarían una distorsión de los datos no realista por la excelente capacidad de la inteligencia artificial para tareas de programación que no se ve reflejada con la programación y los recursos de los que se dispone para este proyecto.

4.3 GHOSTS

El análisis que se ha realizado deja la siguiente tabla:

		ChatGPT4		ChatGPT3.5		Gemini	
Nombre del Subconjunto	# pregunta	Puntuación #1	Puntuación #2	Puntuación #1	Puntuación #2	Puntuación #1	Puntuación #2
Grad-Text	2	5	4	4	4	4	3
	2	3	4	4	3	4	4
	2	2	3	2	2	2	4
	2	5	3	4	3	5	3
	2	4	4	5	4	3	4
Huecos en las pruebas	2	5	5	5	5	5	5
	2	4	4	3	4	3	3
	2	3	4	4	4	4	4
Resolución problemas olimpiadas	2	2	5	2	4	1	4
Integración simbólica	2	2	4	1	4	2	3
MATH	2	5	5	5	5	5	4
	2	5	4	3	2	2	3
	2	4	3	4	5	4	4
	2	5	5	4	4	5	3
Search-Engine-Aspects	2	4	3	5	3	4	3
	2	4	5	3	3	4	3
	2	3	5	3	4	2	5
Total	34	65	70	61	63	59	62
Promedio		76%	82%	72%	74%	69%	73%
			79%		**73%**		**71%**

Tabla 10: Resultados inteligencia artificial GHOST

Las sensaciones durante el proceso de completar la tabla coinciden con las sensaciones en los resultados, y es claramente que ChatGPT4 es el modelo con mayor exactitud cuando devuelve la respuesta. La forma de presentar la solución, a pesar de ser parecida, la del modelo ChatGPT4 era sin duda la que más confianza retornaba.

Estos resultados indican una capacidad de las inteligencias artificiales en resolver problemas matemáticos de distinta índole cercano a un 80% de efectividad. Esto para trabajos de ingeniería que requieran de solución de problemas matemáticos es un gran indicador, aunque el margen de error es demasiado grande para poder establecer una confianza ciega en este tipo de herramientas.

Durante el proceso de completar la tabla, la sensación cuando una inteligencia artificial da una respuesta es que esta siempre va a ser correcta, se necesita conocer la respuesta para descubrir que efectivamente, la IA se ha equivocado. Este fenómeno se estudiará en detalle en las conclusiones.

Por otro lado, de los datos se puede observar que la IA rara vez consigue una puntuación de 1 o 2. Esto destaca la capacidad de la inteligencia artificial de responder no lejos del objetivo de la pregunta, a pesar de que pueda fallar ante preguntas matemáticas ciertamente complejas.

	ChatGPT4	ChatGPT3.5	Gemini	Total
1	0	1	1	2
2	3	4	4	11
3	7	8	10	25
4	12	14	13	39
5	12	7	6	25

Tabla 11: Conteo de veces que ha resultado cada puntuación dividida por IA

ChatGPT4, siendo superior a las otras 2 inteligencias artificiales muestra mayor capacidad de responder a las preguntas complejas con la máxima nota y tiene el mismo número de 5 que de 4, mientras que en las otras dos inteligencias artificiales el 4 es la moda claramente. De nuevo, la superioridad de ChatGPT4 se muestra en la capacidad matemática.

4.4 ANALISIS DE SENTIMIENTO

Los resultados que devuelve el programa en el formato de tabla son los siguientes:

Intervalos	ChatGPT-4	ChatGPT-3,5	Gemini
0-250	72,04%	79,54%	62,64%
250-500	54,63%	70,95%	72,93%
500-750	57,05%	63,49%	50,00%
750-1000	80,00%	75,53%	59,98%
1000-1250	70,93%	66,32%	69,84%
1250-1500	52,29%	63,92%	66,09%
1500-1750	55,17%	67,01%	64,89%
1750-2000	72,71%	77,12%	65,04%
2000-2250	52,16%	69,79%	55,62%
2250-2500	57,02%	71,34%	53,81%
2500-2750	69,13%	72,92%	67,97%
2750-3000	73,59%	74,88%	67,99%
3000-3250	72,35%	78,98%	69,56%
3250-3500	59,00%	66,85%	70,82%
3500-3750	74,23%	67,94%	59,79%
3750-4000	59,41%	70,16%	59,52%
4000-4250	62,16%	69,58%	67,67%
4250-4500	75,30%	69,62%	63,60%
4500-4750	72,28%	76,70%	65,02%
4750-5000	78,50%	69,18%	67,60%
Promedio	**66,00%**	**71,09%**	**64,02%**

Tabla 12: Rendimiento IA en base de pruebas análisis de sentimiento

En primer lugar, los resultados son extremadamente bajos respecto a los resultados que se estaban obteniendo a lo largo del proyecto, no debería ser así, ya que el análisis de sentimiento es para lo que están entrenadas estos algoritmos, que es comprender el lenguaje natural humano. Además, en el apartado anterior han demostrado la capacidad de

comprender ejercicios complejos de matemáticas redactados en lenguaje natural. Esto puede denotar que la base de pruebas era excesivamente exigente, tiene sentido ya que consiste en textos producidos de foros con vocabulario de internet y puede haber confundido a las inteligencias artificiales

Además, y fuera de la tónica general de este trabajo, en el que ChatGPT4 estaba dominando todas las categorías, es ChatGPT3.5 quien por una amplia superioridad ha sacado un rendimiento superior. Con un 71,09% de media ha encontrado un rendimiento por encima de su modelo superior y de pago que es ChatGPT4. Por otro lado, Gemini ha tenido un rendimiento inferior pero parecido a ChatGPT4 en este análisis de sentimiento.

Para ver si se está fuera de orden por culpa de la base de pruebas se va a utilizar una fuente bibliográfica para comprobar que las inteligencias artificiales tienen un sustancial rendimiento superior al que se ha obtenido con Esta base de pruebas.

En análisis bibliográficos, el rendimiento de estas inteligencias artificiales es superior al que se ha obtenido en Esta base de pruebas, en este estudio, *(Is ChatGPT a General-Purpose Natural Language Processing Task Solver?)* la puntuación que obtiene ChatGPT3.5 es de un 88%, cerca de un 17% superior.

Con el objetivo de despejar las dudas sobre si es el código, o la base de pruebas que es demasiado compleja. Ya que consiste en comentarios y maneras de hablar de usuarios de internet. Se va a probar a utilizar el mismo código para la base SST2, que es la que se utiliza en el paper anterior y de esta forma también se pueden realizar comparaciones. Para mantener *ceteris paribus* (todo lo anterior de la misma forma), se va a utilizar la misma programación de Python que para la base de pruebas anterior.

Los resultados que se obtienen son:

Intervalo	ChatGPT-4	ChatGPT-3,5	Gemini
0-250	85,64%	83,15%	77,08%
250-500	87,28%	85,26%	87,83%
500-750	83,25%	76,54%	85,43%
750-1000	87,94%	82,11%	85,00%
1000-1250	85,50%	81,45%	91,67%
1250-1500	86,27%	84,16%	91,23%
1500-1750	85,50%	79,41%	91,56%
1750-2000	91,63%	85,86%	78,82%
2000-2250	86,97%	89,68%	85,05%
2250-2500	84,36%	84,60%	78,29%
2500-2750	89,06%	88,20%	91,17%
2750-3000	83,95%	84,92%	76,85%
3000-3250	89,37%	86,55%	89,54%
3250-3500	84,56%	89,43%	82,05%
3500-3750	83,68%	91,26%	87,75%
3750-4000	87,49%	88,27%	86,12%
4000-4250	88,85%	89,96%	79,61%
4250-4500	87,43%	84,45%	81,76%
4500-4750	86,71%	77,10%	84,35%
4750-5000	86,25%	91,59%	90,13%
Promedio	**86,58%**	**85,20%**	**85,06%**

Tabla 13: Resultados IA en base de pruebas análisis de sentimiento 2

En este caso, para Esta base de pruebas, sí que se ajustan los resultados a los que se han ido obteniendo a lo largo de la investigación. Dónde el modelo de ChatGPT más avanzado se impone en exactitud al resto. Sin embargo, en este caso la cercanía en la exactitud salta a la vista, siendo la base de pruebas en la que los resultados se han ajustado en mayor medida. Esto se puede deber a la capacidad ya desarrollada y testada de estos modelos de comprender el lenguaje natural, y a la menor dificultad de la base de pruebas con el uso de frases cotidianas y formales.

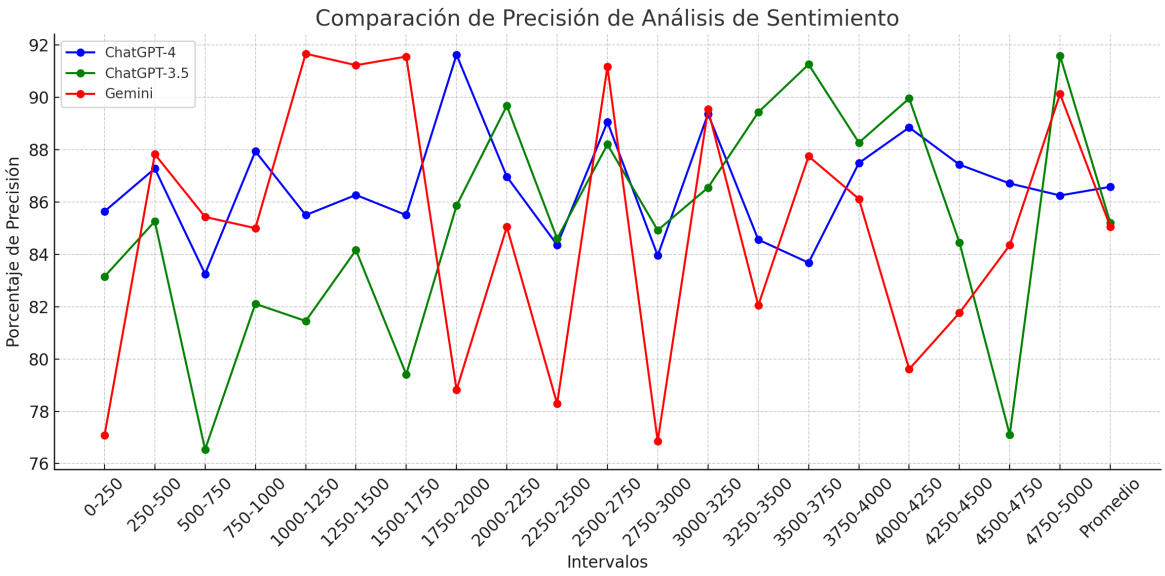

Figura 17: Serie temporal de los resultados obtenidos en el análisis de sentimiento

Este gráfico temporal es extremadamente representativo, muestra las pequeñas diferencias que ha habido en los resultados de las tres inteligencias artificiales, y la elevada capacidad para comprender el lenguaje natural. Sin embargo, destaca los saltos en los diferentes tramos, aunque siempre se mantiene una media elevada. Es necesario realizar un estudio de la correlación, sobre todo en los ChatGPT's que parecen estar altamente correlacionados, pero Gemini, que en situaciones anteriores se quedaba atrás, ha sido capaz de competir en rendimiento.

	ChatGPT4	ChatGPT3,5	Gemini
ChatGPT4	1	0,22036569	0,03438366
ChatGPT3,5	0,22036569	1	-0,0172112
Gemini	0,03438366	-0,0172112	1

Tabla 14: Correlación de la IA en el análisis de sentimiento

5

CONCLUSIONES

5.1 COMPARATIVA GLOBAL

El objetivo de este estudio ha sido evaluar la respuesta efectiva de las diferentes opciones del mercado global de inteligencia artificial en el contexto, necesidades y requerimientos específicos de la ingeniería.

Sus resultados pueden ser un punto de referencia para aquellos ingenieros que acuden a la inteligencia artificial como una herramienta útil para la gestión de alguna de las etapas más complejas del proceso de gestión de cada proyecto.

La IA ha demostrado ser capaz de dar respuesta de manera brillante a las numerosas preguntas relacionadas con las diferentes ramas de la ingeniería. Esto supone, sin duda, un avance mayúsculo. La posibilidad de elevar exponencialmente, con el uso de estas

tecnologías, nuestra capacidad de gestión de un gran volumen de datos e información extremadanamente compleja supone un cambio sustancial en la manera de afrontar la gestión de proyectos de ingeniería en cualquiera de sus ámbitos de acción.

Ya en relación con el presente estudio, los resultados generales del análisis de las diferentes plataformas de inteligencia artificial son los siguientes:

Base de pruebas	ChatGPT4	ChatGPT3.5	Gemini
STEM-AI	82,65%	77,56%	74,81%
GHOSTS	79,00%	73,00%	71,00%
Análisis de sentimiento	86,63%	86,09%	85,48%
Promedio	**82,76%**	**78,88%**	**77,10%**

Tabla 15: Resultados a las bases de pruebas

Esta tabla ilustra el resultado alcanzado por cada una de 3 plataformas observadas, el cual constata diferencias significativas de rendimiento entre ellas:

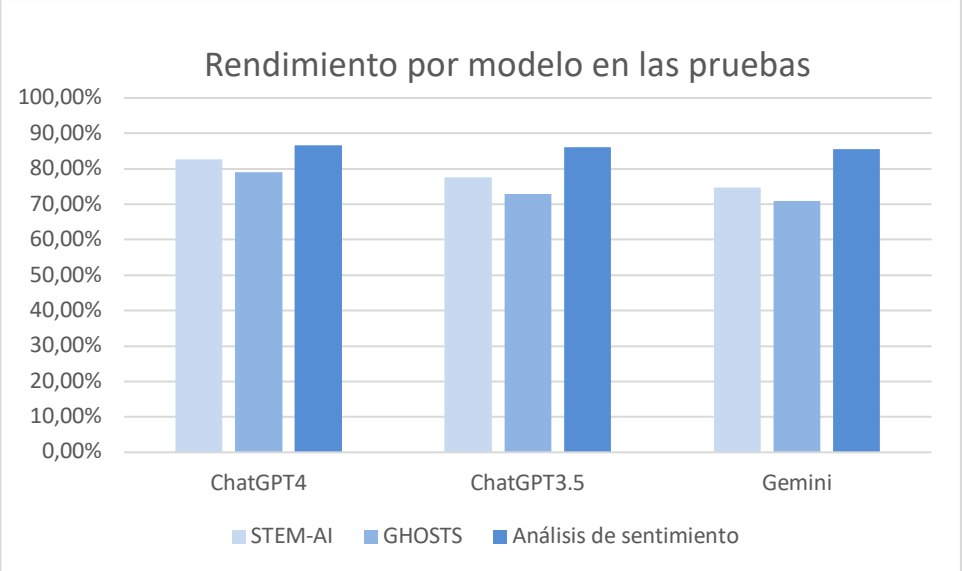

Figura 18: Gráfico de barras con el rendimiento por modelo en las bases de pruebas

ChatGPT4 obtiene los mejores resultados en el ámbito de la ingeniería, en comparación con el resto de las opciones testadas, con un 83,23% de acierto.

En segundo lugar, y muy cercana a Gemini, se sitúa ChatGPT3.5. Es importante mencionar el hecho de que estas dos ofrecen su servicio de manera gratuita, mientras que ChatGPT4 solo es accesible mediante una suscripción mensual.

En todos los casos, el mayor rendimiento se alcanza en el apartado de análisis de sentimiento, la única prueba que es de conocimiento general. El análisis de sentimiento es el primer e indiscutible proceso que tienen que seguir las inteligencias artificiales para comprender el lenguaje natural humano, por lo que es fundamental que sean capaces de realizar estos procesos de manera eficiente. También, la arquitectura de código de tranformers se crea precisamente para poder comprender grandes cantidades de texto sin perder contexto.

En el lado contrario, la prueba que peor rendimiento ha dado ha sido la prueba de matemáticas avanzadas, donde ChatGPT4 casi alcanza el 80%, a cierta distancia del resto de alternativas de mercado.

Como se puede observar, en términos generales las ratios de rendimiento se sitúan normalmente en un rango cercano al 80%. Es un resultado sin duda excelente, en especial si consideramos el relativo corto plazo de tiempo que ha transcurrido desde noviembre del 2022 cuando la inteligencia artificial se empezó a desarrollar de una manera ya decidida en el mercado global.

De cualquier modo, este resultado está lejos de ser suficiente para la ingeniería, en la que los márgenes de error son sustancialmente más estrechos, lo que impediría en este momento la opción de aplicar la inteligencia artificial como una herramienta totalmente autónoma. Sin embargo, se muestra como un complemento muy efectivo para simplificar el proceso de gestión de datos siempre que actúe bajo la supervisión de un responsable de proyecto.

Este modelo de supervisión es todavía indispensable para asegurar su correcta implementación y asegurar una mejora sustancial de la productividad de la industria en su conjunto. El hecho de que los modelos actuales de inteligencia artificial aseguren una tasa de rendimiento del 80% permite a su vez que el individuo pueda focalizarse fundamentalmente en revisar, corregir y afinar ese 20% de información restante que la inteligencia artificial no es capaz de completar de manera exitosa.

5.2 FORTALEZAS Y DEBILIDADES

Este trabajo sirve como punto de referencia para evaluar la posición actual de la inteligencia artificial en términos de desarrollo y capacidad de procesamiento en funciones específicas para la ingeniería. El presente estudio ha documentado algunas fortalezas y limitaciones de esta tecnología, a la vez que ha proporcionado una visión más precisa de su estado actual. Sin embargo, estoy seguro de que en los próximos años se verá un desarrollo significativo en este campo, lo que inevitablemente hará que los hallazgos y análisis presentados aquí queden obsoletos. La rápida evolución de la inteligencia artificial promete mejoras continuas en su rendimiento y capacidades. Cada día salen al mercado nuevas innovaciones, incluso, durante la elaboración de este trabajo OpenAI lanzó al mercado ChatGPT-4 con mayores capacidades que la versión ChatGPT-3.5 con la que se ha experimentado. Este trabajo, por lo tanto, no solo destaca el estado actual de la inteligencia artificial, sino que también subraya la necesidad de seguir investigando para mantener actualizadas las capacidades de estas herramientas.

En base a los resultados de este trabajo se pueden extraer algunas conclusiones sobre las debilidades y fortalezas de la inteligencia artificial en el campo de la ingeniería.

DEBILIDADES IDENTIFICADAS

En primer lugar, una de sus principales debilidades es la procedencia de los datos utilizados para su entrenamiento y evaluación. A menudo, estos datos provienen de fuentes desconocidas o no verificadas, lo que dificulta la tarea de determinar si las respuestas generadas por la IA están sesgadas o son realmente correctas. Sin un conocimiento claro de la calidad y el origen de los datos, es complicado evaluar con precisión la fiabilidad de las soluciones propuestas por la inteligencia artificial. La segunda debilidad es el margen de mejora considerable en términos de precisión y exactitud de las respuestas, un nivel de precisión del 80% no es suficiente, especialmente en el ámbito de la ingeniería por sus efectos en el resultado final de cada proyecto. La precisión con la que se trabaja en ingeniería es superior y no cumple los estándares para sustituir en precisión el trabajo de un ingeniero.

Además, una debilidad de esta disciplina es la cantidad de datos que son necesarios para entrenar eficazmente a las inteligencias artificiales, supone una limitación y barrera de entrada para empresas y personas de menor tamaño que no disponen con la capacidad suficiente para procesar los datos de manera eficiente. Durante la realización de las pruebas con Python, el tiempo de procesamiento de los datos entorpecía el proceso de iteración y perfeccionamiento del código.

FORTALEZAS IDENTIFICADAS

Las fortalezas que ha demostrado la inteligencia artificial en este trabajo son destacables. Es capaz de responder a problemas complejos de distinta índole de manera efectiva y con un margen de confianza apreciable. Esta multimodalidad es un elemento diferencial de las inteligencias artificiales. Además, el potencial inherente de estas herramientas, una vez completado el tiempo requerido para su desarrollo y optimización, conllevará sin duda a una mejora de su capacidad y exactitud.

Las inteligencias artificiales destacan en su habilidad para comprender el sentimiento de las frases y por lo tanto el lenguaje natural, habilidad que permite el desarrollo de futuras características a un nivel similar al desarrollo de estas. A pesar de no haber sido capaz de conseguir resultados relevantes en la base de datos de Python-codes, por los motivos explicados en el proyecto, sí que ha servido de ayuda fundamental para redactar el código utilizado en este proyecto. Ha sido capaz de guiar a una persona con conocimiento básicos en programación a redactar código funcional en el proceso de un proyecto. Por lo que a pesar de no ser capaz de sustentar con datos esta conclusión también destacaría la capacidad de generar y comprender código.

5.3 IMPACTO Y APLICACIONES DE LA IA

La inteligencia artificial se ha consolidado como una herramienta extremadamente poderosa y, a pesar de estar aún en las primeras etapas de su desarrollo, su impacto ya es innegable. Las aplicaciones de esta tecnología son prácticamente infinitas, abarcando desde la medicina hasta la ingeniería, pasando por el análisis de datos y la comunicación. Contar con la inteligencia artificial como ayudante y compañera no solo amplía nuestras capacidades, sino que también optimiza procesos, mejora la precisión y abre nuevas fronteras en la innovación y el conocimiento. Con todo, la inteligencia artificial es un complemento excelente para un director que la guíe y analice los resultados que genera. Aunque requiere todavía de este tutelaje, su uso aumenta enormemente la velocidad en la ejecución de los proyectos gestionados por los directores (ingenieros) que tienen como consecuencia un incremento del rendimiento y la productividad por hora, a la vez que un mayor enfoque en los aspectos más críticos y creativos de sus proyectos. Es decir, la fotografía de la aplicación de la inteligencia artificial en la ingeniería hoy en día es con el rol de optimizar y ayudar con los procesos.

5.4 CONSIDERACIONES Y RECOMENDACIONES FUTURAS

Para finalizar, una consideración importante en lo que se refiere al tipo de análisis aplicado en este trabajo: las numerosas ramas de ingeniería existentes y los infinitos roles y especificidades que cada una de ellas impide afrontar con mucha profundidad y extensión toda la casuística existente. Intentar cubrir cada especialidad no solo sería impracticable, sino también poco útil, ya que ambas disciplinas, tanto la ingeniería como la inteligencia artificial, están en constante evolución. El gasto de recursos necesario para un estudio tan amplio y detallado sería desproporcionado, especialmente si tenemos en cuenta que los avances tecnológicos rápidamente podrían hacer obsoletos los hallazgos obtenidos.

Este trabajo, por lo tanto, no solo destaca el estado actual de la inteligencia artificial, sino que también subraya la necesidad de seguir investigando para mantener actualizadas las capacidades de estas herramientas. La inteligencia artificial se ha consolidado como una herramienta extremadamente poderosa y, a pesar de estar aún en las primeras etapas de su desarrollo, su impacto ya es innegable. Las aplicaciones de esta tecnología son prácticamente infinitas, abarcando desde la medicina hasta la ingeniería, pasando por el análisis de datos y la comunicación. Contar con la inteligencia artificial como ayudante y compañera no solo amplía nuestras capacidades, sino que también optimiza procesos, mejora la precisión y abre nuevas fronteras en la innovación y el conocimiento. Con todo, la inteligencia artificial es un complemento excelente para un director que la guíe y analice los resultados que genera. Aunque requiere todavía de este tutelaje, su uso aumenta enormemente la velocidad en la ejecución de los proyectos gestionados por los directores (ingenieros) que tienen como consecuencia un incremento del rendimiento y la productividad por hora a la vez que un mayor enfoque en los aspectos más críticos y creativos de sus proyectos.

6

BIBLIOGRAFÍA

[1] Vlassis, N.A.; Papakonstantinou, G.; Tsanakas, P. *Dynamic sensory probabilistic maps for mobile robot localization.* Source: Proceedings. 1998 IEEE / RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190) New York, NY, USA: IEEE, 1998.p.718-23 vol.2 of 3 vol. xliv+2010 pp. 11. Últimoacceso 08/05/2024

[2] Loeffler, B. "Cloud Computing: What is Infrastructure as a Service", Microsoft Technet Magazine, October 211. https://technet.microsoft.com/en-us/magazine/hh509051.aspx Últimoacceso 08/05/2024

[3] Herrero Alcántara, T. "Big Data: ¿Moda u oportunidad de negocio para el emprendedor?", Think Big, Octubre 2014. http://blogthinkbig.com/big-data-emprendedor/. Últimoacceso 08/05/2024

[1] Alammar, J (2018). The Illustrated Transformer [Blog post]. Retrieved from https://jalammar.github.io/illustrated-transformer/ Últimoacceso 22/02/2024

[2] Alhashim, I., Shih, K. J., Singh, S., & Wang, Z. (2023). Abstract datasets and benchmarks. Proceedings of the 37th Conference on Neural Information Processing Systems. 10-16 December 2023. Últimoacceso 22/02/2024 https://proceedings.neurips.cc/paper_files/paper/2023/hash/58168e8a92994655d6da3939e7cc0918-Abstract-Datasets_and_Benchmarks.html

[3] Amazon Web Services. (n.d.). What is an API? Amazon Web Services. https://aws.amazon.com/es/what-is/api/ Últimoacceso 22/02/2024

[4] Cambridge University Press. (n.d.). Prompting. In Cambridge dictionary. https://dictionary.cambridge.org/us/dictionary/english/prompting Últimoacceso 22/02/2024

[5] Chen, S., & Li, H. (2023). Towards the next generation of artificial intelligence: Catalyzing the fusion of AI and neuroscience (Version 2). TechRxiv. https://doi.org/10.36227/techrxiv.22683919.v2 Últimoacceso 16/04/2024

[6] Conferencia de Dartmouth: Orígenes de la Inteligencia artificial. ICCSI. (2022, August 25). https://iccsi.com.ar/dartmouth-inteligencia-artificial/ Últimoacceso 22/02/2024

[7] Friederrr. (n.d.). GHOSTS: GitHub repository. GitHub. https://github.com/friederrr/GHOSTS Últimoacceso 25/05/2024

[8] GAMCO. (n.d.). Clasificación de tokens. GAMCO. Retrieved June 1, 2024, from https://gamco.es/glosario/clasificacion-de-tokens/ Últimoacceso 25/05/2024

[9] IBM. (n.d.). ¿Qué es la inteligencia artificial (IA)? https://www.ibm.com/mx-es/topics/artificial-

intelligence#:~:text=Aunque%20varias%20definiciones%20de%20inteligencia,inteligentes%2C%20especialmente%20programas%20inform%C3%A1ticos%20inteligentes. Últimoacceso 10/02/2024

[10] Matej, Jaroslav (2020), "A dataset for machine learning research in the field of stress analyses of mechanical structures", Mendeley Data, V1, https://data.mendeley.com/datasets/wzbzznk8z3/1 Últimoacceso 25/05/2024

[11] Mei, J. P., Chow, K. W., & Chau, H. F. (2020). Fast decoding of polar codes over mixed channels (arXiv:2005.14165v4). arXiv. https://arxiv.org/abs/2005.14165v4 Últimoacceso 25/05/2024

[12] mteb. (n.d.). Tweet sentiment extraction dataset. Hugging Face. https://huggingface.co/datasets/mteb/tweet_sentiment_extraction Últimoacceso 25/05/2024

[13] Olah, C. (2015, August 27). Understanding LSTMs. Colah's Blog. https://colah.github.io/posts/2015-08-Understanding-LSTMs/ Últimoacceso 25/05/2024

[14] Sp1786. (2023). Multiclass sentiment analysis dataset. Hugging Face. https://huggingface.co/datasets/Sp1786/multiclass-sentiment-analysis-dataset/tree/main Últimoacceso 25/04/2024

[15] STEM-AI-mtl. (n.d.). Electrical engineering dataset. Hugging Face. https://huggingface.co/datasets/STEM-AI-mtl/Electrical-engineering?row=3 Últimoacceso 12/04/2024

[16] Theisen, E., & Pekaric, I. (2021). Do institutional investors improve earnings quality? International evidence. ZBW - Leibniz Information Centre for Economics. http://hdl.handle.net/10419/238422 Últimoacceso 12/04/2024

[17] Transformer architecture: The positional encoding. (2019, invierno 9). Kazemnejad.com.https://kazemnejad.com/blog/transformer_architecture_positional_encoding Últimoacceso 22/02/2024

[18] Wang, Y., & Zhao, Y. (2023). Gemini in reasoning: Unveiling commonsense in multimodal large language models. Retrieved from https://arxiv.org/pdf/2312.17661 Últimoacceso 22/02/2024

[19] Word Embedding. (2022, octubre 16). Formación en ciencia de datos | Datascientest.com; DataScientest. https://datascientest.com Últimoacceso 22/02/2024

7

ANEXO

En este apartado se van a recopilar los recursos que complementan la información expuesta en el trabajo.

A continuación, se adjunta el código utilizado para cada una de las bases de pruebas. Por privacidad se mantendrá la clave API de las distintas inteligencias artificiales privadas. Además, por economía solo se adjuntará un ejemplo del código utilizado, son idénticos excepto por la llamada a la clave API entre las herramientas propiedad de OpenAI y Gemini.

Además se añade un extracto de las bases de pruebas que se han utilizado en este proyecto.

Anexo 1: Programación python base de pruebas STEM-AI

Anexo 1.1: Caso para OpenAI (ChatGPT3.5, ChatGPT4)

```
import openai

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine_similarity

openai.api_key = 'Se mantiene privada'
def obtener_respuesta_chatgpt(pregunta):

    response = openai.Completion.create(
```

```
        engine="gpt-3.5-turbo",
        prompt=pregunta,
        max_tokens=num_tokens_deseados
    )
    return response.choices[0].text.strip()

def calcular_similitud(respuesta_chatgpt, respuesta_correcta):
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform([respuesta_chatgpt, respuesta_correcta])
    similitud = cosine_similarity(tfidf_matrix[0:1], tfidf_matrix[1:2])
    return similitud[0][0]

def confirmar_similitud_con_chatgpt(pregunta, respuesta_correcta, respuesta_chatgpt,
tfidf_similitud, num_responses=3):
    similarity_scores = []
    confirmation_prompt = f"""
    Given the problem: "{pregunta}"
    The correct solution is: "{respuesta_correcta}"
    The ChatGPT solution is: "{respuesta_chatgpt}"
    The TF-IDF similarity score is: {tfidf_similitud:.2f}
    Determine if the ChatGPT solution achieves the same objective as the correct solution.
Provide ONLY the similarity percentage from 1-100 being 100 equal answers in content
and effectiveness of answer to the prompt.
    """
```

```
    for _ in range(num_responses):
        response = openai.ChatCompletion.create(
            model="gpt-3.5-turbo",
            messages=[
                {"role": "system", "content": "You are an assistant that evaluates the similarity between your answer and the correct answer, taking into account not only the similarity in text, but the similarity of the objective. Provide ONLY the percentage of similarity answer from 1-100."},
                {"role": "user", "content": confirmation_prompt}
            ],
            max_tokens=10
        )
        similarity_score = response.choices[0]['message']['content'].strip()
        similarity_scores.append(float(similarity_score))

    return sum(similarity_scores) / len(similarity_sc

def procesar_archivo(nombre_archivo):
    if not os.path.isfile(nombre_archivo):
        print(f"Error: El archivo '{nombre_archivo}' no existe.")
        return
    preguntas = []
    respuestas_correctas = []
    with open(nombre_archivo, 'r') as archivo:
        lineas = archivo.readlines()
        for i in range(0, len(lineas), 2):
            if lineas[i].startswith("input:"):
```

```
            pregunta = lineas[i].strip().replace("input:", "").strip()
            respuesta_correcta = lineas[i+1].strip().replace("answer:", "").strip()
            preguntas.append(pregunta)
            respuestas_correctas.append(respuesta_correcta)

    resultados = []
    for pregunta, respuesta_correcta in zip(preguntas, respuestas_correctas):
        respuesta_chatgpt = obtener_respuesta_chatgpt(pregunta)
        print(f"Pregunta: {pregunta}")
        print(f"Respuesta de ChatGPT: {respuesta_chatgpt}\n")
        similitud = calcular_similitud(respuesta_chatgpt, respuesta_correcta)
        confirmacion = confirmar_similitud_con_chatgpt(pregunta, respuesta_correcta,
respuesta_chatgpt)
        print(f"Confirmación de ChatGPT: {confirmacion}%\n")
        resultados.append({
            'pregunta': pregunta,
            'respuesta_chatgpt': respuesta_chatgpt,
            'respuesta_correcta': respuesta_correcta,
            'similitud': similitud,
            'confirmacion_chatgpt': confirmacion })
    calcular_media_por_bloques(resultados)
def calcular_media_por_bloques(resultados, bloque_tamano=100):
    num_bloques = (len(resultados) + bloque_tamano - 1) // bloque_tamano


    for i in range(num_bloques):

        inicio = i * bloque_tamano

        fin = inicio + bloque_tamano
```

```
        bloque = resultados[inicio:fin]
        if not bloque:
            break
        similitud_media = sum([resultado['similitud'] for resultado in bloque]) / len(bloque)
        print(f"{inicio}-{fin}: {similitud_media * 100:.2f}%")
def main():
    nombre_archivo = 'ruta del archivo'
    resultados = procesar_archivo(nombre_archivo)
```

Anexo 1.2: Caso para Gemini

```
import google.generativeai as genai
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import json
from typing_extensions import TypedDict

GOOGLE_API_KEY = userdata.get('Privada')
genai.configure(api_key=GOOGLE_API_KEY)

def obtener_respuesta_gemini(pregunta):
    gemini_llm = genai.GenerativeModel(
        model_name='gemini-1.5-flash',
        generation_config={"response_mime_type": "application/json"}
    )

    prompt = f"""
    You are a helpful coding creator. You are going to create a short code to fulfill the task of the following prompt: {pregunta}
    """
    response = gemini_llm.generate_content(prompt)
    return response.text.strip()

def calcular_similitud(respuesta_gemini, respuesta_correcta):
    vectorizer = TfidfVectorizer()
```

```
    tfidf_matrix = vectorizer.fit_transform([respuesta_gemini, respuesta_correcta])
    similitud = cosine_similarity(tfidf_matrix[0:1], tfidf_matrix[1:2])
    return similitud[0][0]

def calcular_media_por_bloques(resultados, bloque_tamano=100):
    num_bloques = (len(resultados) + bloque_tamano - 1) // bloque_tamano
    for i in range(num_bloques):
        inicio = i * bloque_tamano
        fin = inicio + bloque_tamano
        bloque = resultados[inicio:fin]
        if not bloque:
            break
        similitud_media = sum([resultado['similitud'] for resultado in bloque]) / len(bloque)
        print(f"Bloque {i+1} ({inicio}-{fin}): {similitud_media * 100:.2f}% de similitud")

def procesar_archivo(nombre_archivo):
    preguntas = []
    respuestas_correctas = []
    with open(nombre_archivo, 'r') as archivo:
        lineas = archivo.readlines()
        for i in range(0, len(lineas), 2):
            if lineas[i].startswith("input:"):
                pregunta = lineas[i].strip().replace("input:", "").strip()
                respuesta_correcta = lineas[i+1].strip().replace("answer:", "").strip()
                preguntas.append(pregunta)
                respuestas_correctas.append(respuesta_correcta)
```

```
    resultados = []
    for pregunta, respuesta_correcta in zip(preguntas, respuestas_correctas):
        respuesta_gemini = obtener_respuesta_gemini(pregunta)
        similitud = calcular_similitud(respuesta_gemini, respuesta_correcta)
        resultados.append({
            'pregunta': pregunta,
            'respuesta_gemini': respuesta_gemini,
            'respuesta_correcta': respuesta_correcta,
            'similitud': similitud
        })

    calcular_media_por_bloques(resultados)

def main():
    nombre_archivo = '/Users/Marco/Desktop/BD/Electrical-egineering_CORTA.txt'
    procesar_archivo(nombre_archivo)

if __name__ == "__main__":
    main()
```

Anexo 1.3: Muestra de la base de datos

input: You are an electrical engineer and you will answer questions related to electrical engineering. What is Ohm's law?

answer: Ohm's law states that the electric current through a conductor between two points is directly proportional to the voltage across the two points. It is represented by the equation V = RI, where V is the voltage, R is the resistance, and I is the current.

input: You are an electrical engineer and you will answer questions related to electrical engineering. How does a transformer work?

answer: A transformer works on the principle of electromagnetic induction and is used to change the voltage level of alternating current (AC). It consists of two coils: the primary and secondary, which are not electrically connected but linked by a magnetic field. When AC flows through the primary coil, it creates a varying magnetic field, inducing a voltage in the secondary coil. The voltage change between the primary and secondary coils depends on the ratio of the number of turns in each coil.

input: You are an electrical engineer and you will answer questions related to electrical engineering. What are the differences between AC and DC currents?

answer: AC (Alternating Current) and DC (Direct Current) are two types of electrical currents. AC current changes direction periodically, whereas DC current flows in one direction only. AC is commonly used for power distribution because it is less costly to transmit over long distances and can easily be transformed to different voltages. DC is often used in batteries, electronics, and solar power systems, as it provides a constant voltage or current.

Anexo 2: Programación python base de pruebas para análisis de sentimiento

Anexo 2.1: Caso Open AI (ChatGPT3.5, ChatGPT4)

```
import pandas as pd

import openai

file_path = ‘ruta del archivo’

data = pd.read_excel(file_path)

openai.api_key = 'PRIVADA'


phrases = data.iloc[:, 1].tolist()

correct_labels = data.iloc[:, 2].tolist()


def get_sentiment(phrase):

    response = openai.Completion.create(

        engine="gpt-3.5-turbo",

        prompt=f"Analiza el sentimiento de la siguiente frase y devuelve ÚNICAMENTE 'positivo', 'neutral' o 'negativo': {phrase}",

        max_tokens=150

    )

    sentiment = response.choices[0].text.strip().lower()

    return sentiment


def sentiment_to_score(sentiment):
```

```
    if sentiment == 'positivo':
        return 2
    elif sentiment == 'neutral':
        return 1
    elif sentiment == 'negativo':
        return 0
    else:
        return -1
predicted_scores = [sentiment_to_score(get_sentiment(phrase)) for phrase in phrases]
def calculate_accuracy(predicted, correct):
    correct_predictions = sum(p == c for p, c in zip(predicted, correct))
    return correct_predictions / len(correct)

accuracies = []
for i in range(0, len(predicted_scores), 100):
    interval_scores = predicted_scores[i:i + 100]
    interval_correct = correct_labels[i:i + 100]
    interval_accuracy = calculate_accuracy(interval_scores, interval_correct)
    accuracies.append(interval_accuracy)
overall_accuracy = calculate_accuracy(predicted_scores, correct_labels)

report = f"Informe de Precisión del Análisis de Sentimiento:\n\n"
```

```
for i, accuracy in enumerate(accuracies):
    report += f"Intervalo {i*100} - {(i+1)*100}: {accuracy * 100:.2f}%\n"

report += f"\nPrecisión Media Final: {overall_accuracy * 100:.2f}%\n"

print(report)
```

Anexo 2.2: Caso para Gemini

```
import google.generativeai as genai
import pandas as pd
import time
import os

GOOGLE_API_KEY = 'Privada'
genai.configure(api_key=GOOGLE_API_KEY)

def get_sentiment(phrase):
    gemini_llm = genai.GenerativeModel(
        model_name='gemini-1.5-flash-latest',
        generation_config={"response_mime_type": "application/json"}
    )

    prompt = f"""
    You are a sentiment analysis assistant. Analyze the sentiment of the following phrase and ONLY return 'positive', 'neutral', or 'negative': {phrase}
    """

    while True:
        try:
            response = gemini_llm.generate_content(prompt)
            sentiment = response.text.strip().lower()
            return sentiment
        except genai.exceptions.RateLimitError:
            print("Rate limit exceeded. Waiting for 60 seconds before retrying...")
```

```
            time.sleep(60)
        except Exception as e:
            print(f"An error occurred: {e}")
            return None


file_path = '/Users/Marco/Desktop/BD/Sentiment/BUENAS/Sentiment2.xlsx'
data = pd.read_excel(file_path)
print(data.head())

phrases = data.iloc[:, 1].fillna('').astype(str).tolist()
correct_labels = data.iloc[:, 2].fillna('').astype(str).tolist()

predicted_sentiments = []
for i, phrase in enumerate(phrases):
    sentiment = get_sentiment(phrase)
    predicted_sentiments.append(sentiment)
    if i < 5:
        print(f"Phrase: {phrase} | Predicted Sentiment: {sentiment}")

def calculate_accuracy(predicted, correct):
    correct_predictions = sum(1 for p, c in zip(predicted, correct) if c in p)
    return correct_predictions / len(correct)

overall_accuracy = calculate_accuracy(predicted_sentiments, correct_labels)

report = f"Sentiment Analysis Accuracy Report:\n\n"
```

```
report += f"\nFinal Average Accuracy: {overall_accuracy * 100:.2f}%\n"

print(report)
```

Anexo 2.3: Muestra base de pruebas primer análisis de sentimiento

Just finished watching the series return of the Chasers War on Everything - really quite **** compared to how it used to be	negative
My go-to todo list app. I'd love for the habits feature to be improved, though.	positive
Were getting old Donny. I got one starting high school next yr and one going into the 7th. Where have the years gone?!	neutral
Oh god, a moth was living in a **** power outlet! (Actually, my PowerSquid.)	negative
yy	neutral
Great simple app. I wish it easily sync'd across devices. I also would love a calendar view I can see/print so I can visually see my progress over time.	positive
good as, cya in melbourne	positive
Requires account to use. I was just giving a simple reason why I didn't like it. But since your response indicates I don't have 10 seconds, then I'll waist 5 minutes to make you understand. The point is that it "Requires An Account"! I don't need another stinking account to keep track of a task list. If I wanted to share on multiple devices, I would just use any number of desktop applications that I already own. I don't need another stinking program tracking my business in the sake of "backing up my data' or make it available everywhere. I have that covered already. Just wanted a simple task list for a small price to use on my phone. But without that choice, no thanks. To act as though 10 seconds to give up my privacy is no big deal, well my information and data is not your business or any one else unless I decide I want to do that. That is what you need to understand, some customers like to have options to do something and not have something stuffed down their throat. Downloaded "Tasks: Todo List" and been happy with it. Donated $1.99 to developer for making app for me and not their self.	negative
Very bad app. I organised all of my activities, took me 1 hour to complete my planner, then realised that the app automatically sets the time at am, so all my afternoon activities where at the morning, for example if I had to eat at 1pm it said that I had to eat at 1am, so I changed them to pm and suddenly the app errased my whole schedule. WASTING 1 HOUR	negative
Over-thought tying my shoe, couldn`t figure it out for a while	negative
And so very you. You know I say it w/ nothing but love, dude.	positive
It's a bit redundant when dealing with recurrent tasks, doesn't allow scheduling those recurrent tasks to specific times of the day or set an expiry date for the recurrence, but otherwise quite functional and useful.	positive

Anexo 2.4: Muestra base de pruebas segundo análisis de sentimiento

My dog is officially depressed that my brother`s dogs are gone. He doesn`t want to go outside and when we did, he play half-heartedly.	negative
I just realized that I can`t forward text msgs with my iPhone.	neutral
My frist post... Off to find a new car for my parents, exciting!	positive
Argh noo! Missed The Killers on Wossy! That sucks! Missed out on Brandon. Total failure! Anyone know if it`s repeated? Must investigate!	negative
seriously bored without anyone to talk to... but not tired enough for sleep	negative
I am lost. Please help me find a good home.	negative
It`s a Peter & Gordon morning -> And I, go to pieces and I wanna hide / Go to pieces and I almost die / Ever... ? http://blip.fm/~5yk38	neutral
Ok so I`ve now got a bit of a bad back after lifting all drum hardware into my car downer.	negative
LMAO... Smh! that one threw me off.	neutral
oh nice going!	positive
Is getting ready for work... Working all weekend	neutral
Gonna celebrate Mothers Day with the family but gonna start the partying tonite	positive
i agree with you!	positive
I only do computers. Am hopeless at everything else	negative
90 degrees, gross skies, and thunderstorms...perfect match for my mood lol	positive
_2nd aww thanks!	positive
Sorry RB is on PS3 for me	negative
I saw amazing heeels. But they were too big	neutral
I just stuck my finger down my throat and there are a bunch of bumps on my tongue & throat.	negative
dang last url went down ? http://blip.fm/~7aigm	negative
ohoh i missed all ur tweets im gonna have to stay awake all night to see the announcement now, **** time difference	negative
Damnit all. That sucks. You were one of the ones I thought I`d drag back lol	negative
my boss. She`s moving to NYC	neutral
this is sooo crazy i have fever..	negative
is spending her Saturday morning taking notes for a research essay because some stupid **** recalled the book I`m using. Not fair	negative
I think i need some new friends	neutral
Looking forward to your gig in Ireland!!! See ya there!	neutral

Anexo 3: Alineación con los objetivos de desarrollo sostenible.

Este proyecto enlaza muy estrechamente con el objetivo de desarrollo sostenible número 4: **Garantizar una educación inclusiva, equitativa y de calidad y promover oportunidades de aprendizaje durante toda la vida para todos.** La inteligencia artificial puede ser un medio muy poderoso para abaratar la educación y democratizarla. Entre sus capacidades se encuentra la de la escalabilidad, el coste de aumentar en una unidad los usuarios de esta es prácticamente nulo.

Indagando en los desarrollos de este objetivo número 4, lo que se busca es elevar el nivel de educación de la población siendo la inteligencia artificial una herramienta idónea para crear planes de educación, mejorar la personalización en la educación y ajustar la enseñanza para que eleve el nivel general de todos los estudiantes globales.

Un análisis cómo el que se va a realizar en este proyecto va a ser de ayuda para las entidades educativas para evaluar el rendimiento en las especialidades STEM del uso de la inteligencia artificial para la docencia a nivel universitario.

Tambien, el objetivo número 8 de los ODS enlaza con este proyecto, **Promover el crecimiento económico inclusivo y sostenible, el empleo y el trabajo decente para todos.**

El rol de la inteligencia artificial en el futuro va a ser mayúsculo, va a permitir que los trabajos más repetitivos puedan ser hechos con esta tecnología y va a permitir a las personas desarrollarse profesionalmente en áreas de más creatividad y menos rutinarias.

Este TFG va a tener un impacto en cuanto va a permitir a las empresas de ingeniería medir en que punto de madurez está la inteligencia artificial con respecto los desarrollos y soluciones que llevan a cabo estas empresas. Esta medición va a permitir llevar a cabo un estudio de automatización de ciertos roles que pueden liberar de carga laboral a los empleados.