



GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Ingeniería GPTizada: Explorando modelos de lenguaje
preentrenados para potenciar y simplificar tareas específicas
de ingeniería

Autor: Allende de Yarza González-Lacarra

Director: Alfonso Vázquez Requejo

Madrid

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

Ingeniería GPTizada: Explorando modelos de lenguaje preentrenados para potenciar y simplificar tareas específicas de ingeniería

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2024/25 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.: Allende de Yarza

Fecha: 06/ 07/ 2025

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO



Fdo.: Alfonso Vázquez Requejo

Fecha: ..6.../ ..07.../ 2025



GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Ingeniería GPTizada: Explorando modelos de lenguaje
preentrenados para potenciar y simplificar tareas específicas
de ingeniería

Autor: Allende de Yarza González-Lacarra

Director: Alfonso Vázquez Requejo

Madrid

Agradecimientos

A mi director Alfonso, por su ayuda para desarrollar este proyecto.

INGENIERÍA GPTIZADA: EXPLORANDO MODELOS DE LENGUAJE PREENTRENADOS PARA POTENCIAR Y SIMPLIFICAR TAREAS ESPECÍFICAS DE INGENIERÍA

Autor: de Yarza González-Lacarra, Allende

Director: Vázquez Requejo, Alfonso

Entidad Colaboradora: ICAI – Universidad Pontificia Comillas.

RESUMEN DEL PROYECTO

Este trabajo analiza el potencial de los modelos de lenguaje preentrenados (PLMs) en tareas de ingeniería. Se ha diseñado una metodología experimental que evalúa el rendimiento de modelos como GPT-2, GPT-4, Claude-3 y Gemini en cinco tareas representativas: explicación técnica, redacción de informes, resolución de problemas físicos, interpretación de imágenes y diagnóstico de procesos simulados. Los resultados obtenidos mediante métricas cuantitativas (TF-IDF y embeddings) y cualitativas evidencian que modelos como GPT-4o y Claude-3-Haiku destacan por su precisión, estructura y aplicabilidad práctica. Esta investigación demuestra que los LLMs son herramientas prometedoras para automatizar tareas instructivas y operativas en ingeniería.

Palabras clave: modelos de lenguaje, ingeniería, inteligencia artificial, GPT, evaluación de tareas, Simpy

1. Introducción

En las últimas décadas, la inteligencia artificial ha evolucionado de una tecnología experimental a una fuerza transformadora con impacto directo en el ámbito profesional. En particular, los modelos de lenguaje preentrenados, como GPT, Gemini o Claude, han demostrado capacidades avanzadas en el procesamiento del lenguaje natural (PLN), permitiendo generar texto técnico, interpretar instrucciones o analizar datos en lenguaje natural.

El campo de la ingeniería, tradicionalmente asociado al cálculo, la simulación y la documentación técnica, presenta múltiples oportunidades para integrar modelos LLM. Desde la redacción de informes hasta el diagnóstico de procesos, estos modelos podrían

automatizar tareas repetitivas o asistenciales, liberando tiempo para el análisis estratégico.

Este trabajo plantea una doble aproximación: por un lado, consolidar el marco conceptual de los PLMs, y por otro, evaluar empíricamente su aplicabilidad en tareas de ingeniería, con el objetivo de identificar oportunidades reales de uso.

2. Definición del Proyecto

El proyecto se estructura en dos fases diferenciadas pero complementarias. En primer lugar, se desarrolla un marco conceptual que permite entender los fundamentos y capacidades de los modelos de lenguaje preentrenados en el contexto de la inteligencia artificial y el procesamiento del lenguaje natural (PLN). Esta fase aborda aspectos como la evolución histórica de los modelos, sus arquitecturas más representativas y los componentes internos que los sustentan, incluyendo la tokenización, los embeddings, los mecanismos de atención y la arquitectura Transformer. Asimismo, se incluye una clasificación de los modelos según el tipo de entrada y salida que manejan, así como el tipo de arquitectura empleada. También se introducen conceptos actuales como la eficiencia computacional y la inteligencia aumentada.

En segundo lugar, se diseñan y ejecutan cinco experimentos orientados a evaluar el rendimiento de distintos modelos de lenguaje en tareas técnicas propias del ámbito de la ingeniería. Cada experimento se desarrolla bajo una metodología común que incluye la formulación de un prompt específico, generación por distintos modelos y su posterior evaluación tanto cuantitativa (mediante métricas como similitud TF-IDF o embeddings) como cualitativa (en términos de estructura, coherencia y adecuación técnica). Esta fase permite comparar el desempeño entre modelos y frente a respuestas oficiales, con el objetivo de determinar su utilidad en escenarios reales de aplicación.

3. Descripción del modelo/sistema/herramienta

Para la fase experimental se han utilizado modelos de lenguaje de última generación accesibles tanto mediante API, como es el caso de los modelos de OpenAI, Anthropic y Google, como de forma local, en concreto GPT-2. Las respuestas se han generado

utilizando scripts en Python que envían automáticamente prompts personalizados a cada modelo, almacenan las salidas generadas y evalúan su calidad a través de distintos métodos. La evaluación se ha llevado a cabo mediante dos enfoques complementarios: por un lado, la similitud literal, utilizando vectores TF-IDF combinados con la métrica de similitud del coseno; por otro, la similitud semántica, basada en embeddings que permiten comparar el significado de las respuestas más allá de las coincidencias textuales. Además, se ha empleado la librería `simpy` para simular procesos industriales y generar logs que los modelos debían interpretar. En el caso particular de GPT-2, al tratarse de un modelo descargado y ejecutado localmente, no ha sido necesario utilizar una API, lo que ha permitido ilustrar de forma directa los avances tecnológicos en el campo de los modelos de lenguaje desde su lanzamiento hasta la actualidad.

4. Resultados

A continuación se muestra un resumen de los modelos que han obtenido el mejor rendimiento por tipo de tarea:

Tarea evaluada	Mejor modelo	Métrica destacada	Motivo
Explicación de conceptos técnicos (AC)	GPT-4o	TF-IDF (0.8303) / embeddings (0.8646)	Claridad, precisión, redacción académica
Redacción de informe técnico (laboratorio)	GPT-4o	TF-IDF (0.9191) / embeddings (0.7894)	Estructura, análisis, coherencia
Problema físico clásico (cálculo)	GPT-4 / Claude-3.5	embeddings (0.8554 / 0.8513)	Desarrollo correcto y razonamiento físico completo
Problema físico con imagen (multimodal)	GPT-4o	embeddings (0.8341)	Comprensión de la imagen y formulación coherente
Diagnóstico de proceso simulado (simpy)	GPT-4 / GPT-4o	embeddings (0.8297 / 0.8059)	Identificación del fallo y explicación técnica precisa

Evolución histórica (GPT-2 vs actual)	Comparación	TF-IDF (~0.24 vs >0.75)	Ilustra el progreso en generación técnica y semántica
--	--------------------	-------------------------	---

Resultados más destacados

5. Conclusiones

El estudio confirma que los modelos de lenguaje actuales, en particular GPT-4o y Claude-3-Haiku, poseen un nivel de madurez que permite abordar tareas técnicas complejas con un alto grado de calidad. En tareas como la redacción de informes o el diagnóstico de procesos, estos modelos pueden actuar como asistentes efectivos, siempre bajo revisión humana.

La inclusión de GPT-2 ha sido fundamental para visualizar de forma concreta el progreso en capacidad explicativa, precisión semántica y eficiencia, reforzando la idea de que los modelos contemporáneos son significativamente más útiles en contextos técnicos.

6. Referencias

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser y I. Polosukhin, “*Attention is All You Need*”, in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[2] OpenAI, “*GPT-4 Technical Report*”, OpenAI, 2023. [Online]. Available: <https://openai.com/research/gpt-4>

[3] J. Alammar, “*The Illustrated Transformer*”, 2018. [Online]. Available: <https://jalammar.github.io/illustrated-transformer>

[4] F. Pedregosa et al., “*Scikit-learn: Machine Learning in Python*”, *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[5] OpenAI, “*OpenAI API Documentation*”. [Online]. Available: <https://platform.openai.com/docs>

[6] SimPy, “*SimPy 4.0.1 Documentation*”. [Online]. Available: <https://simpy.readthedocs.io/en/latest/>

GPT-ENGINEERED: EXPLORING PRETRAINED LANGUAGE MODELS TO ENHANCE AND SIMPLIFY ENGINEERING-SPECIFIC TASKS.

Author: de Yarza González-Lacarra, Allende

Supervisor: Vázquez Requejo, Alfonso.

Collaborating Entity: ICAI – Universidad Pontificia Comillas.

PROJECT SUMMARY

This project analyzes the potential of pretrained language models (PLMs) in engineering tasks. An experimental methodology has been designed to evaluate the performance of models such as GPT-2, GPT-4, Claude-3, and Gemini across five representative tasks: technical explanation, academic report writing, physics problem solving, image interpretation, and diagnosis of simulated processes. The results, obtained through both quantitative (TF-IDF and embeddings) and qualitative metrics, show that models like GPT-4o and Claude-3-Haiku stand out for their accuracy, structure, and practical applicability. This research demonstrates that LLMs are promising tools for automating instructional and operational tasks in engineering.

Keywords: language models, engineering, artificial intelligence, GPT, task evaluation, Simpy

1. Introduction

In recent decades, artificial intelligence has evolved from an experimental technology into a transformative force with a direct impact on professional environments. Pretrained language models, such as GPT, Gemini, or Claude, have shown advanced capabilities in natural language processing (NLP), enabling them to generate technical text, interpret instructions, or analyze data written in natural language.

The field of engineering—traditionally associated with calculation, simulation, and technical documentation—presents numerous opportunities to integrate LLMs. From report writing to process diagnostics, these models could automate repetitive or support tasks, freeing up time for strategic analysis.

This project takes a dual approach: on one hand, it consolidates the conceptual framework of PLMs; on the other, it empirically evaluates their applicability in engineering tasks, with the goal of identifying real opportunities for use.

2. Project Definition

The project is structured into two complementary phases. First, a conceptual framework is developed to understand the foundations and capabilities of pretrained language models in the context of artificial intelligence and NLP. This phase addresses topics such as the historical evolution of models, their most representative architectures, and internal components, including tokenization, embeddings, attention mechanisms, and the Transformer architecture. A classification of models is also provided based on input/output types and architecture, along with current concepts like computational efficiency and augmented intelligence.

Second, five experiments are designed and executed to evaluate the performance of various language models in technical tasks relevant to engineering. Each experiment follows a common methodology that includes the formulation of a specific prompt, generation by different models, and subsequent evaluation using both quantitative (TF-IDF similarity or embeddings) and qualitative (structure, coherence, and technical adequacy) criteria. This phase enables comparison across models and with official reference responses, in order to assess their usefulness in real-world applications.

3. Description of the System/Tool

The experimental phase used state-of-the-art language models available via API, such as those from OpenAI, Anthropic, and Google, as well as a local model (GPT-2). Python scripts were used to automatically send tailored prompts to each model, store the generated outputs, and evaluate their quality using different methods. Evaluation was conducted using two complementary approaches: literal similarity (based on TF-IDF vectors and cosine similarity), and semantic similarity (using embeddings to capture meaning beyond surface-level text). Additionally, the `simpy` library was used to simulate industrial processes and generate logs that the models were required to interpret. GPT-

2, downloaded and executed locally, served to illustrate the technological advancements in language models from its release to the present.

4. Results

Below is a summary of the top-performing models for each task:

Evaluated Task	Best Model	Highlight Metric	Key Reason
Technical concept explanation (AC)	GPT-4o	TF-IDF (0.8303) / embeddings (0.8646)	Clarity, precision, academic-style writing
Technical report writing (lab)	GPT-4o	TF-IDF (0.9191) / embeddings (0.7894)	Structure, analysis, coherence
Classical physics problem (calculus)	GPT-4 / Claude-3.5	embeddings (0.8554 / 0.8513)	Correct reasoning and complete physical analysis
Image-based physics problem (multimodal)	GPT-4o	embeddings (0.8341)	Visual understanding and coherent formulation
Simulated process diagnosis (Simpy)	GPT-4 / GPT-4o	embeddings (0.8297 / 0.8059)	Fault detection and precise technical explanation
Historical evolution (GPT-2 vs. current)	Comparison	TF-IDF (~0.24 vs >0.75)	Illustrates progress in technical and semantic generation

Summary of the most relevant results

5. Conclusions

The study confirms that today’s language models—particularly GPT-4o and Claude-3-Haiku—have reached a level of maturity that allows them to tackle complex technical tasks with high quality. In activities such as report writing or process diagnostics, these models can serve as effective assistants, always under human supervision.

The inclusion of GPT-2 proved crucial for visualizing the concrete progress in explanatory capability, semantic precision, and efficiency—reinforcing the idea that contemporary models are significantly more valuable in technical contexts.

1. Referencias

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser y I. Polosukhin, “*Attention is All You Need*”, in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[2] OpenAI, “*GPT-4 Technical Report*”, OpenAI, 2023. [Online]. Available: <https://openai.com/research/gpt-4>

[3] J. Alammar, “*The Illustrated Transformer*”, 2018. [Online]. Available: <https://jalammar.github.io/illustrated-transformer>

[4] F. Pedregosa et al., “*Scikit-learn: Machine Learning in Python*”, *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[5] OpenAI, “*OpenAI API Documentation*”. [Online]. Available: <https://platform.openai.com/docs>

[6] SimPy, “*SimPy 4.0.1 Documentation*”. [Online]. Available: <https://simpy.readthedocs.io/en/latest/>

Índice de la memoria

<i>Índice de tablas</i>	<i>IV</i>
Capítulo 1. Introducción	5
1.1 Contexto y Motivación del proyecto	5
1.2 Objetivos del proyecto	6
1.3 Alineación con los ODS	7
1.4 Descripción de las tecnologías empleadas	8
1.5 Estructura del proyecto	9
Capítulo 2. Estado de la cuestión	10
2.1 Investigaciones actuales	10
2.2 Justificación del proyecto	12
Capítulo 3. Marco teórico	14
3.1 Fundamentos de la Inteligencia Artificial	14
3.2 Procesamiento del lenguaje natural (PLN)	15
3.3 Modelos de lenguaje preentrenados (PLMs)	25
3.4 Augmenting AI: Inteligencia aumentada	31
Capítulo 4. Definición del trabajo experimental	33
4.1 Objetivos del desarrollo experimental	33
4.2 Descripción de los experimentos realizados	34
Capítulo 5. Desarrollo experimental	38
5.1 Experimento 01: Explicación de concepto técnico	38
5.2 Experimento 02: Redacción de informe de laboratorio	56
5.3 Experimento 03: Problema de física	58
5.4 Experimento 04: Problema con input de imagen	60
5.5 Experimento 05: Evaluación de un proceso simulado	65
Capítulo 6. Análisis de resultados	70
6.1 Evaluación experimento 01	70
6.2 Evaluación experimento 02	77
6.3 Evaluación experimento 03	82

6.4	Evaluación experimento 04	85
6.5	Evaluación experimento 05	87
Capítulo 7.	<i>Conclusiones y trabajos futuros.....</i>	90
Capítulo 8.	<i>Bibliography</i>	93
ANEXO I		96

Índice de figuras

Ilustración 1. Ejemplo de tokenización en subpalabras aplicado a la la palabra incomprensiblemente.....	17
Ilustración 2. Representación tridimensional de embeddings de palabras agrupadas por proximidad [16]	19
Ilustración 3. Esquema simplificado Red Neuronal Recurrente (RNN) [18].....	20
Ilustración 4. Ejemplo de funcionamiento de la función softmax [24]	21
Ilustración 5. Arquitectura Transformer Encoder - Decoder [19].....	23
Ilustración 6. Representación esquemática de un Transformer encoder - decoder para tareas de traducción [20].....	25
Ilustración 7. Imagen Problema magnetismo 1	61

Índice de tablas

Tabla 1. Clasificación de modelos de lenguaje según tipo de input, output y arquitectura	30
Tabla 2. Comparativa de Modelos de Lenguaje utilizados en la parte experimental	31
Tabla 3. Resultados tiempos de respuesta experimento 01	71
Tabla 4. Resultados de evaluación con TF-IDF experimento 01, prueba 1	72
Tabla 5. Resultados de evaluación contextualizada experimento 01, prueba 1	74
Tabla 6. Resultados de evaluación con TF-IDF experimento 01, prueba 2	77
Tabla 7. Resultados de evaluación contextualizada experimento 01, prueba 2	77
Tabla 8. Resultados tiempos de respuesta experimento 02	78
Tabla 9. Resultados de evaluación TF-IDF y contextualizada experimento 02, prueba 1	79
Tabla 10. Resultados de evaluación TF-IDF y contextualizada experimento 02, prueba 2	81
Tabla 11. Resultados tiempos de respuesta experimento 03	83
Tabla 12. Resultados de evaluación TF-IDF y contextualizada experimento 03, prueba 1	83
Tabla 13. Resultados de evaluación TF-IDF y contextualizada experimento 03, prueba 2	84
Tabla 14. Resultados tiempos de respuesta experimento 04	86
Tabla 15. Resultados de evaluación TF-IDF y contextualizada experimento 04, prueba 1	86
Tabla 16. Resultados de evaluación TF-IDF y contextualizada experimento 04, prueba 2	87
Tabla 17. Resultados tiempos de respuesta experimento 05	88
Tabla 18. Resultados de evaluación TF-IDF y contextualizada experimento 05, prueba 1	88

Capítulo 1. INTRODUCCIÓN

1.1 CONTEXTO Y MOTIVACIÓN DEL PROYECTO

En los últimos años, la inteligencia artificial (IA) ha dejado de ser una tecnología emergente para convertirse en uno de los principales motores de transformación digital a nivel global. Su capacidad para automatizar procesos, optimizar decisiones y generar contenido ha permitido su integración en múltiples sectores, desde la medicina y la educación hasta la industria o el entretenimiento. Dentro de esta revolución, la inteligencia artificial generativa, y en particular los modelos de lenguaje preentrenados, ha cobrado un protagonismo destacado por su capacidad para producir lenguaje natural con un nivel de coherencia y sofisticación tan cercano al razonamiento humano que comienza a alterar nuestras propias nociones sobre qué es “pensar” o qué es ser “inteligente”.

En el contexto de la IA generativa, el desarrollo de modelos de lenguaje como GPT-4.5 (OpenAI), Claude 4 (Anthropic), Gemini 2.5 Pro (Google) o LLaMA 3 (Meta) marca un antes y un después en el campo. Estos modelos ya no responden a preguntas simples o generan texto de manera general, sino que comienzan a tener capacidad de razonamiento, de resolución de problemas y de adaptación a contextos complejos. Esta tendencia la subrayan informes como el de Morgan Stanley [1], que destaca la inversión reciente de las grandes tecnológicas en dotar a sus modelos de razonamiento adaptado a entornos empresariales exigentes.

Centrándonos en el ámbito de la ingeniería, estas tecnologías abren un abanico de posibilidades aún poco exploradas: desde la generación automática de documentación técnica hasta la asistencia en la resolución de problemas complejos, la evaluación de procesos simulados o la automatización de tareas repetitivas, dejando más tiempo para que los ingenieros se dediquen a tareas de mayor valor. Sin embargo, y a pesar del potencial evidente, la aplicación rigurosa y sistemática de los modelos de lenguaje en contextos técnicos reales sigue siendo limitada. Estudios recientes como el de Akolekar et al. [2] han

demostrado que, si bien estos modelos muestran un rendimiento sobresaliente en tareas generales, su desempeño decrece significativamente al enfrentarse a problemas numéricos o multietapa, característicos del entorno ingenieril.

Ante esta situación, el presente trabajo nace con la motivación de contribuir a llenar ese vacío. A través de una combinación de análisis teórico y experimentación práctica, se pretende evaluar de manera comparativa el rendimiento de distintos modelos de lenguaje en tareas propias de la ingeniería, con el objetivo de identificar sus fortalezas, limitaciones y oportunidades reales de aplicación. Esta línea de trabajo, además de alinearse con las tendencias más actuales de la IA, responde también al interés de la autora por comprender cómo estas tecnologías pueden transformar el ejercicio profesional de la ingeniería en los próximos años.

1.2 OBJETIVOS DEL PROYECTO

Yendo de lo general a lo específico, el objetivo general de este proyecto es proporcionar y documentar una base sólida sobre los fundamentos de los modelos de lenguaje, que se recogerán en el marco teórico, y analizar y evaluar de forma práctica el rendimiento de diferentes modelos de lenguaje en distintas tareas del campo de la ingeniería. El propósito es comprender su funcionamiento, explorar su aplicabilidad real y valorar su potencial como herramienta de apoyo en este ámbito.

Este proyecto también persigue una serie de objetivos más específicos:

- Recoger y estructurar los principales conceptos teóricos relacionados con los modelos de lenguaje preentrenados, incluyendo su evolución, funcionamiento, arquitectura y tipologías actuales.
- Diseñar y ejecutar cinco experimentos prácticos que permitan evaluar la utilidad de estos modelos en tareas concretas de la ingeniería, como la redacción de informes técnicos, la resolución de problemas o el análisis de procesos simulados.

- Detectar fortalezas, puntos débiles y límites actuales en el uso de estos modelos en el ámbito ingenieril, así como identificar oportunidades de aplicación y proponer áreas donde sea coherente integrarlos.
- Generar conclusiones que sirvan de base para futuras investigaciones y desarrollos relacionados con la aplicación de inteligencia artificial generativa en ingeniería, contribuyendo al debate sobre el papel que deberían desempeñar estas herramientas en el entorno técnico y profesional.

1.3 ALINEACIÓN CON LOS ODS

Este trabajo se alinea con varios Objetivos de Desarrollo Sostenible (ODS) definidos por la Agenda 2030 de las Naciones Unidas, contribuyendo a fomentar un desarrollo tecnológico más responsable, inclusivo y eficiente. En primer lugar, está relacionado con el **ODS 9**, centrado en industria, innovación e infraestructura, ya que la investigación y aplicación de modelos de lenguaje preentrenados en el ámbito de la ingeniería impulsa la modernización de procesos industriales y la adopción de soluciones tecnológicas. La incorporación de estas herramientas puede mejorar la eficiencia operativa, acelerar el desarrollo de soluciones técnicas y abrir nuevas oportunidades en términos de automatización y precisión.

Asimismo, este trabajo también contribuye al **ODS 8**, que promueve el trabajo decente y el crecimiento económico. La automatización de tareas repetitivas mediante la IA libera tiempo para que los profesionales del sector de la ingeniería puedan centrarse en actividades de mayor valor añadido, que requieran capacidades por el momento exclusivamente humanas. Esta redistribución del esfuerzo fomenta un entorno laboral más cualificado, impulsa la productividad y la creatividad, y abre nuevas oportunidades vinculadas al desarrollo, implementación y supervisión de las tecnologías emergentes.

Finalmente, también se alinea con el **ODS 12**, relativo a producción y consumo responsables. La aplicación de modelos de lenguaje puede contribuir a optimizar procesos, reducir errores, minimizar retrabajos y favorecer una toma de decisiones más fundamentada. Todo ello

permite un uso más eficiente de los recursos, lo que resulta especialmente importante en un contexto que demanda una producción más sostenible.

1.4 DESCRIPCIÓN DE LAS TECNOLOGÍAS EMPLEADAS

Para el desarrollo de la parte experimental han sido esenciales las siguientes herramientas. En primer lugar, la programación se ha realizado en Python, en su versión 3.12, por su versatilidad y amplio ecosistema de librerías especializadas en inteligencia artificial y modelos de lenguaje.

Entre las librerías más utilizadas se encuentran transformers, que permite cargar y utilizar modelos de lenguaje desde distintos proveedores; torch, que proporciona estructuras y funciones para el trabajo con tensores y la ejecución eficiente de modelos de deep learning; y pandas, que facilita la manipulación, análisis y visualización de resultados. Todo el desarrollo se ha llevado a cabo en el entorno Jupyter Notebook.

Para acceder a modelos como GPT-3.5, GPT-4 y GPT-4o, se ha utilizado la API oficial de OpenAI, que permite enviar prompts y recibir respuestas de forma programática, con distintos niveles de personalización en los parámetros de generación. Asimismo, se han evaluado modelos de otros proveedores, como Claude 3 (Anthropic) y Gemini 1.5 (Google), accediendo a sus respuestas a través de sus respectivas APIs en la nube.

En el análisis de similitud semántica entre las respuestas generadas y las soluciones de referencia se han utilizado dos enfoques: uno mediante la librería transformers para calcular embeddings con modelos preentrenados, y otro mediante scikit-learn, utilizando la métrica del coseno para comparar vectores. Por último, para uno de los experimentos se ha empleado también la librería simpy, utilizada para simular procesos dinámicos y evaluar la capacidad de los modelos para interpretar salidas complejas en un entorno de simulación.

1.5 ESTRUCTURA DEL PROYECTO

Tras este capítulo introductorio, el trabajo se estructura en 6 capítulos adicionales que recogen el desarrollo completo del proyecto, desde el análisis del contexto hasta la evaluación final de los resultados y la propuesta de futuras líneas de investigación.

El **Capítulo 2, Estado de la cuestión**, corresponde al estudio del estado de la cuestión. En él se recogen los antecedentes relevantes para este trabajo y se revisan investigaciones previas relacionadas con el uso de modelos de lenguaje en tareas técnicas, con el objetivo de situar el proyecto dentro del contexto actual.

El **capítulo 3, Marco teórico**, desarrolla el marco teórico y tiene un doble objetivo. Por un lado, sirve como documentación de referencia, ya que recoge los fundamentos de los modelos de lenguaje preentrenados. Por otro, define y explica los conceptos necesarios para comprender adecuadamente la parte experimental que se presenta a continuación.

Los **capítulos 4, 5, 6 y 7** conforman el bloque dedicado al desarrollo práctico del trabajo. En el **capítulo 4, Definición del trabajo**, se describe la metodología experimental, detallando en qué consisten los experimentos planteados, cómo se estructura su ejecución y qué criterios se seguirán para su evaluación. El **capítulo 5, Desarrollo experimental**, presenta el desarrollo de los experimentos diseñados. A continuación, el **capítulo 6, Análisis de resultados**, analiza dichos resultados, interpretando su significado e implicaciones. Finalmente, el **capítulo 7, Conclusiones y trabajos futuros**, recoge las conclusiones generales del proyecto y plantea posibles líneas de mejora, extensión o investigación futura a partir de los descubrimientos obtenidos.

El **capítulo 8**, es la bibliografía del proyecto, donde se recogen todas las fuentes utilizadas en la investigación.

Capítulo 2. ESTADO DE LA CUESTIÓN

Antes de desarrollar este proyecto, fue imprescindible realizar una revisión sobre qué soluciones, estudios o desarrollos existen en el área de los modelos de lenguaje preentrenados y sus aplicaciones en el campo de la ingeniería. Esta revisión tiene un doble objetivo: por un lado, justificar la necesidad de llevar a cabo este trabajo, ya que no se ha profundizado suficientemente en este aspecto; y por otro, orientar el proyecto hacia aquellos puntos donde hay una mayor necesidad, de manera que se optimicen los resultados y se evite, en la medida de lo posible, redundar en lo que ya existe en la industria.

En el caso concreto de este trabajo, la cuestión es identificar si se han llevado a cabo (y en ese caso, cuáles y qué aspectos cubren) investigaciones que evalúen el rendimiento de estos modelos en tareas propias del ámbito de la ingeniería.

Este capítulo incluye una revisión de trabajos existentes, publicaciones académicas, informes técnicos y estudios en el área del procesamiento de lenguaje natural, con el objetivo de situar el proyecto en el marco actual e identificar los “gaps” o limitaciones que este trabajo puede ayudar a cubrir. Este análisis, como se ha señalado anteriormente, contribuye a enfocar el proyecto de manera que suponga un avance real en el estado del arte, abordando necesidades aún no resueltas. Asimismo, sirve de puente entre el planteamiento inicial y el desarrollo, tanto teórico como práctico, proporcionando una base sólida para las decisiones metodológicas que se tomarán en los siguientes capítulos.

2.1 INVESTIGACIONES ACTUALES

Resulta fundamental, en este capítulo, hacer un recorrido por las investigaciones que se han llevado a cabo en el campo de los modelos de lenguaje preentrenados (PLMs) y que evalúan su potencial. Para valorar la similitud con lo que se busca hacer en este proyecto, vamos a centrarnos sobre todo en cómo es el modelo de evaluación que se ha utilizado (si es

cualitativo o cuantitativo), si se ha realizado alguna variación de parámetros, cuántos modelos se han evaluado, etc.

Es cierto que existen investigaciones actuales relacionadas con la evaluación de la aplicabilidad de los modelos a tareas académicas o técnicas. No obstante, la mayoría de estas investigaciones se centran en contextos distintos a los de este proyecto y emplean un sistema de evaluación diferente. A continuación, se repasan algunos de los trabajos previos más relevantes, organizados por tipo de aplicación.

Resolución de exámenes tipo test

Una de las principales líneas de trabajo investigadas es la capacidad de modelos como GPT3.5 y GPT4 en la resolución de exámenes con preguntas de opción múltiple. Varios de estos exámenes pertenecen a grados universitarios de ingeniería. Por ejemplo, Frenkel y Emara [3] examinan el rendimiento de GPT4 frente a GPT3.5 en el examen *FE Mechanical*, concluyendo que GPT4 logra un 76 % de aciertos frente a su predecesor. A pesar del avance, los autores señalan que el modelo no alcanza los estándares requeridos para aprobar, sobre todo por no poder procesar imágenes o gráficos. Sin embargo, el escenario de respuesta tipo test es muy específico y no permite evaluar, como tal, la calidad de la respuesta del modelo.

En un estudio relacionado, Hope et al. [4] analizan la capacidad de ChatGPT para resolver problemas de estadística a nivel universitario. Los resultados mostraron que el modelo obtiene una tasa de éxito del 82 %, superando a muchos de los alumnos. Sin embargo, no es capaz de resolver tareas cualitativas o que impliquen comprensión de la materia, como diferenciar ciertos conceptos, lo que evidencia limitaciones a nivel conceptual que deben ser revisadas. Además, ambos estudios se centran en ejercicios cerrados que no permiten al modelo demostrar su capacidad explicativa o argumentativa, habilidades fundamentales en un contexto profesional real.

Clasificación y procesamiento de documentación técnica

Otro campo de aplicación de los PLMs relacionado con el contenido de este proyecto es el de la clasificación y el procesamiento de documentación técnica. Es uno de los ámbitos en los que se observa mayor potencial, ya que estas tareas son, en parte, susceptibles de automatización. Además, aplicar modelos de lenguaje en este tipo de procesos permite liberar a los ingenieros de tareas más repetitivas, facilitando que puedan centrarse en actividades de mayor valor añadido como el pensamiento crítico, la innovación o el desarrollo de nuevas ideas.

Existen varios estudios en este ámbito que resulta importante comentar aquí. En primer lugar, la investigación de Beltagy et al [5], en la que se desarrolló *SciBERT*, un modelo basado en BERT preentrenado específicamente sobre cuerpos de texto científicos. Como resultado, el modelo muestra mejoras significativas en tareas como la clasificación de resúmenes técnicos. De forma similar, estudios recientes han comparado modelos como *SciBERT*, *BioBERT* o *BlueBERT* en tareas de categorización de *abstracts* científicos, obteniendo valores altos de exactitud y F1-score. Sin embargo, estos estudios se centran en procesos puramente de clasificación o recuperación de documentos, no en la generación de un informe técnico completo como el que se desarrollará en uno de los experimentos de la parte práctica de este trabajo.

2.2 JUSTIFICACIÓN DEL PROYECTO

La creciente presencia de la inteligencia artificial generativa en entornos profesionales, ya introducida anteriormente, plantea la necesidad de entender qué aportan los modelos de lenguaje y cómo lo hacen en contextos técnicos como la ingeniería. Aunque no hay duda de que estas herramientas son eficaces en una amplia variedad de tareas, su rendimiento y aplicabilidad en situaciones que requieren precisión, razonamiento lógico, pensamiento crítico y estructura argumentativa siguen siendo temas de debate.

Por este motivo, surge la necesidad de evaluar su utilidad y desempeño real en tareas específicas del ámbito ingenieril, no solo desde un punto de vista funcional, sino también en

términos de precisión, claridad técnica y adecuación a la tarea solicitada. El desarrollo de este proyecto busca responder precisamente a esa necesidad: estudiar los fundamentos de los modelos de lenguaje desde un enfoque teórico para, posteriormente, analizar en detalle cómo se comportan distintos modelos en tareas representativas de la ingeniería.

La importancia de esta evaluación no reside únicamente en el análisis de sus capacidades actuales, aunque esto sea especialmente relevante, sino también en la identificación de su potencial futuro y en la valoración de oportunidades de uso a partir de los resultados obtenidos. Este proyecto busca identificar esas posibles palancas de aplicación, pero también sus debilidades y límites, con el fin de servir como base para fundamentar decisiones informadas sobre la integración de los modelos de lenguaje en trabajos y aplicaciones reales.

Además, el enfoque comparativo entre modelos con distintas arquitecturas proporciona una perspectiva de conjunto sobre sus capacidades, lo que puede resultar de gran utilidad para profesionales, investigadores o desarrolladores interesados en aplicar la inteligencia artificial generativa en procesos o tareas vinculadas al ámbito de la ingeniería.

La aplicabilidad de los modelos de lenguaje en ingeniería es amplia y diversa. Una de las más inmediatas es la generación de documentación técnica, automatizando memorias, informes y manuales. También destacan su capacidad para generar código en lenguajes como Python, MATLAB o C++, asistiendo en el desarrollo de scripts para simulaciones o análisis de datos. Otra aplicación relevante es la resolución asistida de problemas técnicos, incluyendo ecuaciones, diseños o estructuras lógicas. En fases tempranas de diseño o ideación, los LLMs también pueden emplearse para apoyar procesos de *brainstorming* y exploración de soluciones.

Este trabajo se orienta precisamente a evaluar estas aplicaciones mediante diferentes experimentos, comparando la calidad de las respuestas generadas por los modelos con un respuesta de fuente oficial, elaborada por expertos.

Capítulo 3. MARCO TEÓRICO

Este capítulo trata del marco teórico del proyecto y recoge el estado del arte en el campo de los modelos de lenguaje preentrenados, organizando la información de lo más general a lo más específico. Comienza explicando la inteligencia artificial, que es la disciplina más general a la que pertenecen estos modelos, y continúa con los fundamentos de su funcionamiento y arquitectura, hasta llegar a clasificaciones de modelos según diferentes criterios. Como se ha mencionado anteriormente, tiene el objetivo de documentar los fundamentos teóricos del campo de los modelos de lenguaje preentrenados y, a la vez, poner en contexto al lector para que pueda comprender el desarrollo experimental, ya que hay muchos conceptos sobre el funcionamiento de los modelos que es imprescindible comprender para poder interpretar los resultados de los experimentos realizados.

3.1 FUNDAMENTOS DE LA INTELIGENCIA ARTIFICIAL

La inteligencia artificial ha evolucionado de forma significativa desde sus inicios, pasando por distintas etapas y perspectivas. En sus primeras fases, se centró en el desarrollo de sistemas simbólicos basados en reglas lógicas, con ejemplos como los lenguajes LISP o PROLOG y sistemas expertos como MYCIN [1]. Con el tiempo, estas aproximaciones dieron paso al aprendizaje automático o *machine learning*, que introdujo algoritmos capaces de aprender patrones a partir de datos [2]. A finales del siglo XX y comienzos del XXI, el auge del aprendizaje profundo o *deep learning* permitió mejorar considerablemente el rendimiento en tareas como el reconocimiento de imágenes o el procesamiento del lenguaje natural [3].

Un hito determinante en esta evolución fue la aparición del modelo *Transformer* (Vaswani et al., 2017), presentando el artículo *Attention is all you need* [4], que revolucionó el procesamiento del lenguaje al permitir paralelizar la atención contextual en secuencias largas. Esta arquitectura dio lugar a una nueva generación de modelos de lenguaje, conocidos como PLMs (*Pretrained Language Models*), que son los protagonistas de este trabajo.

Desde una perspectiva funcional, la IA puede clasificarse en varios niveles. La IA estrechao débil, que corresponde a los sistemas actuales, está diseñada para resolver tareas específicas sin conciencia ni comprensión general [5]. Por otro lado, la IA general o fuerte se refiere a sistemas hipotéticos capaces de igualar la inteligencia humana en una amplia gama de tareas cognitivas [6]. Finalmente, la superinteligencia artificial representa una fase aún más avanzada, en la que las máquinas superarían a los humanos en todos los aspectos [7].

En cuanto al aprendizaje, existen tres enfoques principales: el aprendizaje supervisado, basado en datos etiquetados; el no supervisado, que descubre patrones sin guía previa; y el aprendizaje por refuerzo, que optimiza decisiones a partir de recompensas [8]. La IA generativa, foco de este trabajo, se apoya fundamentalmente en el aprendizaje profundo y el modelado de secuencias, y permite generar contenido original como texto, código o imágenes [9].

3.2 PROCESAMIENTO DEL LENGUAJE NATURAL (PLN)

El procesamiento del lenguaje natural (PLN) es una subdisciplina de la inteligencia artificial que permite a las máquinas interpretar, analizar, generar y comprender el lenguaje humano de manera automatizada. Su desarrollo ha sido fundamental para la creación de modelos de lenguaje avanzados, como los que se estudian en este proyecto, al facilitar la interacción fluida entre humanos y sistemas informáticos.

Para que un modelo de inteligencia artificial pueda operar sobre texto, este debe transformarse en un formato que sea interpretable computacionalmente. Esta transformación se realiza a través de un conjunto de procesos que convierten las palabras o frases en representaciones numéricas que pueden ser manipuladas por una red neuronal. Este pipeline comienza con una etapa fundamental: la **tokenización**, que divide el texto en unidades mínimas denominadas *tokens*. Estas unidades pueden ser palabras completas, subpalabras o incluso caracteres individuales, dependiendo del enfoque utilizado por el modelo [10].

Una vez tokenizado el texto, los modelos generan vectores de representación llamados **embeddings**, que sitúan cada token en un espacio vectorial multidimensional. En este

espacio, las distancias entre los vectores reflejan relaciones semánticas: palabras con significados similares tienden a tener representaciones cercanas [11].

Estos procesos, tokenización y generación de embeddings, permiten a los modelos construir una comprensión interna del lenguaje que sirve de base para tareas como la generación automática de texto, la clasificación semántica o la resolución de problemas. En las siguientes secciones se explican en detalle los mecanismos detrás de estas transformaciones, así como las arquitecturas que las hacen posibles.

3.2.1 TOKENIZACIÓN

La tokenización es el primer paso en el procesamiento del lenguaje natural. Consiste en dividir un texto en unidades más pequeñas llamadas *tokens*, que pueden ser palabras, subpalabras o incluso caracteres. Esta segmentación permite que los modelos de lenguaje operen sobre el texto, ya que cada token podrá posteriormente ser transformado en un vector numérico procesable.

En sus formas más simples, la tokenización se realizaba palabra por palabra, separando el texto en función de espacios o signos de puntuación. Sin embargo, este enfoque presenta limitaciones en lenguas con alta flexión morfológica, vocabularios extensos o caracteres especiales. Para solucionar estos problemas, los modelos modernos utilizan técnicas de tokenización más sofisticadas, como la basada en subpalabras.

Entre los métodos más conocidos destacan Byte Pair Encoding (BPE) [12], WordPiece y SentencePiece. Estos algoritmos dividen palabras en fragmentos reutilizables que permiten representar vocabularios mucho más compactos y flexibles. Por ejemplo, como se representa en la Ilustración 1, el término “incomprensiblemente” podría dividirse en los subtokens “in”, “compren”, “sible” y “mente”, que pueden luego recombinarse para formar otras palabras [13].

Este enfoque reduce significativamente el número de palabras desconocidas (*out-of-vocabulary*) y mejora el rendimiento del modelo, ya que es capaz de entender y generar una mayor variedad de expresiones a partir de un número limitado de subunidades.



Ilustración 1. Ejemplo de tokenización en subpalabras aplicado a la la palabra incompresiblemente

3.2.2 REPRESENTACIÓN VECTORIAL DEL LENGUAJE: EMBEDDINGS

Una vez que el texto ha sido tokenizado, el siguiente paso es transformar cada token en una representación numérica que el modelo pueda interpretar y procesar. Esta representación se denomina embedding, y constituye una base fundamental para que los modelos de lenguaje comprendan el significado de las palabras y puedan operar sobre él computacionalmente.

Embeddings clásicos: del one-hot a Word2Vec

Los primeros enfoques representaban cada palabra mediante vectores binarios usando técnicas como el one-hot encoding. En este esquema, cada palabra se codifica con un vector en el que solo una posición toma el valor 1 y todas las demás son 0. Aunque este método era conceptualmente sencillo, presentaba dos grandes limitaciones: por un lado, generaba vectores extremadamente dispersos (sparse) y de alta dimensionalidad; por otro, no capturaba ninguna relación semántica entre palabras distintas.

La necesidad de superar estas limitaciones llevó al desarrollo de word embeddings densos y entrenables, como Word2Vec [14] y GloVe [15]. Estos modelos asignaban a cada palabra un vector continuo de dimensión fija, entrenado sobre grandes corpus textuales. Como resultado, palabras con significados similares terminaban ubicándose cerca unas de otras en el espacio vectorial. Un ejemplo de estas propiedades es la siguiente analogía vectorial:

$$\text{"rey"} - \text{"hombre"} + \text{"mujer"} \approx \text{"reina"}$$

lo que demuestra que estos modelos aprendían relaciones semánticas de tipo género, función o categoría a partir de la coocurrencia estadística en los textos.

Embeddings contextualizados: hacia modelos más inteligentes

A pesar de este avance, tanto Word2Vec como GloVe asignaban una única representación a cada palabra, independientemente del contexto. Esto resulta problemático en casos de ambigüedad léxica. Por ejemplo, palabras polisémicas como “banco” o “vela” tienen múltiples significados: “banco” puede referirse a una entidad financiera o a un asiento, mientras que “vela” puede ser una fuente de luz o parte de una embarcación. ¿Dónde ubicarlas entonces en el espacio vectorial si su significado depende de la frase?

Para resolver esta limitación, surgieron los embeddings contextualizados, introducidos por modelos como ELMo y popularizados por BERT, y posteriormente por modelos más avanzados como GPT-4, Claude 3 o LLaMA 3. Estos embeddings se generan dinámicamente en función del contexto completo de la oración en la que aparece la palabra.

Así, la palabra “banco” tendrá una representación vectorial distinta si aparece en “me senté en un banco del parque” o en “retiré dinero del banco”.

Este enfoque ha supuesto un gran avance en el procesamiento del lenguaje natural, ya que permite a los modelos captar matices semánticos, desambiguar significados y mejorar en tareas como comprensión lectora, generación de texto o traducción automática.

Visualización de embeddings

La Ilustración 2 muestra una representación tridimensional de embeddings generados para distintas palabras agrupadas semánticamente. Se observa cómo términos relacionados, como “dog”, “cat” y “wolf” o “apple” y “banana”, tienden a agruparse en regiones cercanas del espacio vectorial. Esto evidencia cómo los modelos de lenguaje capturan similitudes semánticas y estructuran el conocimiento en espacios continuos, donde la proximidad entre vectores refleja afinidad de significado.

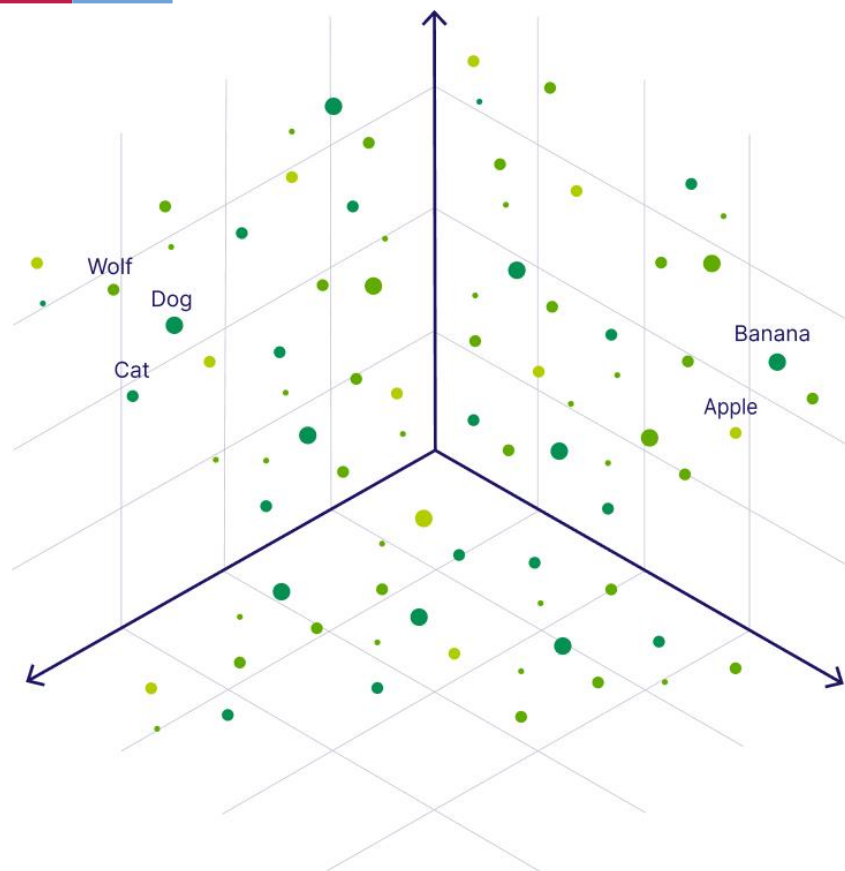


Ilustración 2. Representación tridimensional de embeddings de palabras agrupadas por proximidad [16]

3.2.3 LA CAPA DE ATENCIÓN

De RNN a la capa de atención

La introducción de la capa de atención supuso un avance disruptivo en el procesamiento del lenguaje natural. Hasta su desarrollo los modelos de procesamiento del lenguaje natural se basaban su funcionamiento principalmente en arquitecturas secuenciales como las redes neuronales recurrentes (RNN). Este tipo de redes procesaban el texto palabra por palabra en orden, manteniendo un estado oculto que se actualizaba a medida que avanzaba por la secuencia. Aunque esto permitía al modelo tener una “memoria” del contexto anterior, presentaba limitaciones importantes: dificultaba la captura de relaciones a largo plazo, la paralelización del entrenamiento y el manejo eficiente de dependencias complejas dentro del texto [17].

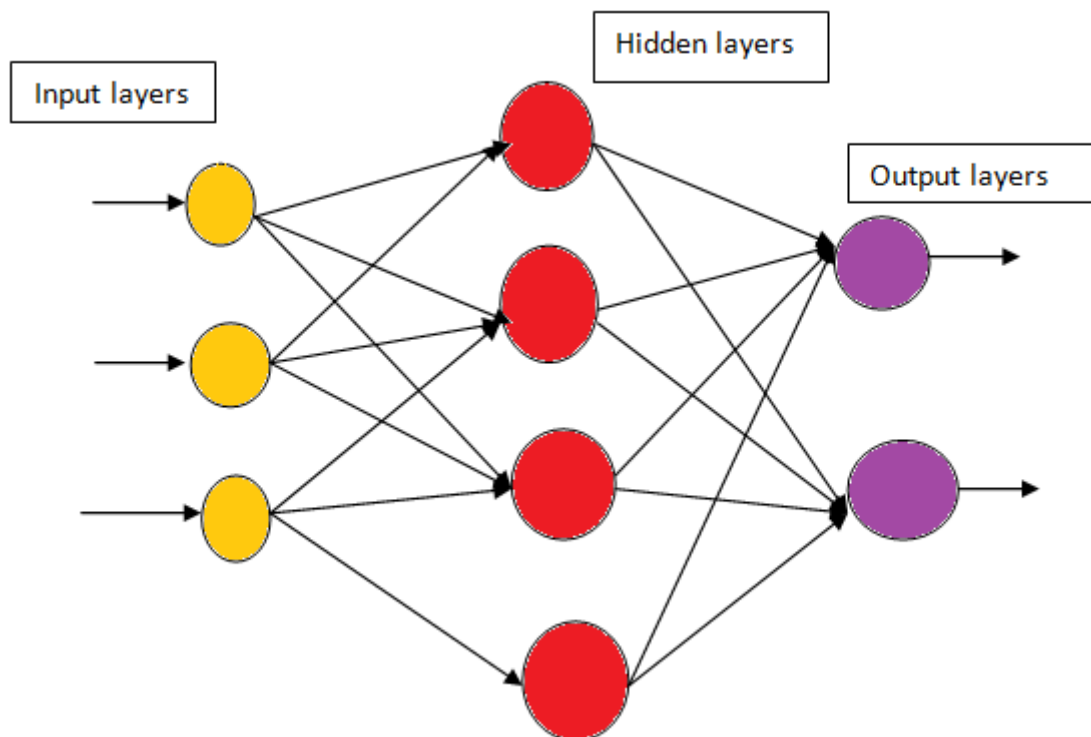


Ilustración 3. Esquema simplificado Red Neuronal Recurrente (RNN) [18]

En este enfoque secuencial, cada palabra dependía exclusivamente de la información acumulada en los pasos previos, como se observa en el esquema de la ilustración 3, lo que ocasionaba una pérdida progresiva del contexto conforme aumentaba la longitud de la secuencia. Además, era habitual que los gradientes se desvanecieran o explotaran, afectando negativamente al aprendizaje. Por tanto, existía una necesidad clara de mecanismos que pudieran superar estas barreras.

Surgimiento de la atención

El mecanismo de atención surge para superar las limitaciones mencionadas. En lugar de procesar las palabras de manera secuencial, la atención permite a los modelos identificar y centrar en las partes más relevantes de la secuencia, sin importar su posición u orden. Podemos resumirlo en una pregunta simple: ¿a qué debería prestar atención el modelo para interpretar correctamente esta palabra? En lugar de tratar todas las palabras por igual, el mecanismo de atención asigna diferentes pesos a cada una, en función de su importancia relativa para generar representación contextual de la palabra objetivo.

Por ejemplo, en la frase “El ingeniero resolvió el problema porque *sabía* cómo funcionaba el sistema”, el modelo debe aprender que *sabía* se refiere a *el ingeniero*, y no a *el problema*. La atención permite establecer esa relación, incluso si las palabras están separadas por varias posiciones.

El mecanismo de atención se implementa utilizando tres vectores derivados de los embeddings de entrada: queries (Q), keys (K) y values (V). Cada palabra se transforma en estos tres vectores mediante matrices entrenables. La atención se calcula como una combinación ponderada de los valores (V), donde los pesos se determinan midiendo la similitud entre la query y todas las keys mediante el producto escalar escalado y una función softmax, como se muestra en la Ilustración 4.

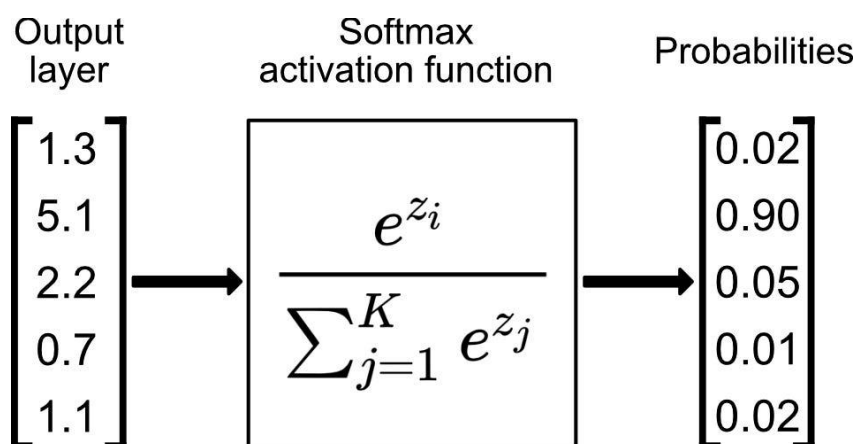


Ilustración 4. Ejemplo de funcionamiento de la función softmax [24]

Multi-head Attention y transformers

Una mejora introducida por el artículo “*Attention Is All You Need*” (Vaswani et al., 2017) es el mecanismo de **Multi-Head Attention**, que permite al modelo aprender múltiples relaciones en paralelo. Cada “cabeza” de atención opera de forma independiente, sobre proyecciones distintas de Q, K y V, y sus salidas se concatenan y combinan posteriormente.

Este mecanismo ha demostrado ser fundamental para capturar dependencias a largo plazo, ambigüedades semánticas y relaciones sintácticas, constituyendo la base de los modelos de lenguaje más avanzados desarrollados en los últimos años.

3.2.4 ARQUITECTURA TRANSFORMER

La arquitectura **Transformer** es un modelo de red neuronal diseñado para procesar secuencias de datos y que surge para cubrir las limitaciones de las redes neuronales recurrentes. La arquitectura fue presentada por Vaswani et al. en 2017 en el artículo *Attention Is All You Need*, y se basa íntegramente en el uso de La **capa de atención**

, explicada en el apartado anterior, eliminando la necesidad de procesar las palabras en orden secuencial.

A diferencia de las redes neuronales recurrentes (RNNs), que mantenían un estado interno que se actualizaba paso a paso, los Transformers permiten procesar toda la secuencia de entrada en paralelo. Esto no solo mejora la eficiencia computacional, sino que también facilita la captura de **relaciones a largo plazo** dentro del texto.

El Transformer aprende a asignar dinámicamente distintos pesos a cada palabra de la secuencia para generar representaciones contextualizadas, gracias a su mecanismo de **atención multicabeza**, explicado en el capítulo anterior. Además, introduce innovaciones como los **embeddings posicionales**, que le permiten tener en cuenta el orden de las palabras a pesar de no procesarlas de forma secuencial.

Estructura general: encoder-decoder

Un modelo Transformer completo se compone de **dos bloques principales**: el *encoder* y el *decoder*, ambos formados por múltiples capas idénticas apiladas. Esta estructura de doble bloque fue diseñada inicialmente para tareas de traducción automática, en las que se transforma una secuencia de entrada (por ejemplo, una frase en inglés) en una secuencia de salida (su traducción al español).

Tal como se muestra en la Ilustración 5, el encoder recibe la secuencia de entrada (tokenizada y embebida) y genera una representación interna con información contextual. A partir de esta representación, el decoder genera la salida paso a paso, aprendiendo a predecir el siguiente token basándose en los anteriores y en la información codificada por el encoder.

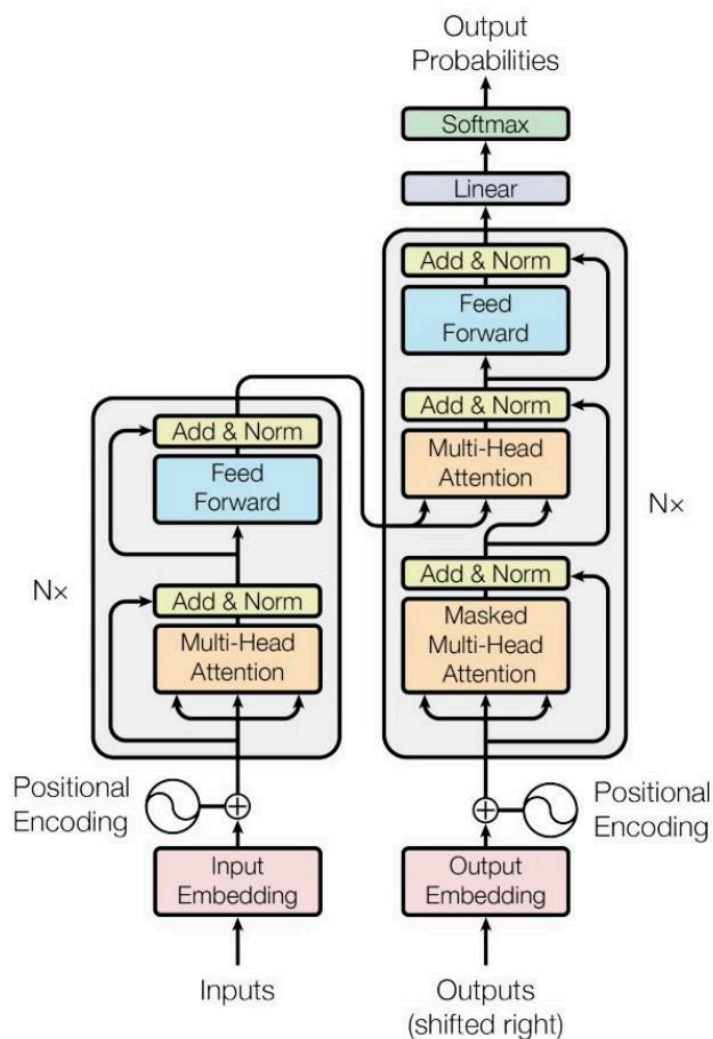


Ilustración 5. Arquitectura Transformer Encoder - Decoder [19]

Para entender esta arquitectura se trabaja a continuación un ejemplo clásico de la arquitectura Transformer en una tarea de traducción automática (el primer tipo de aplicación para que el que se diseñó): convertir una frase en francés (“*Je suis étudiant*”) a su equivalente en inglés (“*I am a student*”). El proceso como anticipábamos antes se lleva a cabo en dos fases:

- 1. Codificación (encoder):** cada palabra de la frase de entrada se convierte primero en un vector numérico a través de una capa de embedding, al que se le añade una codificación posicional que guarda el orden de las palabras. A continuación, esta secuencia de palabras embebidas pasa por una pila de encoder. Cada encoder aplica mecanismo de autoatención (self-attention) y capas de feed-forward para construir representaciones de cada palabra, considerando su contexto en la oración. El resultado es una matriz de vectores que encapsula la información semántica de la entrada completa.
- 2. Decodificación (decoder):** esta representación matricial se transfiere a la pila de decoder. El decoder genera la frase de salida palabra por palabra. Para realizar esto, cada bloque de la pila del decoder utiliza la información del encoder, las palabras que ya ha generado anteriormente y un mecanismo de enmascaramiento que impide al modelo acceder a palabras futuras durante el entrenamiento, simulando así un proceso de generación paso a paso.

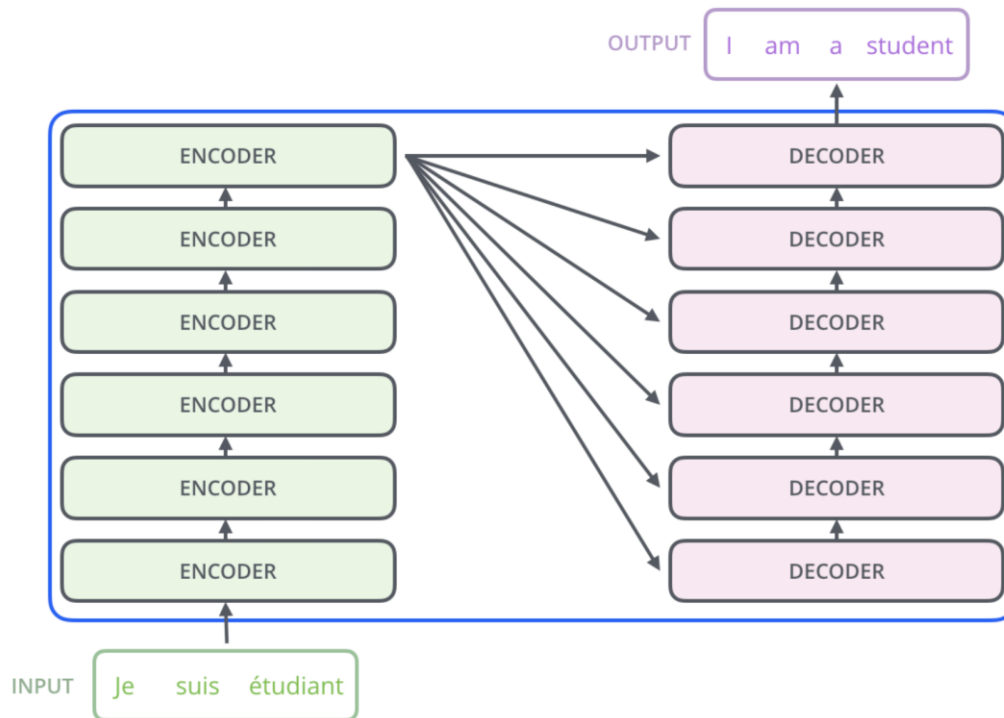


Ilustración 6. Representación esquemática de un Transformer encoder - decoder para tareas de traducción [20]

3.3 MODELOS DE LENGUAJE PREENTRENADOS (PLMs)

3.3.1 MODELOS PREENTRENADOS

En el contexto de la inteligencia artificial generativa y el aprendizaje automático, un modelo preentrenado es aquel que, como indica su nombre, ha sido entrenado previamente sobre una gran cantidad de datos, primero de forma genérica y posteriormente pudiendo ser adaptado para una tarea específica. Esta estrategia permite que el modelo adquiriera en primer lugar una comprensión generalista, pero que pueda reutilizarse y afinarse para resolver problemas concretos, ahorrando así esfuerzo computacional al no tener que entrenar un modelo desde cero para cada aplicación específica.

3.3.2 ¿QUÉ ES UN MODELO DE LENGUAJE?

Un modelo de lenguaje es un tipo de modelo estadístico o de aprendizaje automático diseñado para procesar y generar texto natural. Por texto natural nos referimos al lenguaje humano tal como se usa en la vida cotidiana. Su objetivo, o también podría decirse funcionamiento, es predecir la probabilidad de aparición de una palabra o secuencia de palabras dada una entrada anterior. Este proceso permite al modelo generar texto coherente, completar frases, responder preguntas o realizar tareas de traducción.

Para lograrlo, los modelos aprenden patrones y estructuras del lenguaje a partir de grandes volúmenes de texto, capturando tanto relaciones gramaticales como semánticas. Los modelos de lenguaje son la base para lo que ahora conocemos como inteligencia artificial conversacional, asistentes virtuales, traductores automáticos y muchas otras aplicaciones que requieren comprensión o generación de texto.

3.3.3 MODELO DE LENGUAJE PREENTRENADO

El modelo de lenguaje preentrenado es el resultado de la combinación de los dos conceptos explicados anteriormente: por un lado, el modelo preentrenado, entrenado inicialmente sobre grandes corpus de texto para adquirir una comprensión general del lenguaje; y por otro, el modelo de lenguaje, cuyo objetivo es aprender regularidades y estructuras del lenguaje natural.

En apartados anteriores se ha explicado cómo modelos como los Transformers introdujeron arquitecturas altamente eficientes basadas en mecanismos de atención, lo que ha permitido escalar el preentrenamiento a dimensiones antes impredecibles.

El resultado de esto son modelos versátiles y con altas capacidades que, tras haber sido expuestos a cantidades masivas de texto, son capaces de abordar múltiples tareas del procesamiento del lenguaje natural (traducción, resumen, respuesta a preguntas, redacción técnica, etc.) con poca o ninguna supervisión adicional.

3.3.4 ENTRENAMIENTO Y CAPACIDADES GENERALES

El entrenamiento de un modelo de lenguaje preentrenado (PLM) se divide habitualmente en dos fases: preentrenamiento y ajuste fino (*fine-tuning*). Durante la primera fase, el modelo se entrena con una enorme cantidad de texto no etiquetado, como libros, artículos o páginas web, utilizando tareas de predicción del lenguaje como la modelación causal (predecir la siguiente palabra dada una secuencia) o la modelación de lenguaje enmascarado (predecir palabras ocultas dentro de una oración). Estas tareas permiten al modelo aprender relaciones gramaticales, dependencias a largo plazo y patrones semánticos del lenguaje humano [21].

Posteriormente, el modelo puede ser ajustado para tareas específicas mediante entrenamiento adicional con conjuntos de datos más pequeños pero etiquetados, lo que permite especializarlo para tareas concretas como clasificación de texto, análisis de sentimiento, generación de respuestas o traducción automática [22].

Gracias a este proceso, los PLMs desarrollan una gran versatilidad. Una vez preentrenados, pueden aplicarse a una amplia variedad de tareas, dando lugar a aplicaciones prácticas en campos como la ingeniería, el derecho, la medicina o la educación. Modelos como GPT-4, Claude o LLaMA son capaces de redactar documentos técnicos, explicar conceptos complejos, generar código o resumir artículos científicos, demostrando una comprensión contextual avanzada del lenguaje.

La clave de esta capacidad generalista reside en la combinación de tres elementos: la escala del preentrenamiento, el volumen de parámetros del modelo y la arquitectura Transformer. Cuanto mayor es el volumen de texto expuesto, el número de parámetros y la calidad de los datos, mayor es la capacidad del modelo para generalizar, razonar y adaptarse a tareas diversas sin necesidad de ser reentrenado desde cero .

3.3.5 CLASIFICACIÓN DE PLMs

Los modelos de lenguaje preentrenados (PLMs) pueden clasificarse desde distintos enfoques, en función de las características de entrada y salida que manejan, así como de la

arquitectura sobre la que están contruidos. Esta clasificación permite entender mejor su funcionamiento, sus capacidades y los casos de uso más adecuados para cada tipo.

Desde el punto de vista del **tipo de entrada (input)**, los modelos pueden dividirse en varias categorías. Los modelos textuales son los más tradicionales y comunes en el procesamiento del lenguaje natural (PLN), ya que reciben texto como entrada. Ejemplos destacados incluyen GPT-4o, Claude 3 o LLaMA 3. Por otro lado, los modelos visuales están diseñados para procesar imágenes y generar texto relacionado, como descripciones o etiquetas, siendo CLIP o BLIP-2 ejemplos paradigmáticos. También existen modelos auditivos, como Whisper o VALL-E, que convierten audio en texto o incluso generan voz sintética. Finalmente, los modelos multimodales combinan diferentes tipos de entrada, como texto, imágenes o audio, para realizar tareas complejas que requieren comprensión integrada. Gemini 1.5 o GPT-4o son ejemplos recientes de este tipo de modelos que permiten interpretar y razonar sobre varios formatos simultáneamente.

En cuanto al **tipo de salida (output)**, la clasificación también varía según la tarea a la que se destina el modelo. Muchos modelos generan texto continuo de forma autoregresiva, útil para redacción automática, diálogos o generación creativa. Otros están diseñados para tareas de clasificación, donde la salida es una etiqueta o categoría, como ocurre con BERT ajustado para análisis de sentimiento. En tareas de etiquetado secuencial, como el reconocimiento de entidades nombradas (NER), el modelo genera una etiqueta por cada token de entrada. También existen modelos cuya salida son representaciones vectoriales (embeddings), que permiten calcular similitudes semánticas entre textos o utilizarlos como input para otros modelos. Además, hay modelos orientados a generar estructuras complejas, como fragmentos de código, respuestas en JSON o datos organizados según una lógica específica.

Por último, desde el punto de vista **arquitectónico**, los modelos de lenguaje pueden dividirse en tres grandes grupos. Los modelos autoregresivos, también llamados *decoder-only*, generan texto de forma secuencial, prediciendo cada token a partir del contexto anterior. Este diseño es el más eficaz para tareas generativas y se encuentra en modelos como GPT-4o, Claude 3 o Mistral. Por otro lado, los modelos basados únicamente en el encoder están especializados en tareas de comprensión y clasificación, como BERT o RoBERTa, y no son

capaces de generar texto por sí mismos. En tercer lugar, los modelos encoder-decoder, o *sequence-to-sequence*, están diseñados para transformar secuencias de entrada en otras de salida, lo que los hace especialmente adecuados para traducción automática, resumen de textos o preguntas y respuestas. Ejemplos de esta arquitectura son T5 o BART. En la actualidad, también han surgido modelos más sofisticados, como Gemini o Gato de DeepMind, que combinan distintas arquitecturas, capacidades multimodales o incluso razonamiento iterativo, configurando una nueva generación de modelos más adaptables y complejos.

Modelo	Tipo de Input	Tipo de Output	Tipo de Arquitectura
GPT-2	Texto	Texto continuo	Decoder-only
GPT-3.5 / GPT-4	Texto	Texto continuo	Decoder-only
GPT-4o	Multimodal	Texto continuo	Decoder-only
Claude 3 (Opus)	Texto	Texto continuo	Decoder-only
Claude 3.5 Sonnet	Texto	Texto continuo	Decoder-only
Gemini 1.5 Pro	Multimodal	Texto continuo	Encoder-decoder+
Gemini Flash	Multimodal	Texto continuo	Encoder-decoder+
O3-mini	Texto	Texto breve/ estructurado	Decoder-only
T5	Texto	Texto continuo	Encoder-decoder
BERT	Texto	Texto (seq-to-seq)	Encoder-only
RoBERTa	Texto	Etiquetas / embeddings	Encoder-only
LLaMA 3	Texto	Etiquetas / embeddings	Decoder-only

ICAI	ICADE	CIHS	
Mistral	Texto	Texto continuo	Decoder-only
Gato (Deepmind)	Multimodal	Texto continuo	Mixto (multi-arch)

Tabla 1. Clasificación de modelos de lenguaje según tipo de input, output y arquitectura

3.3.6 PLMS CON LOS QUE SE EXPERIMENTA

Aprovechando que en esta parte del documento estamos clasificando PLMs, resulta muy interesante hacer un repaso sobre los PLMs con los que más adelante se experimentará. Esto ayudará a entender algunos de los resultados obtenidos y también a tener un punto de comparación entre qué outputs son coherentes y cuáles no. Por ejemplo, GPT-4o es de los modelos más recientes e innovadores y, por lo tanto, se esperan resultados mejores frente a otros modelos con menos capacidades en varias aplicaciones.

Modelo	Organización	Lanzamiento	Parámetros estimados	Aplicaciones comunes
GPT-2	OpenAI	2019	1.5B	Generación de texto, traducción, clasificación básica
GPT-3.5	OpenAI	2022	~6.7B	Chatbots, generación de texto, asistencia educativa
GPT-4	OpenAI	2023	~1T (estimado, no público)	Redacción técnica, resumen, asistencia legal, tareas complejas
GPT-4o	OpenAI	2024	Similar a GPT-4	Asistentes conversacionales, interpretación de entradas complejas

ICAI	ICADE	CIHS		
O3-mini	OpenAI	2024	No especificado	Bots embebidos, generación rápida, inferencias simples
Claude 3.5 Sonnet	Anthropic	2024	No público	Asistente de programación, redacción creativa, análisis de documentos
Claude 3 Haiku	Anthropic	2024	No público	Tareas rápidas, asistentes móviles, inferencia en tiempo real
Gemini 1.5 Flash	Google DeepMind	2024	No público	Búsqueda rápida, clasificación, chat en apps móviles
Gemini 1.5 Pro	Google DeepMind	2024	No público	Codificación, investigación, análisis avanzados

Tabla 2. Comparativa de Modelos de Lenguaje utilizados en la parte experimental

Como se comentaba anteriormente, la tabla I muestra una comparativa de algunos de los modelos de lenguaje preentrenados utilizados en este trabajo, incluyendo información sobre su año de lanzamiento, número aproximado de parámetros y aplicaciones más comunes. Esta visión global permite contextualizar el rendimiento observado en los experimentos posteriores y resaltar las diferencias entre modelos de distinta generación y propósito.

3.4 AUGMENTING AI: INTELIGENCIA AUMENTADA

En los últimos años, ha cobrado fuerza una nueva forma de entender el papel de la inteligencia artificial en la interacción con las personas, especialmente en ámbitos

profesionales y técnicos. Esta visión, conocida como inteligencia aumentada (*Augmenting AI*), se basa en la idea de que los modelos de lenguaje no deben concebirse como sistemas autónomos que sustituyen al ser humano, sino como herramientas complementarias diseñadas para mejorar y ampliar sus capacidades cognitivas.

Este enfoque plantea que el verdadero potencial de la IA se alcanza cuando se combina con la experiencia, el juicio crítico y la creatividad humanas. En lugar de delegar por completo una tarea técnica o intelectual en el modelo, el usuario se mantiene en el centro del proceso, interactuando con la IA para generar ideas, evaluar opciones, detectar errores o reformular soluciones. De esta manera, se establece una dinámica colaborativa en la que la máquina actúa como asistente inteligente, capaz de proponer contenidos útiles, pero siempre bajo la supervisión y validación del usuario.

En el contexto del procesamiento del lenguaje natural, esta estrategia se concreta en flujos de trabajo iterativos en los que el modelo genera un primer resultado (por ejemplo, un informe técnico o una solución a un problema), que luego es revisado, afinado o incluso reenviado al modelo para ser mejorado. Algunas de las versiones más avanzadas de modelos generativos, como GPT-4o o Claude 3 Opus, integran mecanismos internos de autorreflexión o verificación, lo que les permite analizar sus propias respuestas y producir una versión final más coherente o ajustada al criterio solicitado. Esta capacidad de autoajuste progresivo es especialmente valiosa en tareas complejas o delicadas.

El concepto de *Augmenting AI* es especialmente relevante para la ingeniería, ya que permite acelerar tareas sin comprometer la precisión o el control, y abre la puerta a nuevos modelos de trabajo asistido por IA. En el presente proyecto, se exploran precisamente este tipo de interacciones, evaluando hasta qué punto los modelos de lenguaje pueden actuar como aliados eficaces en la redacción de documentación técnica, la interpretación de problemas o la propuesta de soluciones viables.

Capítulo 4. DEFINICIÓN DEL TRABAJO EXPERIMENTAL

4.1 OBJETIVOS DEL DESARROLLO EXPERIMENTAL

OBJETIVOS GENERALES

El objetivo general de la parte práctica consiste en evaluar el rendimiento de distintos modelos de lenguaje preentrenados en diversas tareas del ámbito de la ingeniería, mediante una serie de experimentos diseñados específicamente para este fin.

OBJETIVOS ESPECÍFICOS

Además, se busca cumplir los siguientes objetivos específicos:

1. Diseñar una serie de experimentos que cubran distintas áreas de interés y aplicación de los modelos de lenguaje en tareas técnicas de ingeniería.
2. Comparar las respuestas generadas por los diferentes modelos y documentar cualquier anomalía o comportamiento inesperado.
3. Analizar el impacto de la variación de parámetros de generación en la calidad de las respuestas.
4. Documentar y organizar los resultados obtenidos de forma que sirvan como base de análisis para estudios posteriores.
5. Identificar, en función de los resultados, oportunidades reales de uso para cada modelo en distintos contextos técnicos, así como posibles líneas futuras de investigación.

4.2 DESCRIPCIÓN DE LOS EXPERIMENTOS REALIZADOS

DISEÑO Y TAREAS ABORDADAS

En los próximos capítulos se desarrolla la parte experimental del proyecto, fundamentada en los conceptos teóricos expuestos en capítulos anteriores. Esta sección consiste en la evaluación de distintos modelos de lenguaje preentrenados, aplicados a diferentes tareas representativas del ámbito de la ingeniería.

Para ello, se han llevado a cabo una serie de experimentos diseñados para cubrir una serie de tareas, desde la redacción técnica y la resolución de problemas numéricos, hasta la evaluación de procesos simulados. Cada experimento se centra en una tipología concreta y se han probado de manera progresiva, aumentando su complejidad a medida que se obtenían resultados coherentes en cada uno de los experimentos anteriores.

En total, se han desarrollado cinco experimentos. El primero consiste en la explicación técnica de un concepto básico de ingeniería utilizando lenguaje académico. El segundo plantea un problema numérico sencillo que debe resolverse paso a paso. El tercero requiere la redacción de un breve informe técnico sobre el efecto fotoeléctrico, basándose en una serie de resultados proporcionados como parte del enunciado. El cuarto presenta un ejercicio de magnetismo extraído de un examen de ICAI. El quinto implica una simulación con la librería SimPy, en la que se describe un proceso dinámico y se solicita su análisis o interpretación por parte del modelo, especialmente en presencia de un fallo introducido intencionadamente en el sistema.

Todos estos experimentos se han llevado a cabo utilizando modelos de lenguaje de última generación ofrecidos por distintas compañías, concretamente modelos de la familia GPT (GPT3.5, GPT4, GPT4oy o3-mini), Claude (Claude 3 Opus) y Gemini (Gemini 2.5 Flash y Gemini 2.5 Pro).

La evaluación se ha realizado comparando las respuestas generadas por cada modelo con una solución de referencia (oficial o elaborada por una fuente experta), analizando su similitud utilizando el teorema del coseno.

Además, se han introducido variaciones en los parámetros de generación, como el valor de *temperature* o el tipo de instrucciones incluidas en el *prompt*, con el objetivo de observar cómo afectan al rendimiento y a la formulación de las respuestas.

El parámetro *temperature* controla el nivel de aleatoriedad en las respuestas generadas por el modelo: valores bajos (por ejemplo, 0.2) generan respuestas más deterministas y conservadoras, mientras que valores altos (por ejemplo, 0.9) permiten más creatividad y variabilidad. Por otro lado, el *prompt* es el texto de entrada que se proporciona al modelo para indicarle qué tarea debe realizar o qué tipo de respuesta se espera. Su redacción tiene un impacto directo en la calidad y relevancia de la respuesta generada, por lo que ha sido cuidadosamente ajustado en cada experimento.

Además del parámetro *temperature*, también se han considerado otros parámetros de generación como *top_k*, *top_p* y *max_tokens*. El parámetro *top_k* restringe las opciones del modelo a las k palabras con mayor probabilidad, mientras que *top_p* (también conocido como *nucleus sampling*) selecciona las palabras más probables cuya suma acumulada de probabilidades no supere p . Por su parte, *max_tokens* establece un límite máximo en la longitud de la respuesta generada. Aunque se realizaron pruebas preliminares con distintas combinaciones de estos parámetros, no se obtuvieron variaciones suficientemente significativas que justificaran su análisis detallado. Dado que su impacto no era sustancial frente al incremento notable del tiempo de generación y la complejidad de interpretación, se optó por centrar la evaluación principal únicamente en el parámetro *temperature*, manteniendo valores razonables por defecto para los demás.

A continuación, se detallan las herramientas utilizadas para llevar a cabo los experimentos, así como una descripción individual de cada uno de ellos y los resultados obtenidos.

METODOLOGÍA COMÚN APLICADA

Antes de entrar en detalle individualmente con cada uno de los experimentos, es importante explicar la metodología común que se ha seguido para llevarlos a cabo. El objetivo de

mantener una metodología común es establecer una estructura unificada y garantizar que los criterios de evaluación sean lo más homogéneos posible.

El proceso comienza con la definición del propósito del experimento: qué se pretende conseguir con él y qué tarea o habilidad relacionada con la ingeniería se desea evaluar. Una vez definido este objetivo, se elabora una respuesta de referencia que se considera óptima, ya sea porque está redactada por un experto o porque proviene de una fuente reconocida. Por ejemplo, en el caso de un problema extraído de un examen, esta respuesta será la oficial proporcionada por ICAI. Esta respuesta sirve de base para comparar los resultados obtenidos, especialmente en la parte cuantitativa.

El segundo paso consiste en la redacción del *prompt*, es decir, la instrucción o enunciado que se le pasa al modelo. Este proceso se realiza cuidadosamente y tras varias pruebas, ya que el *prompt* actúa como guía directa para el modelo, y pequeños cambios en su redacción pueden provocar alteraciones significativas en las respuestas generadas. Finalmente, se selecciona el *prompt* que ofrece mejores resultados y se emplea para la ejecución del experimento.

Cada uno de los modelos especificados en el código genera su respuesta, y esta se almacena en un archivo de texto individual. En caso de obtenerse respuestas anómalas (por ejemplo, respuestas vacías o con errores graves o incoherencias conocidas como *alucinaciones*), se documenta la incidencia, se investiga la causa y se analiza su impacto antes de continuar con la evaluación del resto de resultados.

Para las respuestas válidas, se lleva a cabo una evaluación cuantitativa basada en la similitud semántica entre la respuesta generada por el modelo y la respuesta de referencia. Para ello, se emplea el **teorema del coseno**, una técnica común en el procesamiento del lenguaje natural que permite comparar textos mediante el cálculo del ángulo entre sus vectores de representación (*embeddings*). Cuanto más cercano a 1 sea el valor del coseno, mayor será la similitud semántica entre ambos textos, independientemente de la redacción exacta.

En este proyecto se han utilizado dos métodos distintos para calcular dicha similitud. El primero consiste en usar un **modelo de embeddings contextualizado** de OpenAI,

concretamente el modelo all-MiniLM-L6-v2, que genera representaciones vectoriales teniendo en cuenta el significado global del texto. Este enfoque es especialmente útil para capturar el sentido general de una respuesta y comparar ideas de forma más precisa, aunque estén redactadas de forma distinta.

El segundo método se basa en un enfoque más clásico mediante **TF-IDF (Term Frequency–Inverse Document Frequency)**, que transforma cada texto en un vector según la frecuencia y relevancia relativa de sus palabras. Aunque este modelo no tiene en cuenta el contexto ni el orden de los términos, permite detectar coincidencias exactas de contenido léxico y resulta útil para identificar diferencias más literales entre respuestas.

Ambos métodos aportan perspectivas complementarias: mientras que los embeddings permiten evaluar la proximidad semántica de los textos en un espacio vectorial más profundo, el modelo TF-IDF ofrece una visión basada en las coincidencias de términos específicos. La combinación de ambas aproximaciones permite realizar una comparación más robusta y matizada.

Esta estructura metodológica permite evaluar no solo si los modelos son capaces de resolver una tarea, sino también cómo la resuelven. Esto aporta valor añadido, ya que permite detectar anomalías, identificar puntos fuertes o débiles en cada tipo de tarea y descubrir oportunidades de mejora o aplicaciones más adecuadas para cada modelo. Por ejemplo, si un modelo destaca especialmente en tareas de redacción, pueden plantearse aplicaciones educativas o de documentación técnica; si otro ofrece mejores resultados en resolución matemática, será más adecuado para tareas numéricas o de cálculo.

Capítulo 5. DESARROLLO EXPERIMENTAL

Durante este capítulo, como se anticipaba en el anterior, se lleva a cabo toda la documentación de la parte experimental. Se recoge el desarrollo y los resultados de los cinco experimentos realizados, siguiendo una estructura uniforme a lo largo de todos ellos. Se muestran ejemplos de resultados de los modelos. Aunque se extrae alguna conclusión de los resultados, se profundiza en las comparaciones y en el análisis de implicaciones de dichos resultados en el capítulo 6, **Análisis de resultados**.

5.1 EXPERIMENTO 01: EXPLICACIÓN DE CONCEPTO TÉCNICO

DESCRIPCIÓN Y OBJETIVOS DEL EXPERIMENTO

El primer experimento tiene como objetivo evaluar la capacidad de los modelos para explicar de manera clara y técnicamente precisa, pero entendible un concepto. En este caso, se ha seleccionado un concepto relacionado con la ingeniería como ejemplo para su realización, aunque este experimento es escalable o trasladable a cualquier otro concepto. El concepto elegido es la corriente alterna (AC). El objetivo específico de este experimento es evaluar la capacidad de los modelos para explicarlo. Buscamos identificar cuál es el modelo, o cuáles son los modelos, con mejor desempeño en un ámbito explicativo y educacional, sugiriendo que los PLMs pueden utilizarse como herramienta para la docencia, permitiendo que las explicaciones teóricas puedan realizarse de forma más autónoma.

DISEÑO DEL PROMPT

El prompt actúa como una primera guía para el modelo, y pequeños cambios en este pueden suponer alteraciones significativas en los resultados. Se han realizado diferentes pruebas ajustando el prompt y se ha mantenido aquel que ofrecía mejores resultados en la evaluación. Para ello, se seleccionó uno de los modelos evaluados, concretamente GPT-4, por tratarse de un modelo intermedio dentro del conjunto de modelos analizados, tanto por su fecha de lanzamiento como por su rendimiento esperado. Se probaron distintas variantes del prompt

y se evaluaron los resultados obtenidos mediante el teorema del coseno, comparando las respuestas generadas con una respuesta de referencia. Esto permitió identificar qué formulación conducía a una mayor similitud semántica y, por tanto, a una mejor alineación con la respuesta considerada óptima. Se realiza este proceso únicamente con un modelo para poder estudiar de manera aislada el efecto del prompt, sin que las diferencias entre modelos interfieran en el análisis.

Finalmente, el prompt utilizado para este experimento ha sido el siguiente: "Explica qué es una corriente alterna (AC) en ingeniería eléctrica, con un lenguaje claro, técnico y adecuado para estudiantes universitarios."

Al final, lo más efectivo ha resultado ser un prompt sencillo, directo, que solicita el concepto a explicar y especifica que se trata de una definición en un contexto de ingeniería eléctrica. Además, se indica que el público objetivo son estudiantes universitarios, lo que permite al modelo adecuar su nivel de expresión y profundidad técnica.

DISEÑO Y FUENTE DE LA RESPUESTA OFICIAL PARA COMPARACIÓN

Para realizar una comparación fiable, lo más importante es contar con una buena referencia con la que evaluar la calidad de las respuestas generadas. Si la respuesta oficial utilizada como base de comparación no es adecuada, no dispondremos de un punto de referencia válido que permita obtener resultados significativos. Por esta razón, todas las respuestas modelo utilizadas en este trabajo provienen de, o han sido elaboradas a partir de, fuentes oficiales y reconocidas.

No se ha seguido exactamente la misma metodología en todos los experimentos, ya que la naturaleza de cada uno conlleva necesidades diferentes en cuanto a la selección de la respuesta de referencia. En este caso concreto, al tratarse de la explicación de un concepto teórico, nos encontramos ante una evaluación más subjetiva que en otros experimentos (por ejemplo, en un problema numérico, donde existe una única solución correcta y verificable).

Por ello, se ha optado por utilizar dos respuestas ejemplo para reforzar la fiabilidad del análisis y permitir una comparación más robusta. Ambas respuestas provienen de manuales o libros de texto reconocidos en el ámbito de la ingeniería eléctrica.

Respuesta ejemplo 1

La corriente alterna (AC) es una forma de corriente eléctrica en la cual la magnitud y la dirección de la corriente varían sinusoidalmente con el tiempo. Su forma general puede expresarse como $i(t) = I_0 \sin(\omega t + \phi)$, donde I_0 es la amplitud, ω es la frecuencia angular y ϕ es la fase inicial. Esta variación periódica permite que la energía eléctrica sea transmitida de manera más eficiente en sistemas de potencia.

Una de las principales ventajas de la corriente alterna frente a la corriente continua es la facilidad con la que puede transformarse entre diferentes niveles de voltaje utilizando transformadores, lo que resulta fundamental para la transmisión de electricidad a larga distancia con menores pérdidas.

La mayoría de los sistemas eléctricos comerciales utilizan corriente alterna con una frecuencia de 50 o 60 Hz. En el análisis de circuitos de corriente alterna, se utilizan representaciones fasoriales y conceptos como impedancia, potencia compleja y factor de potencia para describir el comportamiento de los sistemas [23].

Respuesta ejemplo 2

La corriente alterna (AC) es una forma de energía eléctrica caracterizada por una inversión periódica del sentido de la corriente. A diferencia de la corriente continua, en la cual la dirección del flujo de carga es constante, la AC alterna su polaridad a intervalos regulares, generalmente siguiendo una forma de onda sinusoidal.

La frecuencia de la corriente alterna, que determina cuántas veces por segundo cambia de dirección, se mide en hertzios (Hz). Las frecuencias estándar en los sistemas de suministro eléctrico son de 50 Hz o 60 Hz, dependiendo de la región geográfica.

La corriente alterna es utilizada predominantemente en los sistemas de distribución de energía eléctrica debido a su capacidad para cambiar de nivel de tensión mediante

transformadores, lo que permite una transmisión eficiente a larga distancia con menores pérdidas energéticas [24] [25].

Estas respuestas sirven como modelo de referencia para calcular métricas de similitud, como la similitud del coseno.

DESARROLLO EXPERIMENTAL

Generación de respuestas

A continuación se muestra el código utilizado para generara las respuestas de los modelos de GPT, Claude y Gemini [26] [27] [28]. En esta primera prueba se dejan los parámetros predeterminados de los modelos especificados en las API. A continuación se describe el funcionamiento general del código y las líneas más destacadas.

GPT 2

En este primer experimento, además de evaluar modelos actuales como GPT-4o o Claude, se ha incluido GPT-2 con el objetivo de proporcionar un punto de comparación representativo de los modelos generativos más antiguos. GPT-2, lanzado por OpenAI en 2019, marcó un hito en el desarrollo de modelos de lenguaje, pero hoy queda notablemente superado en múltiples aspectos: calidad, contexto, razonamiento, velocidad y alineación con instrucciones.

Su inclusión en esta prueba sirve para ilustrar la evolución que ha experimentado esta tecnología en apenas cinco años, permitiendo visualizar con claridad hasta qué punto han mejorado las capacidades de los modelos.

```
from transformers import GPT2LMHeadModel, GPT2Tokenizer
import torch

# Carga el modelo y el tokenizador GPT-2
model_name = "gpt2"
tokenizer = GPT2Tokenizer.from_pretrained(model_name)
model = GPT2LMHeadModel.from_pretrained(model_name)
```

```
ICAI ICAE CIHS
# Coloca el modelo en modo evaluación
model.eval()

# Prompt del experimento (corriente alterna)
prompt = "Explica qué es una corriente alterna (AC) en
ingeniería eléctrica, con un lenguaje claro, técnico y
adecuado para estudiantes universitarios."

# Tokeniza el prompt
inputs = tokenizer(prompt, return_tensors="pt")

# Genera texto
with torch.no_grad():
    outputs = model.generate(
        **inputs,
        max_new_tokens=200,
        do_sample=True,
        temperature=0.7,
        top_p=0.9,
        top_k=50
    )

# Decodifica y muestra el resultado
respuesta = tokenizer.decode(outputs[0],
skip_special_tokens=True)

print("\n--- Respuesta de GPT-2 ---\n")
print(respuesta)
```

A diferencia del resto de modelos evaluados, GPT-2 puede ejecutarse localmente sin necesidad de acceder a la API de OpenAI, lo que facilita su integración mediante la librería transformers de Hugging Face. El código consiste en cargar el modelo y su tokenizador, definir el prompt del experimento (explicación de la corriente alterna), generar una respuesta de hasta 300 tokens utilizando parámetros de muestreo como temperature, top_k y top_p, y finalmente guardar el resultado en un archivo. Esta configuración permite obtener una respuesta representativa del rendimiento de los modelos generativos anteriores a la llegada

de los sistemas más avanzados actuales, sin requerir conexión externa ni clave de acceso a una API.

GPT

```
from openai import OpenAI
import time
import os

# Inicializa el cliente de OpenAI con API Key
client = OpenAI(api_key="sk-...") # Se mantiene privada
por motivos de seguridad

# Prompt del experimento
prompt = "Explica qué es una corriente alterna (AC) en
ingeniería eléctrica, con un lenguaje claro, técnico y
adecuado para estudiantes universitarios."

# Lista de modelos a probar
models = ["gpt-3.5-turbo", "gpt-4", "gpt-4o", "o3-mini"]

# Crea carpeta de salida si no existe
output_folder = "respuestas2"
os.makedirs(output_folder, exist_ok=True)

# Ejecuta cada modelo
for model in models:
    print(f"\n--- Modelo GPT: {model} ---")
    try:
        start = time.time()

        # Prompt directo si es un modelo 'mini'
        if "mini" in model:
            response = client.chat.completions.create(
                model=model,
                messages=[{"role": "user", "content":
prompt}],
                max_completion_tokens=500
            )
        else:
```

```
ICAI ICADE CIHS
# Prompt con rol de sistema si es modelo GPT
estándar

response = client.chat.completions.create(
    model=model,
    messages=[
        {"role": "system", "content": "Actúa
como un ingeniero docente, explicando conceptos técnicos
con claridad."},
        {"role": "user", "content": prompt}
    ],
    temperature=0.3,
    max_tokens=500
)

# Guarda el contenido de la respuesta
content =
response.choices[0].message.content.strip()
duration = round(time.time() - start, 2)
filename =
f"{output_folder}/{model}_experimento_04.txt"

# Escribe la respuesta en un archivo de texto
with open(filename, "w", encoding="utf-8") as f:
    f.write(content)

print(f"✓ Guardado en {os.path.basename(filename)}
| {duration} segundos")

except Exception as e:
    print(f"X Error con {model}: {e}")
```

El script de python mostrado utiliza la API de Open AI para tener acceso a los modelos y así poder generar respuestas. El funcionamiento general es el siguiente: En primer lugar, se llama a cada modelo con el prompt. En segundo lugar, se guardan todas las respuestas en un archivo común, para tener un vista general, y en archivos separados que son los que se utilizarán en la comparación. Finalmente se imprime un mensaje de que se ha guardado el

archivo y los tiempos en elaborar la respuesta de cada modelo, ya que es un parámetro que también se va a valorar en la evaluación.

Claude

```
import requests
import time
import os

# API key de Anthropic (Claude)
api_key = "sk-ant-..." # sustituir por clave, se omite por
motivos de seguridad

# Prompt del experimento
prompt = "Explica qué es una corriente alterna (AC) en
ingeniería eléctrica, con un lenguaje claro, técnico y
adecuado para estudiantes universitarios."

# Modelos Claude a probar
modelos = ["claude-3-5-sonnet-20241022", "claude-3-haiku-
20240307"]

# Crea carpeta de salida si no existe
output_folder = "respuestas2"
os.makedirs(output_folder, exist_ok=True)

# Ejecuta cada modelo
for modelo in modelos:
    print(f"\n--- Modelo Claude: {modelo} ---")
    start = time.time()

    # Realiza la solicitud a la API de Anthropic
    response = requests.post(
        url="https://api.anthropic.com/v1/messages",
        headers={
            "x-api-key": api_key,
            "anthropic-version": "2023-06-01",
            "content-type": "application/json"
        },
        json={
```

```
        "model": modelo,
        "max_tokens": 300,
        "messages": [
            {"role": "user", "content": prompt}
        ]
    }
)

# Procesa la respuesta
if response.status_code == 200:
    data = response.json()
    # Extrae el contenido generado por el modelo
    content = data["content"][0]["text"].strip()
    duration = round(time.time() - start, 2)

    # Guarda la respuesta en archivo
    filename =
f"{output_folder}/{modelo}_experimento_04.txt"
    with open(filename, "w", encoding="utf-8") as f:
        f.write(content)

    print(f"✓ Guardado en {os.path.basename(filename)}
| {duration} segundos")
else:
    print(f"X Error con {modelo}:
{response.status_code} - {response.text}")
    duration =
round(time.time() - start, 2)
    if response.status_code == 200:
        data = response.json()
        message = data["content"][0]["text"] if "content"
in data and len(data["content"]) > 0 else "[Respuesta
vacía]"
        filename =
f"{output_folder}/{modelo}_experimento_04.txt"
        with open(filename, "w", encoding="utf-8") as f:
            f.write(message)

        print(f"✓ Guardado en {os.path.basename(filename)}
| {duration} segundos")
    else:
```

```
print(f" Error {response.status_code}:  
{response.text}")
```

El script en Python mostrado emplea la API de **Anthropic** para acceder a los modelos **Claude 3.5 Sonnet** y **Claude 3 Haiku**, con el fin de generar respuestas al mismo prompt utilizado en los demás experimentos. Su funcionamiento general es el siguiente: En primer lugar, se lanza una petición POST a la API de Anthropic para cada uno de los modelos, especificando el prompt, el número máximo de tokens a generar y los encabezados necesarios para la autenticación.

En segundo lugar, si la respuesta es válida, el contenido generado por el modelo se guarda en un archivo .txt individual dentro de una carpeta común de salida. Finalmente, se imprime un mensaje informando del éxito de la operación y del tiempo que ha tardado en generarse la respuesta, ya que este es un parámetro adicional que también se tendrá en cuenta durante la evaluación comparativa.

Gemini

```
import requests  
import time  
import os  
import json  
  
# Establece la API key de Google Gemini  
api_key = "AIza..."  
  
# Define el prompt del experimento  
prompt = "Explica qué es una corriente alterna (AC) en  
ingeniería eléctrica, con un lenguaje claro, técnico y  
adecuado para estudiantes universitarios."  
  
# Define los modelos Gemini a probar  
modelos = ["gemini-2.5-pro-latest", "gemini-2.5-flash-  
latest"]
```

```
ICAI ICADE CIHS
# Crea la carpeta de salida si no existe
output_folder = "respuestas2"
os.makedirs(output_folder, exist_ok=True)

# Define los headers comunes para la API
headers = {
    "Content-Type": "application/json",
    "x-goog-api-key": api_key
}

# Define el endpoint base de la API de Gemini
endpoint_base =
"https://generativelanguage.googleapis.com/v1beta/models"

# Ejecuta cada modelo
for modelo in modelos:
    print(f"\n--- Modelo Gemini: {modelo} ---")
    start = time.time()

    # Construye la URL completa para la generación de
    contenido
    url = f"{endpoint_base}/{modelo}:generateContent"

    # Construye el payload con el prompt
    payload = {
        "contents": [
            {
                "parts": [
                    {
                        "text": prompt
                    }
                ]
            }
        ]
    }

    # Realiza la solicitud a la API de Gemini
    response = requests.post(url, headers=headers,
data=json.dumps(payload))
    duration = round(time.time() - start, 2)
```

```
ICAI ICAE CIHS

# Procesa la respuesta
if response.status_code == 200:
    data = response.json()

    # Extrae el texto generado por el modelo
    content = data.get("candidates",
[{}])[0].get("content", {}).get("parts",
[{}])[0].get("text", "[Respuesta vacía]")

    # Guarda la respuesta en un archivo de texto
    filename =
f"{output_folder}/{modelo}_experimento_04.txt"
    with open(filename, "w", encoding="utf-8") as f:
        f.write(content)

    # Imprime confirmación de guardado
    print(f"✓ Guardado en {os.path.basename(filename)}
| {duration} segundos")
else:
    # Imprime mensaje de error si la API falla
    print(f"X Error {response.status_code}:
{response.text}")
```

Este script en Python utiliza la API de Google Gemini para acceder a los modelos Gemini 2.5 Pro y Gemini 2.5 Flash. Su funcionamiento es similar al de los scripts anteriores, aunque presenta algunas particularidades técnicas propias de la API de Google.

En primer lugar, la llamada a la API se realiza a través del endpoint específico de cada modelo, usando el método POST con un payload estructurado en formato JSON. A diferencia de OpenAI o Anthropic, Google organiza el contenido del prompt dentro de una jerarquía `contents > parts > text`, que debe seguirse estrictamente para que la API procese correctamente la solicitud.

En segundo lugar, la respuesta generada por el modelo se extrae de una estructura algo más compleja: se accede a la lista `candidates`, y dentro de ella se localiza el texto en `content >`

parts > text. Esta diferencia en la estructura de respuesta es una de las principales particularidades técnicas de Gemini frente a otras APIs.

Por último, al igual que en los demás scripts, el contenido generado se guarda en archivos .txt individuales, uno por modelo, dentro de una carpeta común. También se imprime el tiempo de ejecución, que será considerado en la evaluación comparativa.

Prueba I: parámetros estándar

Ejemplo de resultados

A continuación, se presentan dos ejemplos de respuestas generadas por el modelo o3-mini y GPT-4o de OpenAI al prompt definido en el experimento. Se muestran dos ejemplos para tener una referencia del estilo de respuestas pero no se expondrán a lo largo del proyecto los ejemplos de todos los modelos salvo que hay algún aspecto a destacar o alguna anomalía que merezca la pena analizar y se necesite visualizar la respuesta para ello.

Respuesta generada por o3-mini (OpenAI)

La corriente alterna (AC) es un tipo de corriente eléctrica en la que el flujo de carga cambia de dirección de forma periódica. Esto contrasta con la corriente continua (DC), en la que los electrones fluyen en una única dirección. A continuación, se detallan algunas características y aspectos técnicos relevantes para comprender la corriente alterna en el contexto de la ingeniería eléctrica:

1. Dirección y forma de onda:

En la AC, la magnitud del voltaje (y, por lo tanto, la corriente) varía de forma sinusoidal, aunque pueden encontrarse otras formas de onda (como cuadradas o triangulares) en aplicaciones específicas. En el caso más común, el voltaje se expresa como:

$$V(t) = V_0 \cdot \sin(2\pi ft + \varphi)$$

donde:

- V_0 es el valor máximo (amplitud).
- f es la frecuencia (en Hz), que determina la cantidad de ciclos por segundo.
- ϕ es la fase, que indica el desplazamiento en el tiempo de la onda.
- t es el tiempo.

2. Frecuencia:

La frecuencia es un parámetro fundamental de la AC. En la mayoría de las redes eléctricas de consumo se utiliza una frecuencia de 50 Hz.

Respuesta generada por *GPT-4o* (OpenAI)

La corriente alterna, comúnmente abreviada como AC por sus siglas en inglés (Alternating Current), es un tipo de corriente eléctrica en la que el flujo de carga eléctrica cambia de dirección periódicamente. A diferencia de la corriente continua (DC), donde el flujo de electrones es unidireccional, en la corriente alterna los electrones oscilan hacia adelante y hacia atrás.

Características de la Corriente Alterna:

1. Frecuencia: La frecuencia de una corriente alterna es el número de ciclos completos que ocurren en un segundo. Se mide en hertzios (Hz). Por ejemplo, en la mayoría de los países, la frecuencia estándar de la red eléctrica es de 50 Hz o 60 Hz.
2. Forma de onda: La forma de onda más común para la corriente alterna es la sinusoidal. Esta forma de onda es ideal porque permite una transmisión eficiente de energía y es fácil de generar y transformar. Sin embargo, también existen otras formas de onda como la cuadrada o la triangular, que se utilizan en aplicaciones específicas.
3. Valor eficaz: Dado que la corriente alterna varía con el tiempo, se utiliza el concepto de valor eficaz (o RMS, Root Mean Square) para describir su magnitud.

Ventajas de la Corriente Alterna:

- Transformación de voltaje mediante transformadores, fundamental para transmisión a larga distancia.
- Generación más sencilla y económica en grandes centrales.
- Infraestructura diseñada para AC en la mayoría de los sistemas eléctricos del mundo.

Aplicaciones: Hogares, industrias y comercios.

Conclusión: Comprender la corriente alterna es fundamental en ingeniería eléctrica, ya que constituye la base de la mayoría de los sistemas eléctricos actuales.

Prueba II: variación de temperatura

Con el objetivo de estudiar cómo afecta la aleatoriedad a la calidad de las respuestas generadas, en este experimento se analiza el impacto del parámetro *temperature* en el comportamiento de los modelos. Este parámetro, que toma valores entre 0 y 1, controla el grado de imprevisibilidad en la generación: valores bajos producen respuestas más deterministas y conservadoras, mientras que valores altos fomentan mayor creatividad y variabilidad en las salidas del modelo.

Para cada experimento se seleccionan dos modelos representativos con el fin de aplicar esta variación y observar diferencias en precisión, claridad y nivel técnico. La elección concreta de los modelos sobre los que se aplicará esta variación se justifica en función de los resultados obtenidos en la prueba anterior (*Prueba I: parámetros estándar*), de modo que se analicen casos relevantes por su rendimiento destacado, comportamiento inesperado o contraste entre modelos de distinta capacidad.

Metodología de evaluación

Para evaluar objetivamente la calidad de las respuestas generadas por los modelos, se ha implementado un sistema de comparación automática con dos respuestas de referencia elaboradas a partir de fuentes oficiales. Esta evaluación se realiza desde dos enfoques complementarios:

Similitud léxica mediante TF-IDF

Se utiliza la técnica TF-IDF (*Term Frequency – Inverse Document Frequency*) para vectorizar el contenido de cada respuesta, calculando después la similitud del coseno entre cada respuesta generada y la oficial. Este método permite detectar coincidencias literales en el contenido y es útil para evaluar correspondencias directas entre términos utilizados.

Similitud semántica mediante embeddings

Para una evaluación más profunda, se emplea el modelo all-MiniLM-L6-v2 de sentence-transformers, que transforma cada texto en un vector denso contextualizado. Luego, se calcula la similitud del coseno entre estos vectores. Este método permite comparar el significado general de los textos, incluso si están redactados de forma distinta.

El siguiente script en Python automatiza este proceso, recorriendo todas las respuestas generadas en una carpeta específica y calculando sus similitudes con cada una de las respuestas oficiales según ambos métodos:

```
import os
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sentence_transformers import SentenceTransformer, util

# Define la ruta donde se encuentran las respuestas
carpeta = "respuestas_temperatura"

# Carga las respuestas generadas por los modelos (excluye las oficiales)
respuestas = {}
for archivo in os.listdir(carpeta):
    if archivo.endswith(".txt") and not
    archivo.startswith("Respuesta_oficial"):
        with open(os.path.join(carpeta, archivo), "r",
encoding="utf-8") as f:
            respuestas[archivo] = f.read()
```

```
ICAI ICADE CIHS

# Carga las respuestas oficiales
respuesta_oficial_1 = open(os.path.join(carpeta,
"Respuesta_oficial.txt"), encoding="utf-8").read()
respuesta_oficial_2 = open(os.path.join(carpeta,
"Respuesta_oficial_2.txt"), encoding="utf-8").read()

# ----- TF-IDF sin embeddings -----
print("Similitudes del coseno con la respuesta oficial:
Respuesta_oficial.txt (TF-IDF)")
for nombre_modelo, texto in respuestas.items():
    vectorizer = TfidfVectorizer().fit([texto,
respuesta_oficial_1])
    vectores = vectorizer.transform([texto,
respuesta_oficial_1])
    similitud = cosine_similarity(vectores[0],
vectores[1])[0][0]
    print(f"{nombre_modelo:<40} → Similitud:
{similitud:.4f}")

print("\nSimilitudes del coseno con la respuesta oficial:
Respuesta_oficial_2.txt (TF-IDF)")
for nombre_modelo, texto in respuestas.items():
    vectorizer = TfidfVectorizer().fit([texto,
respuesta_oficial_2])
    vectores = vectorizer.transform([texto,
respuesta_oficial_2])
    similitud = cosine_similarity(vectores[0],
vectores[1])[0][0]
    print(f"{nombre_modelo:<40} → Similitud:
{similitud:.4f}")

# ----- Con embeddings -----
# Carga el modelo de SentenceTransformer
modelo_embeddings = SentenceTransformer("all-MiniLM-L6-v2")

# Codifica las respuestas oficiales como vectores de
embedding
emb_oficial_1 =
modelo_embeddings.encode(respuesta_oficial_1,
convert_to_tensor=True)
```

```
emb_oficial_2 =
modelo_embeddings.encode(respuesta_oficial_2,
convert_to_tensor=True)

# Compara cada respuesta con la oficial 1
print("\nSimilitudes del coseno con la respuesta oficial:
Respuesta_oficial.txt (Embeddings)")
for nombre_modelo, texto in respuestas.items():
    emb = modelo_embeddings.encode(texto,
convert_to_tensor=True)
    similitud = util.cos_sim(emb, emb_oficial_1).item()
    print(f"{nombre_modelo:<40} → Similitud:
{similitud:.4f}")

# Compara cada respuesta con la oficial 2
print("\nSimilitudes del coseno con la respuesta oficial:
Respuesta_oficial_2.txt (Embeddings)")
for nombre_modelo, texto in respuestas.items():
    emb = modelo_embeddings.encode(texto,
convert_to_tensor=True)
    similitud = util.cos_sim(emb, emb_oficial_2).item()
    print(f"{nombre_modelo:<40} → Similitud:
{similitud:.4f}")
```

El script anterior implementado permite evaluar automáticamente la calidad de las respuestas generadas por los modelos de lenguaje mediante el cálculo de la similitud con dos respuestas oficiales de referencia. Para ello, se utilizan dos enfoques diferentes y complementarios que permiten medir la cercanía entre textos desde una perspectiva léxica y semántica. En primer lugar, se aplica la técnica TF-IDF (Term Frequency – Inverse Document Frequency), que transforma cada texto en un vector en función de la frecuencia relativa de sus palabras. A partir de estos vectores, se calcula la similitud del coseno, una métrica que mide el ángulo entre ellos y cuya interpretación es directa: cuanto más cercano a 1 sea el valor, mayor será la coincidencia literal entre los textos. Este método permite detectar correspondencias directas en el uso del lenguaje, pero no capta el significado global del mensaje.

En segundo lugar, se emplea un modelo de *sentence embeddings* contextualizados (all-MiniLM-L6-v2 de la librería sentence-transformers), que genera representaciones

vectoriales densas y de dimensión fija a partir de textos completos. Estos vectores no se basan únicamente en la frecuencia de las palabras, sino que integran relaciones semánticas y dependencias contextuales entre términos. También en este caso se calcula la similitud del coseno entre la respuesta generada por el modelo y la respuesta oficial, obteniendo así una medida de proximidad conceptual que resulta útil para identificar textos que, aunque no estén redactados de manera idéntica, transmiten ideas equivalentes.

El funcionamiento del script es el siguiente: se recorre de forma automática una carpeta que contiene archivos .txt con las respuestas generadas por los modelos y se filtran aquellas que no corresponden a respuestas oficiales. Para cada archivo, se compara su contenido con cada una de las dos respuestas oficiales, primero mediante TF-IDF y luego mediante embeddings. En cada comparación se calcula el valor de la similitud del coseno y se imprime en pantalla el nombre del modelo y el resultado obtenido. De este modo, se obtiene una doble medida de evaluación para cada respuesta, que permite analizar de forma más completa la calidad, coherencia y adecuación del contenido generado por los distintos modelos de lenguaje.

A partir de este punto, se presentan el resto de experimentos realizados. Para evitar repeticiones innecesarias, solo se profundizará en los aspectos que varían respecto al experimento anterior, manteniéndose constante la metodología general de evaluación.

5.2 EXPERIMENTO 02: REDACCIÓN DE INFORME DE LABORATORIO

DESCRIPCIÓN Y OBJETIVOS DEL EXPERIMENTO

El segundo experimento tiene como objetivo evaluar la capacidad de los modelos de lenguaje para redactar secciones técnicas de un informe de laboratorio. Se busca determinar cuáles modelos presentan un mejor desempeño en la generación de documentación técnica de calidad, con el fin de explorar su posible aplicación en la automatización de tareas de redacción en el ámbito de la ingeniería.

Dado el límite de tokens y capacidad de las llamadas a la API, se ha acotado la tarea a la redacción de la sección de **“Resultados y Discusión”**, una parte especialmente relevante del

informe, ya que no solo requiere claridad expositiva, sino también interpretación de datos, razonamiento técnico y conclusiones fundamentadas.

Aunque el experimento de referencia es el del **efecto fotoeléctrico**, el enfoque adoptado es generalizable a otros informes técnicos en distintas disciplinas. Este caso de estudio permite evaluar con precisión la capacidad de los modelos para generar textos académicos complejos y estructurados a partir de datos experimentales.

DISEÑO DEL PROMPT

Para este experimento se ha seguido un proceso similar al descrito en el apartado anterior, realizando múltiples pruebas con un modelo de capacidad intermedia (GPT-4) para garantizar que el prompt fuera lo suficientemente robusto y pudiera generalizarse a otros modelos con distintas arquitecturas.

Durante este proceso, se probaron varias formulaciones iniciales y se observó cómo influían en la calidad de las respuestas generadas. Por ejemplo:

6. Prompt inicial: *“Resume los resultados del experimento del efecto fotoeléctrico”* → generaba una respuesta superficial, sin análisis ni estructura clara.
7. Prompt intermedio: *“Redacta un breve informe sobre el experimento del efecto fotoeléctrico”* → ofrecía mayor detalle, pero sin separar correctamente las secciones esperadas.

Tras iteraciones sucesivas, se definió el siguiente prompt como versión final:

Redacta la sección “Resultados y Discusión” de un informe de laboratorio técnico-académico sobre el efecto fotoeléctrico. Utiliza un lenguaje formal, técnico y claro. La sección debe incluir: (1) presentación de datos obtenidos, (2) análisis cualitativo de los resultados, (3) discusión del ajuste lineal, y (4) conclusión técnica.

Este enunciado se utilizó de forma homogénea en todos los modelos para asegurar comparabilidad entre las respuestas generadas.

DISEÑO Y FUENTE DE LA RESPUESTA MODELO PARA COMPARACIÓN

La respuesta modelo utilizada como referencia en este experimento ha sido adaptada a partir de un informe académico real sobre el efecto fotoeléctrico, elaborado por la Universidad de Groningen en el marco de la asignatura *Physics Laboratory I* [35]. Este informe incluye una sección completa de “Resultados y Discusión”, estructurada según los estándares habituales de redacción científica, e incorpora tanto la presentación de datos como su interpretación y análisis técnico.

A partir de este material, se ha redactado una versión simplificada y estandarizada que actúa como respuesta de referencia para evaluar las salidas generadas por los modelos. Esta adaptación conserva la estructura técnica esperada en un entorno universitario, lo que permite aplicar sobre ella comparaciones tanto cualitativas como cuantitativas, especialmente mediante medidas de similitud semántica como la del coseno con embeddings.

DESARROLLO EXPERIMENTAL

El procedimiento de generación de respuestas y evaluación se mantiene idéntico al descrito en el Experimento 01. Se ha utilizado el mismo script para enviar los prompts a los modelos GPT, Claude y Gemini, almacenar sus respuestas y calcular la similitud con la respuesta modelo mediante TF-IDF y embeddings (all-MiniLM-L6-v2).

En este experimento, la única variación ha sido el prompt utilizado, adaptado específicamente para solicitar la sección “Resultados y Discusión” de un informe de laboratorio técnico.

5.3 EXPERIMENTO 03: PROBLEMA DE FÍSICA

DESCRIPCIÓN Y OBJETIVOS DEL EXPERIMENTO

El tercer experimento ha consistido en evaluar a los modelos en la resolución de problemas físicos, centrándonos concretamente en el campo magnético. En este caso, se ha planteado

una situación clásica de equilibrio en el contexto del electromagnetismo: una espira circular apoyada sobre una superficie horizontal y una pared vertical, sometida a un campo magnético uniforme perpendicular a su plano. Se pide a los modelos que analicen las fuerzas y momentos que actúan sobre la espira para determinar la condición de equilibrio. Se trata de un problema correspondiente a la asignatura de Física de 1º de ICAI.

Profundizando en los objetivos específicos de este experimento, se busca comprobar si los modelos son capaces de identificar correctamente el momento magnético de la espira, aplicar el equilibrio de momentos respecto a un eje y obtener la condición de estabilidad del sistema.

Con este experimento obtenemos una visión clara del potencial de los modelos como herramienta de asistencia en la resolución de problemas en el ámbito de la física aplicada a la ingeniería.

DISEÑO DEL PROMPT

El prompt de este experimento ha sido diseñado para evaluar la capacidad de los modelos de lenguaje a la hora de resolver un problema físico avanzado, manteniendo claridad conceptual, coherencia lógica y rigor matemático. Como en los casos anteriores, se ha seguido un proceso iterativo de pruebas sobre un modelo intermedio (GPT-4) para ajustar la formulación del enunciado, asegurando que la redacción final fuera capaz de guiar al modelo sin inducir sesgos ni simplificaciones innecesarias.

El prompt definitivo utilizado en el experimento es el siguiente:

Resuelve el siguiente problema de física de manera clara, ordenada y con enfoque universitario:

Una espira circular de radio R y masa m es recorrida por una corriente constante I . La espira apoya en el suelo y en una pared, formando con esta un ángulo θ . Un campo magnético uniforme B paralelo al suelo atraviesa dicha espira.

Se pide:

1. Calcular el valor del momento magnético que sufre la espira.
2. Determinar el valor del campo B para que la espira se mantenga en equilibrio en su posición.

Explica cada paso del razonamiento físico y matemático, usando las fórmulas necesarias.

DISEÑO Y FUENTE DE LA RESPUESTA MODELO PARA COMPARACIÓN

La respuesta modelo utilizada como referencia en este experimento ha sido elaborada a partir del **Ejemplo 03**, ver en Anexos, de un examen oficial de la asignatura de Física Aplicada de ICAI. Este ejercicio aborda el equilibrio de una espira circular apoyada en el suelo y en una pared bajo la acción de un campo magnético uniforme, y requiere aplicar conceptos de momento dipolar magnético, fuerza magnética y equilibrio de momentos. Puede consultarse la respuesta completa en el Anexo.

DESARROLLO EXPERIMENTAL

El procedimiento de generación de respuestas y evaluación se ha mantenido idéntico al descrito en los experimentos anteriores. Se ha utilizado el mismo script para enviar el prompt a los modelos GPT-3.5, GPT-4, GPT-4o y o3-mini a través de la API de OpenAI, recoger las respuestas y guardarlas en archivos individuales para su posterior análisis.

La evaluación se ha realizado mediante las métricas habituales de similitud semántica (TF-IDF y embeddings all-MiniLM-L6-v2), comparando cada respuesta generada con la solución oficial

5.4 EXPERIMENTO 04: PROBLEMA CON INPUT DE IMAGEN

DESCRIPCIÓN Y OBJETIVOS DEL EXPERIMENTO

Aunque estamos valorando diferentes versiones de los modelos, que tienen aplicaciones para las que funcionan mejor o peor, existen versiones claramente más recientes y avanzadas (a nivel general, lo cual no implica necesariamente un mejor rendimiento en una aplicación

concreta, que es precisamente lo que se estudia en este trabajo). Entre estas versiones destaca GPT-4o. Este modelo permite recibir imágenes como input, lo que abre nuevas posibilidades muy interesantes, como evaluar su capacidad para resolver un problema que incluye una imagen en el enunciado y que requiere interpretarla o analizarla para extraer datos fundamentales para su resolución.

En concreto, se presenta un problema clásico de física en el que se combinan un **campo eléctrico (E)** y un **campo magnético (B)**. Tal como se muestra en la **ilustración 7**, se representa una situación en la que una partícula cargada q entra con cierta **velocidad $\vec{v}$** en una región delimitada por dos planos paralelos. Entre ellos se ha generado un **campo eléctrico $\vec{E}$** , perpendicular a los planos y dirigido de arriba hacia abajo. Al mismo tiempo, existe un **campo magnético uniforme $\vec{B}$** saliendo del plano (representado por los círculos con cruces). Se busca comprobar si los modelos son capaces de integrar correctamente la información visual con conceptos físicos, estructurando un razonamiento lógico que les lleve a una conclusión coherente.

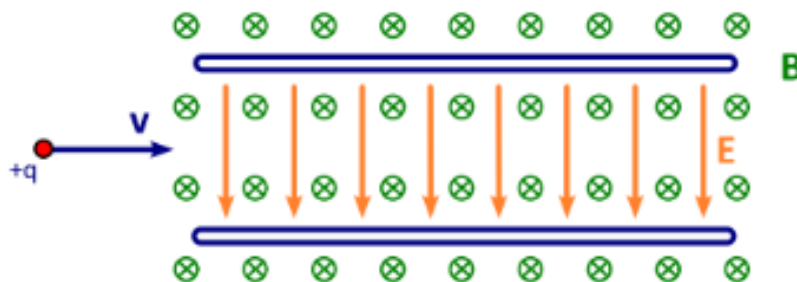


Ilustración 7. Imagen Problema magnetismo I

La ilustración 7, anteriormente explicada, constituye el input visual proporcionado al modelo, acompañado de un prompt textual que contiene el enunciado detallado del ejercicio. De este modo, el modelo debe procesar conjuntamente la información visual y textual para resolver el problema. Puede ampliarse información sobre este problema (enunciado y solución oficial) en **Anexos**.

En ingeniería, una gran variedad de problemas requiere la interpretación de imágenes o gráficos. Por ello, resulta especialmente relevante evaluar la capacidad de estos modelos para este tipo de tareas, así como identificar qué versiones y parámetros ofrecen mejores resultados para su aplicación en este contexto.

DISEÑO DEL PROMPT

El prompt ha sido diseñado específicamente para evaluar la capacidad de los modelos multimodales de integrar información visual con razonamiento físico. En este caso, el enunciado plantea una configuración clásica de campos eléctricos y magnéticos cruzados, acompañada de una imagen que representa la situación. El modelo debe interpretar correctamente la escena, identificar las direcciones de los vectores involucrados (E , B , v), aplicar la ley de Lorentz y deducir simbólicamente la condición para que una partícula cargada atraviere sin desviarse.

El texto del prompt guía al modelo paso a paso e indica de forma explícita qué fórmulas debe utilizar, con el objetivo de obtener una resolución estructurada, formal y técnicamente coherente. El enunciado final fue el siguiente:

Esta imagen muestra dos planos paralelos con un campo eléctrico (E) hacia abajo y un campo magnético (B) perpendicular hacia dentro. Una partícula con carga positiva entra con velocidad v en la dirección horizontal. Resuelve paso a paso qué velocidad debe tener para que atraviere sin desviarse, utilizando la ley de Lorentz y el equilibrio de fuerzas. Incluye fórmulas físicas como $F = qE$ y $F = qvB$, y concluye con la expresión simbólica $v = E / B$.

DISEÑO Y FUENTE DE LA RESPUESTA MODELO PARA COMPARACIÓN

La respuesta modelo utilizada como referencia en este experimento corresponde a la resolución del **Ejemplo 01**, ver anexos, del conjunto de ejercicios de electromagnetismo proporcionados por el Departamento de Física de ICAI. El enunciado plantea el equilibrio de una partícula cargada que se desplaza en un sistema con campos eléctricos y magnéticos

perpendiculares, y la resolución requiere aplicar correctamente la ley de Lorentz e igualar las fuerzas implicadas para deducir la velocidad necesaria para que la partícula no se desvíe.

DESARROLLO EXPERIMENTAL

El procedimiento seguido en este experimento es similar al de casos anteriores, con la particularidad de que en este caso se ha utilizado un modelo multimodal, GPT-4o, capaz de procesar simultáneamente texto e imagen. Para ello, se ha desarrollado un script en Python que codifica la imagen del enunciado (ProblemaMagnetismo1.jpg) en formato base64, conforme a las especificaciones de la API de OpenAI, y la incluye junto con el texto del prompt como parte del input del modelo.

```
from openai import OpenAI
import base64
import time
import os

# Inicializa el cliente de OpenAI con API Key
client = OpenAI(
    api_key="sk-..."
)

# Carga y codifica la imagen en base64
with open("ProblemaMagnetismo1.jpg", "rb") as img_file:
    base64_img =
base64.b64encode(img_file.read()).decode("utf-8")

# Crea la URL en formato base64 para enviar a la API
data_url = f"data:image/jpeg;base64,{base64_img}"

# Define el prompt que acompaña a la imagen
prompt_text = (
    "Esta imagen muestra dos planos paralelos con un campo
eléctrico (E) hacia abajo y un campo magnético (B)
perpendicular hacia dentro. "
    "Una partícula con carga positiva entra con velocidad v
en la dirección horizontal. "
```

```
ICAI ICADE CIHS

"Resuelve paso a paso qué velocidad debe tener para que
atraviase sin desviarse, "
"utilizando la ley de Lorentz y el equilibrio de
fuerzas. "
"Incluye fórmulas físicas como  $F = qE$  y  $F = qvB$ , y
concluye con la expresión simbólica  $v = E / B$ ."
)

# Mide el tiempo de generación
start_time = time.time()

# Realiza la solicitud al modelo con imagen + texto
response = client.chat.completions.create(
    model="gpt-4o",
    messages=[
        {
            "role": "user",
            "content": [
                {"type": "text", "text": prompt_text},
                {"type": "image_url", "image_url": {"url":
data_url}}
            ]
        }
    ],
    max_tokens=700,
    temperature=0.3
)

# Extrae el contenido de la respuesta y mide el tiempo
total
respuesta = response.choices[0].message.content.strip()
tiempo = round(time.time() - start_time, 2)

# Muestra la respuesta por consola
print("\nRespuesta del modelo:\n")
print(respuesta)
print(f"\n↩ Generado en {tiempo} segundos.")

# Guarda la respuesta en un archivo de texto
output_folder = "respuestas"
```

```
os.makedirs(output_folder, exist_ok=True)
output_path = os.path.join(output_folder, "gpt-
4o_imagen_magnetismo.txt")

with open(output_path, "w", encoding="utf-8") as f:
    f.write(respuesta)
```

5.5 EXPERIMENTO 05: EVALUACIÓN DE UN PROCESO SIMULADO

DESCRIPCIÓN Y OBJETIVOS DEL EXPERIMENTO

En el sector de la ingeniería, muchas aplicaciones están orientadas a los procesos. Por ello, un gran avance sería involucrar a los modelos de lenguaje en la revisión de dichos procesos. Es decir, una vez diseñado y simulado un proceso, contar con la capacidad de verificar que todo funciona correctamente, identificar posibles anomalías y detectar fallos potenciales.

De esta necesidad, o posible oportunidad de aplicación de los LLMs en entornos industriales, surge este experimento, en el que se simula un proceso mediante la librería Simpy y se proporciona dicha simulación como input en el prompt de los modelos, con el objetivo de que evalúen el proceso simulado. Se busca así valorar su capacidad para analizar y diagnosticar procesos, lo cual, en un futuro, podría extrapolarse a aplicaciones reales con supervisión humana.

DISEÑO DEL PROMPT

El diseño del prompt en este experimento ha seguido un enfoque distinto al de los anteriores. En lugar de construir un problema teórico, se ha generado un log realista de una línea de montaje simulada con simpy, y este log ha sido incluido de forma completa como entrada para el modelo.

El prompt solicita al modelo que analice este registro de eventos desde el punto de vista de un ingeniero de procesos, con el objetivo de identificar comportamientos anómalos, cuellos de botella o posibles fallos. Se especifica en el rol del sistema que el modelo debe

comportarse como un ingeniero especializado, y se le pide una justificación técnica, estructurada y clara.

Esta aproximación permite evaluar no solo la comprensión semántica del modelo, sino también su capacidad para detectar patrones temporales, interpretar secuencias y razonar sobre procesos industriales a partir de datos.

DISEÑO Y FUENTE DE LA RESPUESTA MODELO PARA COMPARACIÓN

Para realizar una evaluación objetiva de las respuestas generadas, se ha elaborado una respuesta modelo de referencia, basada en el análisis manual del log simulado. Esta respuesta identifica correctamente el comportamiento anómalo introducido (un tiempo excesivo en la estación 3 para el Producto 6), explica el impacto operativo que puede suponer un cuello de botella, y plantea recomendaciones razonadas para detectar y evitar fallos similares en entornos reales.

DESARROLLO EXPERIMENTAL

En este experimento se ha seguido una aproximación diferente, enfocada a evaluar si los modelos de lenguaje son capaces de analizar y diagnosticar un proceso simulado, tal como se haría en una línea de producción real.

La simulación se ha desarrollado utilizando la librería `simpy` de Python, que permite modelar sistemas discretos y gestionar eventos temporales de forma precisa. Se simula una línea de montaje compuesta por **tres estaciones**, por las que pasan **diez productos** secuencialmente. En el caso del **Producto 6**, se introduce un **tiempo anómalo de 20 segundos** en la tercera estación, con el objetivo de representar un posible fallo u obstrucción del proceso.

El código fuente de la simulación se incluye a continuación:

```
import simpy
import random

# Define el proceso de montaje de un producto a través de
las estaciones
```

```
def proceso_montaje(env, nombre, linea, tiempos, log):
    for i, t in enumerate(tiempos):
        with linea[i].request() as req:
            yield req
            log.append(f"{env.now:.2f} - {nombre} entra en
estación {i+1}")
            yield env.timeout(t)
            log.append(f"{env.now:.2f} - {nombre} sale de
estación {i+1}")

# Define la función principal de simulación
def simulacion_linea_de_produccion():
    env = simpy.Environment()
    log = []

    # Crea una línea con 3 estaciones de capacidad 1
    estaciones = [simpy.Resource(env, capacity=1) for _ in
range(3)]

    # Define la lógica de generación de productos
    def generar_productos():
        for i in range(10):
            tiempos = [
                random.uniform(2, 4),      # Tiempo aleatorio
en estación 1
                random.uniform(4, 6),      # Tiempo aleatorio
en estación 2
                20 if i == 5 else random.uniform(3, 5) #
Falla intencionada en el Producto_6
            ]
            env.process(proceso_montaje(env,
f"Producto_{i+1}", estaciones, tiempos, log))
            yield env.timeout(1) # Espera entre
lanzamientos de productos

    # Inicia el proceso de generación
    env.process(generar_productos())

    # Ejecuta la simulación durante 100 unidades de tiempo
    env.run(until=100)
```

```
return log

# Ejecuta la simulación y guarda el log
log_simulacion = simulacion_linea_de_produccion()

# Crea la carpeta de salida si no existe
os.makedirs("respuestas", exist_ok=True)

# Guarda el log en un archivo de texto
with open("respuestas/simpy_fallo_exp06_log.txt", "w",
encoding="utf-8") as f:
    f.write("\n".join(log_simulacion))
```

El log resultante de la simulación, que se utiliza como entrada directa para el modelo, es el siguiente (fragmento relevante):

...

30.14 - Producto_5 entra en estación 3

30.14 - Producto_6 entra en estación 2

33.88 - Producto_5 sale de estación 3

35.23 - Producto_6 sale de estación 2

35.23 - Producto_6 entra en estación 3

...

55.23 - Producto_6 sale de estación 3

Como puede observarse, el Producto 6 permanece aproximadamente 20 segundos en la estación 3, mientras que el resto de productos completan esa etapa en menos de 5 segundos. Este comportamiento constituye el fallo intencionado que el modelo debe identificar y justificar.

A diferencia de otros experimentos del presente trabajo, en este caso no se realiza una variación del parámetro *temperature* en los modelos evaluados. Esto se debe a que el objetivo del experimento no es generar respuestas variadas, sino comprobar si los modelos son capaces de detectar correctamente una anomalía específica y objetiva en un log de simulación. Dado que se trata de una tarea diagnóstica y analítica, se opta por mantener la temperatura en un valor bajo (0.3).

Capítulo 6. ANÁLISIS DE RESULTADOS

Esta sección recoge el análisis de resultados y las conclusiones de los seis experimentos realizados en el capítulo anterior. Se comparan los resultados obtenidos, se destacan sus implicaciones y se interpreta qué significan en el contexto del proyecto. Asimismo, se utilizan estos resultados para evaluar qué objetivos planteados inicialmente se han conseguido cumplir y cuáles son las principales aportaciones que derivan de ellos.

6.1 *EVALUACIÓN EXPERIMENTO 01*

Este primer experimento tenía como objetivo evaluar la **capacidad explicativa técnica** de los modelos de lenguaje en relación con un concepto fundamental de la ingeniería eléctrica: la corriente alterna (AC). Para ello, se utilizaron dos pruebas complementarias:

- **Prueba 1:** análisis con parámetros estándar y evaluación mediante tiempo de respuesta y similitud del coseno.
- **Prueba 2:** análisis de sensibilidad al parámetro de temperatura.

Se puede consultar el resultado del experimento en **Experimento 01: Explicación de concepto técnico**.

6.1.1 PRUEBA 1 – PARÁMETROS ESTÁNDAR

Tiempo de respuesta

Modelo	Tiempo de respuestas
GPT 2	18.82
GPT-3.5-turbo	2.62
GPT-4	15.27

ICAI	ICADE	CIHS
GPT-4o	7.93	
o3-mini	5.29	
Claude-3.5-sonnet	14.76	
Claude-3-Haiku	4.23	
Gemini-2.5-flash	22.44	
Gemini-2.5-pro	35.58	

Tabla 3. Resultados tiempos de respuesta experimento 01

Como se recoge en la Tabla 4, el tiempo de respuesta varía significativamente entre los distintos modelos de lenguaje evaluados. Los modelos más ligeros como GPT-3.5-turbo, o3-mini y Claude-3-Haiku ofrecen respuestas en menos de 6 segundos, destacando por su eficiencia computacional. Esto se debe a que están optimizados para tareas rápidas, con arquitecturas menos complejas o con menor número de parámetros, lo que permite una generación de texto más ágil.

En el extremo opuesto, modelos más sofisticados como Gemini-2.5-pro y Gemini-2.5-flash requieren tiempos de ejecución considerablemente mayores, alcanzando los 35,58 s y 22,44 s, respectivamente. Este aumento en el tiempo de respuesta es coherente con su mayor capacidad de razonamiento y generación, ya que suelen incorporar arquitecturas más avanzadas y pesadas. GPT-4 también presenta un tiempo elevado (15,27 s), en línea con su posicionamiento como modelo con capacidad de comprensión compleja.

Por tanto, se observa una relación directa entre la **complejidad del modelo y su tiempo de respuesta**. Los modelos más potentes tienden a tardar más en generar respuestas, lo que sugiere una compensación entre velocidad y profundidad en la generación de contenido. Esta diferencia puede ser relevante en aplicaciones donde el tiempo de respuesta es crítico, como en asistentes en tiempo real, frente a otras donde la precisión técnica es prioritaria.

Análisis con TF-IDF

Modelo	Similitud del coseno Respuesta Oficial 1	Similitud del coseno Respuesta Oficial 2
GPT 2	0.233	0.247
GPT-3.5-turbo	0.7553	0.7719
GPT-4	0.7366	0.7766
GPT-4o	0.8303	0.8069
o3-mini	0.7549	0.7355
Claude-3.5-sonnet	0.5976	0.6239
Claude-3-Haiku	0.8010	0.7582
Gemini-2.5-flash	0.7783	0.7348
Gemini-2.5-pro	0.8033	0.7597

Tabla 4. Resultados de evaluación con TF-IDF experimento 01, prueba 1

Los resultados obtenidos a partir de la similitud del coseno utilizando vectores TF-IDF permiten identificar qué modelos generan respuestas más cercanas a las oficiales en términos de redacción, terminología y estructura técnica. En esta prueba se evalúa cada modelo frente a dos respuestas oficiales de referencia, mostrando cómo varía su rendimiento y si los resultados son coherentes con las expectativas según el tipo de modelo.

El modelo **GPT-4o** obtiene las puntuaciones más altas frente a ambas respuestas oficiales, con valores de 0.8303 y 0.8069 respectivamente. Este resultado confirma su capacidad para generar respuestas alineadas con el estilo técnico de referencia. GPT-4o es actualmente uno

de los modelos más avanzados de OpenAI, diseñado para optimizar tanto la calidad como la eficiencia, lo que se traduce en explicaciones precisas, completas y bien estructuradas.

También destacan **Claude-3-Haiku** (0.8010 y 0.7582) y **Gemini-2.5-pro** (0.8033 y 0.7597), ambos modelos de última generación desarrollados por Anthropic y Google. Estos modelos están entrenados con grandes volúmenes de datos y afinados para tareas de generación instructiva, lo que se refleja en su alta similitud con las respuestas oficiales. Claude-3-Haiku, a pesar de ser la versión ligera de la familia Claude, demuestra un comportamiento muy competitivo, lo que indica una arquitectura eficiente y un entrenamiento optimizado para tareas explicativas. Gemini-2.5-pro, por su parte, confirma su capacidad para interpretar correctamente prompts técnicos y estructurar la información de manera acorde con patrón original.

En una posición intermedia se encuentran **GPT-3.5-turbo**, **GPT-4**, **Gemini-2.5-flash** y **o3-mini**, con valores comprendidos entre 0.73 y 0.78. GPT-3.5-turbo muestra un rendimiento muy estable (0.7553 y 0.7719), lo que refleja su capacidad para generar explicaciones claras, aunque con menor profundidad que modelos más recientes. **GPT-4**, pese a ser más potente, presenta una menor coincidencia literal (0.7366 y 0.7766), ya que su estrategia de generación introduce con frecuencia matices adicionales o reorganiza el contenido para enriquecerlo, lo que reduce la similitud medida con TF-IDF. Este mismo efecto se observa en **Gemini-2.5-flash**, un modelo orientado a la rapidez, que tiende a simplificar o resumir parte de la información, lo que afecta a su puntuación (0.7783 y 0.7348). **o3-mini**, un modelo compacto de OpenAI, ofrece resultados notablemente buenos (0.7549 y 0.7355), lo que indica una respuesta eficaz y una redacción directa, bien ajustada al prompt.

El modelo con peor rendimiento en esta métrica es **Claude-3.5-sonnet** (0.5976 y 0.6239). Este modelo tiende a adoptar un estilo más interpretativo y menos literal, reestructurando con frecuencia el contenido, lo que reduce considerablemente su coincidencia con el texto oficial. Su comportamiento se aleja del patrón técnico clásico esperado en este tipo de tareas.

En conjunto, los resultados demuestran que la fidelidad estructural, el uso de terminología técnica precisa y una redacción clara y directa son factores clave para alcanzar una alta

similitud con el texto oficial cuando se utiliza TF-IDF. Además, se confirma que no siempre los modelos más grandes o recientes ofrecen la mayor alineación si están diseñados para enriquecer el contenido de manera creativa. Modelos bien calibrados y directos, como GPT-4o o Claude-3-Haiku, consiguen una mayor fidelidad al objetivo instructivo del experimento.

Análisis con embeddings

Modelo	Similitud del coseno Respuesta Oficial 1	Similitud del coseno Respuesta Oficial 2
GPT 2	0.3030	0.2981
GPT-3.5-turbo	0.8654	0.8546
GPT-4	0.8223	0.7319
GPT-4o	0.8646	0.8335
o3-mini	0.8783	0.8043
Claude-3.5-sonnet	0.7715	0.7668
Claude-3-Haiku	0.8322	0.7761
Gemini-2.5-flash	0.8212	0.7649
Gemini-2.5-pro	0.7312	0.7303

Tabla 5. Resultados de evaluación contextualizada experimento 01, prueba 1

Al introducir el contexto en el análisis mediante embeddings, se observa un cambio generalizado en los valores de similitud del coseno respecto al análisis previo con TF-IDF. En la mayoría de los modelos, la similitud con las respuestas oficiales aumenta, lo que indica que la inclusión de contexto permite capturar con mayor precisión el significado general de las respuestas.

El modelo **o3-mini** obtiene los valores más altos de similitud con ambas respuestas oficiales (0.8783 y 0.8043), lo que resulta especialmente destacable considerando su menor tamaño respecto a otros modelos más potentes. Este buen desempeño sugiere que o3-mini es especialmente eficiente para tareas que requieren una comprensión global del texto y que su entrenamiento ha optimizado adecuadamente la retención de significado general.

GPT-3.5-turbo y **GPT-4o** presentan también un desempeño muy sólido, con valores superiores a 0.85 en al menos una de las dos comparaciones. En el caso de **GPT-3.5-turbo**, logra 0.8654 con la primera respuesta oficial y 0.8546 con la segunda, posicionándose como uno de los modelos más consistentes del análisis. **GPT-4o**, por su parte, mejora notablemente su rendimiento respecto al análisis anterior (TF-IDF), lo que sugiere que el modelo se beneficia especialmente del enfoque contextualizado.

Claude-3-Haiku mantiene una buena posición con valores elevados (0.8322 y 0.7761), alineándose con su buen comportamiento en el análisis anterior. En cambio, **Claude-3.5-Sonnet**, a pesar de ser una versión más avanzada, muestra un rendimiento algo más bajo, con valores de 0.7715 y 0.7668. Esto refuerza la hipótesis de que Haiku, aunque más ligero, está mejor optimizado para tareas de respuesta breve o instruccional como la planteada en este experimento.

Los modelos **Gemini-2.5**, tanto en su versión flash como pro, muestran rendimientos más discretos. **Gemini-2.5-pro** es el modelo con los valores más bajos en ambas comparaciones (0.7312 y 0.7303), mientras que **Gemini-2.5-flash** alcanza valores algo superiores pero sigue quedando por debajo de la mayoría de modelos de OpenAI y Anthropic. Esto sugiere que, en esta tarea específica, los modelos de Google ofrecen una comprensión menos precisa del contexto semántico general, a pesar de su tamaño y complejidad.

En conjunto, el análisis contextualizado confirma que los modelos que presentan mejores resultados en esta evaluación (como o3-mini, GPT-3.5-turbo o Claude-3-Haiku) son aquellos que mantienen una alta fidelidad semántica respecto a las respuestas oficiales, lo que resulta clave para tareas instructivas y de soporte técnico en ingeniería.

Por último, cabe destacar el caso de **GPT-2**, que obtiene valores significativamente inferiores al resto de modelos tanto en el análisis TF-IDF como en el análisis con embeddings (0.3030 y 0.2981). Aunque su rendimiento es claramente limitado en comparación con los modelos actuales, se ha incluido como punto de referencia histórico, con el objetivo de ilustrar de forma cuantitativa el avance sustancial que ha experimentado el campo de los modelos de lenguaje en los últimos años. Debido a sus limitaciones estructurales y de generación, GPT-2 no será considerado en los siguientes experimentos, que se centrarán exclusivamente en modelos contemporáneos con capacidad explicativa y contextual más avanzada.

6.1.2 PRUEBA 2 – VARIACIÓN DE TEMPERATURA

El objetivo de esta prueba es analizar el impacto que tiene la temperatura, uno de los parámetros más relevantes en los modelos de lenguaje, sobre la calidad de las respuestas generadas.

El análisis se realiza variando la temperatura en tres niveles: 0.2, 0.5 y 0.8, cubriendo así desde un comportamiento determinista hasta uno más creativo.

ANÁLISIS CON TF-IDF

Modelo	Similitud del coseno Respuesta Oficial 1	Similitud del coseno Respuesta Oficial 2
GPT-3.5-turbo-temp0.8	0.7949	0.7722
GPT-3.5-turbo-temp0.5	0.7482	0.7444
GPT-3.5-turbo-temp0.2	0.7542	0.7521
GPT-4-turbo-temp0.8	0.7534	0.7401
GPT-4-turbo-temp0.5	0.8245	0.8214
GPT-4-turbo-temp0.2	0.7998	0.7787

Tabla 6. Resultados de evaluación con TF-IDF experimento 01, prueba 2

Análisis con embeddings

Modelo	Similitud del coseno Respuesta Oficial 1	Similitud del coseno Respuesta Oficial 2
GPT-3.5-turbo-temp0.8	0.8754	0.8889
GPT-3.5-turbo-temp0.5	0.8803	0.8651
GPT-3.5-turbo-temp0.2	0.8661	0.8151
GPT-4-turbo-temp0.8	0.8023	0.7824
GPT-4-turbo-temp0.5	0.8007	0.8490
GPT-4-turbo-temp0.2	0.7624	0.7772

Tabla 7. Resultados de evaluación contextualizada experimento 01, prueba 2

6.2 EVALUACIÓN EXPERIMENTO 02

Este segundo experimento tiene como objetivo evaluar la capacidad de los modelos de lenguaje para **redactar una sección de “Resultados y Discusión”** en un contexto técnico-académico, a partir de un conjunto de datos experimentales previamente descritos.

Esta tarea implica habilidades diferentes a las del experimento anterior: ya no se trata solo de explicar un concepto, sino de **interpretar datos, generar discurso analítico y aplicar conocimiento técnico a partir de resultados numéricos**.

Como en el caso anterior, se han realizado dos pruebas:

- **Prueba 1:** evaluación con parámetros estándar (tiempo de respuesta + similitud literal y contextual con la respuesta oficial).
- **Prueba 2:** análisis de sensibilidad al parámetro de temperatura.

Se puede consultar una explicación detallada del experimento en Experimento 02: Redacción de informe de laboratorio.

6.2.1 PRUEBA 1 – PARÁMETROS ESTÁNDAR

Tiempo de respuesta

Modelo	Tiempo de respuestas
GPT-3.5-turbo	5.38
GPT-4	29.28
GPT-4o	12.12
o3-mini	10.3
Claude-3.5-sonnet	10.78
Claude-3-Haiku	6.81
Gemini-2.5-flash	16.74
Gemini-2.5-pro	38.66

Tabla 8. Resultados tiempos de respuesta experimento 02

Los tiempos de respuesta obtenidos se recogen en la Tabla 8. En general, los modelos presentan un incremento en el tiempo de generación respecto al experimento anterior, lo cual es coherente con la **mayor complejidad de la tarea**, que exige razonamiento estructurado y análisis de múltiples partes. GPT-4, por ejemplo, supera los 29 segundos, mientras que Gemini-2.5-pro alcanza 38.66 s.

Como es habitual, los modelos más ligeros como GPT-3.5-turbo (5.38 s) o Claude-3-Haiku (6.81 s) ofrecen tiempos significativamente más bajos, aunque en esta ocasión incluso

modelos intermedios como Claude-3.5-sonnet (10.78 s) o o3-mini (10.3 s) muestran eficiencia destacable.

Análisis con TF-IDF y embeddings

Modelo	Similitud del coseno TF-IDF	Similitud del coseno Embeddings
GPT-3.5-turbo	0.8775	0.6718
GPT-4	0.8799	0.7337
GPT-4o	0.9191	0.7894
o3-mini	0.7655	0.7489
Claude-3.5-sonnet	0.8822	0.7298
Claude-3-Haiku	0.9068	0.7758
Gemini-2.5-flash	0.8604	0.7536
Gemini-2.5-pro	0.9068	0.5858

Tabla 9. Resultados de evaluación TF-IDF y contextualizada experimento 02, prueba 1

En la métrica TF-IDF, los modelos **GPT-4o**, **Claude-3-Haiku** y **Gemini-2.5-pro** destacan con valores muy similares (0.9191, 0.9068 y 0.9068 respectivamente), lo que indica que todos ellos generan respuestas con una estructura similar a la oficial, utilizando terminología técnica adecuada y un estilo académico coherente. Sin embargo, esta métrica no capta con precisión la calidad del razonamiento.

Cuando se aplica un análisis más contextual mediante **embeddings**, el modelo **GPT-4o** se consolida como el más completo, con el valor más alto (0.7894). Esto refleja no solo su capacidad para estructurar correctamente el texto, sino también para **comprender el**

propósito de la sección, interpretar los datos y formular conclusiones técnicas bien **fundadas**. Le siguen Claude-3-Haiku (0.7758), Gemini-2.5-flash (0.7536) y o3-mini (0.7489), cuyo rendimiento es especialmente destacable dado su menor tamaño, lo que confirma que para tareas de razonamiento técnico moderado no siempre se requiere un modelo de gran escala.

En contraste, **Claude-3.5-sonnet**, pese a obtener una alta puntuación con TF-IDF, muestra una menor similitud contextual (0.7298), lo que sugiere que su respuesta, aunque correctamente redactada, carece de cohesión o profundidad en el análisis.

Estos resultados confirman que en tareas que implican **análisis** de datos y redacción técnica con razonamiento, como una sección de “Resultados y Discusión”, es fundamental que el modelo mantenga una coherencia semántica profunda y alineación con el propósito técnico. Modelos como GPT-4o y Claude-3-Haiku, afinados para generar contenido instructivo y argumentado, se adaptan mejor a este tipo de tareas. En cambio, modelos más complejos pero menos calibrados para el razonamiento estructurado, como Gemini-2.5-pro, no necesariamente ofrecen mejores resultados, especialmente cuando la tarea exige interpretación técnica más que redacción literal.

6.2.2 PRUEBA 2 – VARIACIÓN DE TEMPERATURA

Para esta prueba se han seleccionado los modelos **GPT-4o** y **Claude-3.5-sonnet**, por haber ofrecido un rendimiento destacable en la Prueba 1 en al menos una de las métricas consideradas. GPT-4o fue el modelo con mejor comportamiento general, combinando una alta similitud estructural y contextual, mientras que Claude-3.5-sonnet logró una de las puntuaciones más altas en la métrica TF-IDF. Además, estos dos modelos presentan estilos de generación distintos y pertenecen a arquitecturas diferentes, lo que permite explorar cómo la variación del parámetro de temperatura afecta a modelos con enfoques complementarios. El análisis se realiza variando la temperatura en dos niveles: 0.5 y 0.8.

Análisis con TF-IDF y embeddings

Modelo	Similitud del coseno	Similitud del coseno
	TF-IDF	Embeddings
GPT-4o-temp0.5	0.8775	0.8002
GPT-4o-temp0.8	0.8721	0.7641
Claude-3.5-sonnet-temp0.5	0.9191	0.6841
Claude-3.5-sonnet-temp0.8	0.7655	0.6295

Tabla 10. Resultados de evaluación TF-IDF y contextualizada experimento 02, prueba 2

La temperatura en los modelos de lenguaje regula el grado de aleatoriedad o creatividad en la generación del texto. Los resultados obtenidos confirman esta hipótesis. En el caso de **GPT-4o**, la reducción de temperatura a 0.5 mejora significativamente la métrica contextual (embeddings), alcanzando un valor de **0.8002** frente al **0.7641** obtenido con temperatura 0.8. Este incremento sugiere que, al reducir la creatividad del modelo, se mejora su alineación con el propósito técnico del texto, reforzando la claridad, cohesión y relevancia de los argumentos presentados. La métrica TF-IDF, por su parte, se mantiene prácticamente constante, lo que indica que la estructura general del texto apenas cambia, pero **su calidad interpretativa mejora notablemente**.

El comportamiento de **Claude-3.5-sonnet** es aún más sensible. Con temperatura 0.5, obtiene el valor más alto en TF-IDF (0.9191), pero una puntuación contextual moderada (0.6841). Sin embargo, al aumentar la temperatura a 0.8, su rendimiento cae drásticamente tanto en TF-IDF (0.7655) como en embeddings (0.6295). Este descenso indica que el modelo tiende a divergir del contenido técnico esperado cuando se le permite mayor libertad generativa, produciendo respuestas más creativas pero menos alineadas con los objetivos del experimento.

Estos resultados refuerzan la idea de que, en tareas técnicas que requieren interpretación objetiva de datos experimentales, es preferible utilizar temperaturas bajas para limitar la

aleatoriedad en la generación y asegurar respuestas estructuradas, precisas y coherentes. La sensibilidad a este parámetro varía entre modelos, pero en todos los casos se observa que el aumento de temperatura tiende a comprometer la calidad técnica del resultado.

6.3 *EVALUACIÓN EXPERIMENTO 03*

Este experimento evalúa la capacidad de los modelos para resolver un problema físico planteado en un contexto académico real, concretamente una pregunta de examen del Grado en Ingeniería de Telecomunicación. A diferencia de los dos experimentos anteriores, aquí se busca evaluar no solo la capacidad explicativa o discursiva, sino también la competencia de los modelos en tareas de razonamiento físico estructurado.

Se puede consultar más información de este experimento en **Experimento 03: Problema de física**.

6.3.1 PRUEBA 1 – PARÁMETROS ESTÁNDAR

Tiempo de respuesta

Modelo	Tiempo de respuestas
GPT-3.5-turbo	3.32
GPT-4	20.21
GPT-4o	12.2
o3-mini	4.47
Claude-3.5-sonnet	11.41
Claude-3-Haiku	4.94
Gemini-2.5-flash	83.98

ICAI	ICADE	CIHS
Gemini-2.5-pro	88.11	

Tabla 11. Resultados tiempos de respuesta experimento 03

Los resultados de tiempo de respuesta se mantienen en la línea de lo esperado: modelos ligeros como GPT-3.5-turbo, o3-mini o Claude-3-Haiku ofrecen respuestas rápidas (entre 3 y 5 segundos), mientras que los modelos más pesados como Gemini-2.5-flash y Gemini-2.5-pro presentan tiempos excesivos (más de 80 s), lo que puede comprometer su aplicabilidad práctica en entornos interactivos.

Análisis con TF-IDF y embeddings

Modelo	Similitud del coseno TF-IDF	Similitud del coseno Embeddings
GPT-3.5-turbo	0.6712	0.8254
GPT-4	0.7060	0.8554
GPT-4o	0.6462	0.7340
o3-mini	0.0000	0.0000
Claude-3.5-sonnet	0.7168	0.8513
Claude-3-Haiku	0.7398	0.8154
Gemini-2.5-flash	0.6528	0.7951
Gemini-2.5-pro	0.5786	0.6668

Tabla 12. Resultados de evaluación TF-IDF y contextualizada experimento 03, prueba 1

Desde el punto de vista cualitativo, lo más relevante es que **o3-mini falló en el experimento**, generando una respuesta vacía (0.0000 en ambas métricas), lo que sugiere que modelos más compactos pueden no ser capaces de manejar correctamente problemas más complejos.

En cuanto a la similitud con la respuesta oficial, destaca **Claude-3-Haiku** con el mayor valor en TF-IDF (0.7398), mientras que **GPT-4** y **Claude-3.5-sonnet** obtienen los valores más altos en embeddings (0.8554 y 0.8513 respectivamente). Estos resultados sugieren que estos dos modelos no solo identifican bien el enfoque físico correcto, sino que además lo desarrollan con una lógica alineada con la solución esperada.

Por el contrario, **GPT-4o** presenta un comportamiento más débil en esta tarea, especialmente en TF-IDF (0.6462), lo que puede explicarse por su tendencia a diversificar las explicaciones o adoptar formulaciones alternativas que, aunque válidas, se alejan del formato oficial del ejercicio.

6.3.2 PRUEBA 2 – VARIACIÓN DE TEMPERATURA

En este caso, se analizó la sensibilidad al parámetro de temperatura en dos de los modelos con mejor rendimiento en la prueba anterior: **GPT-4** y **Claude-3.5-sonnet**.

Análisis con TF-IDF y embeddings

Modelo	Similitud del coseno DTF-IF	Similitud del coseno Embeddings
GPT-4-temp0.1	0.7215	0.8177
GPT-4-temp0.8	0.6645	0.7177
Claude-3.5-sonnet-temp0.1	0.7018	0.8763
Claude-3.5-sonnet-temp0.8	0.7154	0.8667

Tabla 13. 1Resultados de evaluación TF-IDF y contextualizada experimento 03, prueba 2

En el caso de GPT-4, la temperatura baja (0.1) mejora tanto la estructura como la interpretación (0.7215 en TF-IDF y 0.8177 en embeddings), mientras que con temperatura 0.8 se pierde claridad y coherencia, reduciéndose la similitud contextual a 0.7177. Este

patrón refuerza la idea de que en tareas de resolución de problemas técnicos, una menor aleatoriedad mejora la calidad del razonamiento.

Curiosamente, **Claude-3.5-sonnet** muestra una estabilidad notable: con temperatura 0.1 alcanza el valor contextual más alto de todo el experimento (0.8763), pero incluso con temperatura 0.8 mantiene un nivel elevado (0.8667), lo que sugiere que este modelo está especialmente bien afinado para tareas instructivas de física, y que su estilo de generación más conservador le permite mantener coherencia incluso con parámetros más creativos.

6.4 EVALUACIÓN EXPERIMENTO 04

Este experimento evalúa la capacidad de un modelo de lenguaje para resolver un problema físico que incluye una imagen como parte de su enunciado: una espira circular apoyada entre el suelo y una pared, sometida a un campo magnético, en equilibrio. La resolución requiere interpretar información geométrica incluida en una imagen, aplicar principios de física (como el momento dipolar magnético) y desarrollar una respuesta técnica coherente y razonada.

Dado que el problema incluye una figura como parte fundamental del enunciado, se eligió el modelo GPT-4o, que era en el momento del experimento el único modelo disponible con capacidad de procesamiento multimodal mediante programación, y que además había demostrado en experimentos previos una alta calidad técnica y semántica en tareas similares.

Se puede consultar más información de este experimento **Experimento 04: Problema con input de imagen**.

6.4.1 PRUEBA 1 – PARÁMETROS ESTÁNDAR

Tiempo de respuesta

Modelo	Tiempo de respuestas
GPT-4o	18.15

Tabla 14. Resultados tiempos de respuesta experimento 04

El tiempo de 18.15 s es **ligeramente superior** al de los experimentos 02 y 03, lo cual tiene sentido, ya que en este caso el modelo no solo procesa texto, sino también una imagen de entrada, lo que añade una capa adicional de complejidad computacional.

Análisis con TF-IDF y embeddings

Modelo	Similitud del coseno DTF-IF	Similitud del coseno Embeddings
GPT-4o	7822	0.8341

Tabla 15. Resultados de evaluación TF-IDF y contextualizada experimento 04, prueba 1

La calidad de la respuesta es elevada, con un valor de **0.7822 en TF-IDF** y **0.8341 en embeddings**, lo que indica que la solución es tanto estructural como conceptualmente alineada con la respuesta oficial.

Este resultado sugiere que GPT-4o es capaz de interpretar correctamente diagramas **técnicos sencillos** y razonar sobre ellos, integrando la información visual con el contenido físico requerido para resolver el problema. Esta capacidad resulta especialmente interesante para tareas de ingeniería asistida por IA, donde los inputs visuales, como gráficos o imágenes, son muy habituales.

6.4.2 PRUEBA 2 – VARIACIÓN DE TEMPERATURA

Análisis con TF-IDF y embeddings

Modelo	Similitud del coseno DTF-IF	Similitud del coseno Embeddings
GPT-4o-temp0.1	0.5572	0.7435
GPT-4o-temp0.5	0.5246	0.7283

	ICAI	ICADE	CIHS
GPT-4o-temp0.7	0.4336		0.7136
GPT-4o-temp0.9	0.4111		0.7052

Tabla 16. Resultados de evaluación TF-IDF y contextualizada experimento 04, prueba 2

Aprovechando que en este experimento se trabaja únicamente con GPT-4o, se ha decidido evaluar un mayor número de configuraciones de temperatura (0.1, 0.5, 0.7 y 0.9) con el objetivo de analizar más a fondo su sensibilidad en tareas con componente visual y razonamiento técnico.

Los resultados muestran una **tendencia progresiva de degradación** en la calidad de la respuesta a medida que aumenta la temperatura. Con un valor bajo (0.1), el modelo mantiene una buena coherencia semántica (0.7435), cercana al valor estándar. Sin embargo, al incrementar la temperatura, se observa cómo el modelo comienza a desviarse del enfoque técnico esperado, introduciendo elementos innecesarios o perdiendo precisión. A temperatura 0.9, la similitud literal cae hasta **0.4111**, lo que confirma que en este tipo de tareas es fundamental mantener la creatividad del modelo bajo control para preservar la rigurosidad técnica.

6.5 EVALUACIÓN EXPERIMENTO 05

En este experimento se plantea una situación diferente a las anteriores: se simula un proceso ingenieril mediante la librería `simpy`, en el que se introduce de forma intencionada un fallo funcional. El objetivo es evaluar si los modelos son capaces de identificar el comportamiento anómalo a partir del output del sistema y explicar técnicamente qué está fallando y por qué.

Se puede ampliar información sobre el diseño del experimento en **Experimento 05: Evaluación de un proceso simulado**

Tiempo de respuesta

Modelo	Tiempo de respuestas
GPT-4	10.45
GPT-4o	15.13
Claude-3.5-sonnet	11.66
Gemini-2.5-flash	22.48
Gemini-2.5-pro	42.26

Tabla 17. Resultados tiempos de respuesta experimento 05

Los tiempos de respuesta varían entre los distintos modelos, desde 10.45 s (GPT-4) hasta 42.26 s (Gemini-2.5-pro), lo que refleja nuevamente una relación entre la complejidad del modelo y el tiempo de generación. GPT-4o se sitúa en un punto intermedio (15.13 s), al igual que Claude-3.5-sonnet (11.66 s). Gemini-2.5-flash requiere más de 22 s, lo que sugiere un mayor procesamiento para interpretar el output textual del sistema simulado.

Análisis con TF-IDF y embeddings

Modelo	Similitud del coseno DTF-IF	Similitud del coseno Embeddings
GPT-4	0.7405	0.8297
GPT-4o	0.7705	0.8059
Claude-3.5-sonnet	0.6809	0.7110
Gemini-2.5-flash	0.5816	0.7339
Gemini-2.5-pro	0.7772	0.8063

Tabla 18. Resultados de evaluación TF-IDF y contextualizada experimento 05, prueba 1

En cuanto a la calidad de las respuestas, los resultados de similitud obtenidos se presentan en la Tabla 18. En **TF-IDF**, el modelo con mejor puntuación es **Gemini-2.5-pro** (0.7772), seguido de GPT-4o (0.7705), lo que indica que ambos modelos generan respuestas estructuralmente cercanas a la oficial. Sin embargo, al observar la similitud semántica con embeddings, es GPT-4 quien lidera el análisis (0.8297), seguido de cerca por Gemini-2.5-pro (0.8063) y GPT-4o (0.8059). Esto sugiere que estos tres modelos comprenden correctamente la lógica interna del proceso y son capaces de razonar sobre su comportamiento.

Modelos como Claude-3.5-sonnet (0.7110) y Gemini-2.5-flash (0.7339) presentan resultados más discretos en embeddings, lo que indica que aunque pueden detectar parcialmente el fallo, sus explicaciones tienden a ser menos estructuradas desde el punto de vista técnico.

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

Para finalizar este trabajo, se presentan las conclusiones generales derivadas del desarrollo experimental, así como las líneas de trabajo futuras identificadas. Esta sección recoge de forma estructurada los objetivos alcanzados, los descubrimientos más relevantes, las posibilidades de aplicación práctica, y una reflexión ética.

En primer lugar, puede afirmarse que se han cumplido los objetivos planteados al inicio del proyecto. Se ha evaluado de forma detallada el rendimiento de distintos modelos de lenguaje preentrenados en tareas relacionadas con la ingeniería, tanto desde una perspectiva cualitativa como cuantitativa. A lo largo del trabajo se han analizado sus capacidades para explicar conceptos técnicos, redactar secciones de informes académicos, resolver problemas físicos y diagnosticar fallos en procesos simulados. Además, se ha desarrollado una metodología de evaluación replicable y se ha estudiado el impacto de parámetros como la temperatura sobre la calidad de las respuestas generadas.

Los resultados permiten identificar ciertos patrones de comportamiento y establecer comparaciones claras entre modelos. Los modelos de la familia GPT, en particular GPT-4 y GPT-4o, han mostrado un rendimiento especialmente sólido en tareas que requieren razonamiento técnico, estructuración del contenido y precisión semántica. Claude-3-Haiku, a pesar de ser un modelo más ligero, ha demostrado ser una opción eficiente y sorprendentemente competitiva, sobre todo en tareas explicativas. En cambio, los modelos Gemini han presentado un rendimiento más irregular, con resultados aceptables en tareas sencillas pero menos consistencia en escenarios que exigen mayor profundidad conceptual.

Cuando se analizan los resultados por tipo de tarea, se observa que GPT-4o y Claude-3-Haiku son especialmente adecuados para explicar conceptos técnicos de forma clara y accesible. GPT-4-turbo, con una temperatura de 0.5, ha sido el modelo más eficaz para la redacción de informes académicos, por su equilibrio entre claridad y rigor técnico. En problemas físicos complejos, los mejores resultados se han obtenido con GPT-4 y Claude-3.5-sonnet, que han demostrado una comprensión precisa de los principios involucrados. Por

otro lado, GPT-4o ha capaz de interpretar visualmente un esquema técnico, resolviendo un problema basado en imagen, y tanto GPT-4 como GPT-4o han logrado identificar con éxito un fallo funcional en un proceso simulado, lo que demuestra su aplicabilidad en entornos de ingeniería de mantenimiento. La comparación de los resultados obtenidos por modelos recientes con la generación producida por GPT-2 en el primer experimento ofrece una perspectiva clara de la evolución que ha experimentado el campo en pocos años. GPT-2, uno de los primeros modelos de lenguaje ampliamente accesibles, presenta respuestas más simples, genéricas y limitadas en términos de precisión técnica. Su rendimiento contrasta marcadamente con modelos como GPT-4o o Claude-3.5, que demuestran una comprensión profunda, estructuración compleja y un estilo de redacción más ajustado a los estándares técnicos y académicos. Esta diferencia no solo pone de relieve los avances en arquitectura y entrenamiento de modelos, sino también el potencial transformador de estas tecnologías cuando se aplican a tareas especializadas.

Desde una perspectiva ética, el uso de modelos de lenguaje en ingeniería plantea oportunidades y responsabilidades. Si bien su potencial para mejorar la eficiencia, la documentación y el aprendizaje técnico es evidente, también lo es la necesidad de mantener la supervisión humana y evitar una automatización acrítica de decisiones. No obstante, el despliegue de estas tecnologías debe realizarse con criterios éticos, garantizando la transparencia de los modelos utilizados y promoviendo el pensamiento crítico ante sus resultados.

A partir de los resultados obtenidos, se abren diversas líneas de trabajo futuras. Por ejemplo, se ha observado que los modelos de la familia GPT, especialmente GPT-4 y GPT-4o, ofrecen un rendimiento sobresaliente en tareas que requieren razonamiento técnico estructurado y diagnóstico de procesos, por lo que sería interesante diseñar experimentos adicionales que profundicen en este tipo de tareas, incorporando simulaciones más complejas o casos de uso reales en ingeniería industrial. Asimismo, dado que Claude-3-Haiku ha demostrado ser muy eficaz en la explicación de conceptos, se podría explorar su uso como asistente pedagógico, diseñando experimentos orientados a la docencia técnica.

Por otro lado, sería relevante ampliar la muestra de modelos evaluados, incorporando alternativas open-source como LLaMA 3, Mistral o Phi-3, así como futuras versiones de modelos cerrados, con el fin de entender cómo se posicionan frente a los modelos líderes actuales. se podrían diseñar tareas interdisciplinares o multimodales más exigentes, que combinen texto, imagen y código.

En conjunto, este trabajo demuestra que los modelos de lenguaje preentrenados pueden ser una herramienta de gran valor en el ámbito de la ingeniería, tanto para mejorar procesos como para apoyar la formación. No obstante, su uso debe estar siempre acompañado de una comprensión crítica de sus limitaciones y de una reflexión ética sobre su impacto.

Capítulo 8. BIBLIOGRAPHY

- [1] B. G. Buchanan y E. H. Shortliffe, Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project, Reading, MA: Addison-Wesley, 1984.
- [2] T. M. Mitchell, Machine Learning, New York: McGraw-Hill, 1997.
- [3] Y. LeCun, Y. Bengio y G. Hinton, «Deep learning,» *Nature*, vol. 521, p. 436–444, 2015.
- [4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser y I. Polosukhin, «Attention is All You Need,» de *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [5] S. Russell y P. Norvig, Artificial Intelligence: A Modern Approach, Hoboken, NJ: Pearson, 2021.
- [6] B. Goertzel, «Artificial general intelligence: Concept, state of the art, and future prospects,» *Journal of Artificial General Intelligence*, vol. 5, nº 1, p. 1–48, 2014.
- [7] N. Bostrom, Superintelligence: Paths, Dangers, Strategies, Oxford: Oxford University Press, 2014.
- [8] R. S. Sutton y A. G. Barto, Reinforcement Learning: An Introduction, Cambridge, MA: MIT Press, 2018, p. arXiv:2108.07258.
- [9] R. e. a. Bommasani, «On the opportunities and risks of foundation models,» *arXiv preprint*, p. arXiv:2108.07258, 2021.

-
- [10] D. Jurafsky y J. H. Martin, *Speech and Language Processing*, Standford: Stanford University (Draft 3rd Edition, online), 2025.
- [11] T. Mikolov, K. Chen, G. Corrado y J. Dean, «Efficient Estimation of Word Representations in Vector Space,» *arXiv preprint*, p. arXiv:1301.3781, 2013.
- [12] B. H. A. B. Rico Sennrich, «Neural Machine Translation of Rare Words with Subword Units,» *arXiv preprint*, 2016.
- [13] M.-W. C. K. L. K. T. Jacob Devlin, «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,» *arXiv preprint*, 2019.
- [14] T. Mikolov, K. Chen, G. Corrado y J. Dean, «Efficient Estimation of Word Representations in Vector Space,» *arXiv preprint*, p. arXiv:1301.3781, 2013.
- [15] J. Pennington, R. Socher y C. D. Manning, «GloVe: Global Vectors for Word Representation,» Doha, Qatar, 2014.
- [16] Weaviate, «Vector Embeddings Explained: What They Are and How They Work,» Weaviate, 14 June 2023. [En línea]. Available: <https://weaviate.io/blog/vector-embeddings-explained>. [Último acceso: 2025].
- [17] C. Olah, «Understanding LSTM Networks,» [En línea]. Available: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>.
- [18] Splunk, «5 Big Myths of AI and Machine Learning—Debunked,» Splunk Inc., 2023. [En línea]. Available: https://www.splunk.com/en_us/form/5-big-myths-of-ai-and-machine-learning-debunked.html.
- [19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser y I. Polosukhin, «Attention Is All You Need,» *arXiv preprint arXiv:1706.03762*, 2017.

-
- [20] J. Alammar, «: The Illustrated Transformer,» 2018. [En línea]. Available: <https://jalammar.github.io/illustrated-transformer/>.
- [21] N. S. A. R. K. L. S. N. M. M. Y. Z. W. L. P. J. L. Colin Raffel, «Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer,» *Journal of Machine Learning Research*, 2020.
- [22] S. Ruder, «NLP's ImageNet moment has arrived,» 2021. [En línea]. Available: <https://thegradient.pub/nlps-imagenet-moment-has-arrived/>.
- [23] Towards Data Science, «Visualising Word Embeddings in 3D,» 2020. [En línea]. Available: <https://towardsdatascience.com/visualising-word-embeddings-in-3d-6ef29f414c52>.
- OpenAI, “ChatGPT”, 2024. [Online]. Available: <https://chat.openai.com/>

ANEXO I

A continuación se recogen extractos relevantes del documento de problemas de Física de ICAI, utilizados para el diseño de los experimentos desarrollados en este trabajo.

MAGNETOSTÁTICA

¿Por qué la aguja de una brújula se alinea con el norte-sur? ¿Qué es un imán? ¿Cómo funciona el imán que está pegado en la puerta de la nevera? ¿Por qué el Sol invierte sus polos norte y sur cada once años? ¿Qué es la desviación magnética? ¿En qué se basa una resonancia magnética nuclear?

En este capítulo trataremos de contestar estas y otras preguntas sobre la base de lo que se conoce como "fenómeno magnético", que es algo que a la humanidad le ha parecido mágico durante cientos y cientos de años.

De lo primero que nos dimos cuenta es de que hay ciertos minerales que misteriosamente ejercen fuerzas entre sí pero sin tocarse, es decir, a distancia. Y, como ya hemos visto antes en clase, muchas veces hemos aplicado un fenómeno físico en nuestro beneficio sin entender aún cómo funciona. Ese es el caso de la brújula, un desarrollo técnico que usamos durante siglos sin entender su funcionamiento.

Aunque nos costó mucho comprender el campo magnético, ahora sabemos que está en todas partes, desde las partículas subatómicas hasta las estrellas, jugando un papel fundamental en el universo.

Como tenemos que aprender mucho en poco tiempo, no vamos a seguir el orden histórico sino que arrancaremos con ciertas definiciones básicas que nos introducirán en el mundo del campo magnético.

Vamos a ello:

DEFINICIONES

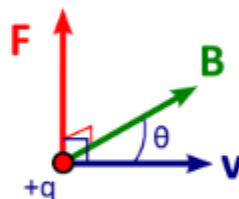
1. Llamaremos al campo magnético con la letra $\mathbf{B}$ y lo consideraremos un vector que siempre es parte de un campo vectorial. Esto te suena familiar porque acabamos de estudiar el campo eléctrico en el capítulo anterior, pero ya veremos que las propiedades de $\mathbf{B}$ son bastante diferentes a las de $\mathbf{E}$.
2. El campo $\mathbf{B}$ se manifiesta aplicando fuerzas solo sobre cargas en movimiento. Primera diferencia importante con el campo $\mathbf{E}$, que también se manifiesta sobre cargas que están quietas.
3. La fuerza que aplica $\mathbf{B}$ sobre una carga en movimiento se calcula como: $\mathbf{F} = q \cdot \mathbf{v} \times \mathbf{B}$, donde $\mathbf{F}$ es la fuerza en Newton (N), q es la carga eléctrica en Coulombs (C), $\mathbf{v}$ es la velocidad (m/s) y $\mathbf{B}$ es el valor del campo magnético en tesla (T).

Vamos a detenernos momentáneamente aquí con las definiciones para analizar sus consecuencias.

Por lo pronto podemos ver que la unidad de campo magnético se denomina tesla (T) y la podemos deducir de la expresión de la fuerza como $1 \text{ T} = \text{N} \cdot \text{s} / \text{C} \cdot \text{m}$.

Además, vemos que la interacción entre el campo $\mathbf{B}$ y la velocidad de la carga $\mathbf{v}$ es por medio de un producto vectorial. Esto tiene como consecuencia que la fuerza $\mathbf{F}$ sea, en general, perpendicular al plano

formado por los vectores $\mathbf{v}$ y $\mathbf{B}$. Esto implica que la descripción de $\mathbf{B}$ y sus acciones sean intrínsecamente 3D y nos agrega una dificultad extra al tratamiento matemático de los problemas y la tecnología que involucre a $\mathbf{B}$. También vemos que si una carga se mueve paralela al campo $\mathbf{B}$ entonces no se ve afectada y, por el contrario, si se mueve perpendicularmente a $\mathbf{B}$ el efecto es máximo como consecuencia del producto vectorial. Lo que sí comparten $\mathbf{B}$ y $\mathbf{E}$ es que el sentido de las fuerzas que producen se ve afectado por el signo de la carga eléctrica, de tal forma que si este signo cambia la fuerza se invierte.



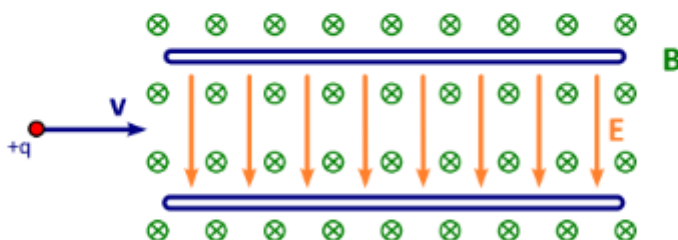
Es importante destacar que si combinamos la expresión de la fuerza que aplica un campo eléctrico sobre una carga eléctrica con la la que el campo magnético produce sobre una carga en movimiento obtendremos:

$$\mathbf{F} = q \cdot \mathbf{E} + q \cdot \mathbf{v} \times \mathbf{B} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B})$$

Expresión archiconocida como **Fuerza de Lorentz** que gobierna el movimiento de las partículas cargadas que se encuentran bajo la acción de campos electromagnéticos; como ocurre en los aceleradores de partículas, por ejemplo.

Ahora veamos un ejemplo de resolución un de problema en el que se combinan la acciones simultáneas de un campo magnético y uno eléctrico.

Ejemplo 1. La figura muestra dos planos paralelos infinitos entre los que existe una diferencia de potencial que produce un campo eléctrico ($\mathbf{E}$) perpendicular a los mismos como se muestra en la imagen. En la dirección perpendicular existe un campo magnético ($\mathbf{B}$). Determinar la velocidad con la que debe entrar una partícula con carga q para que atraviese el sistema sin desviarse.



RESOLUCIÓN:

Fíjate que el planteamiento del problema es en 3D y esto va a ser una constante a partir de ahora porque, como ya comentamos, el producto vectorial nos mete de lleno en el espacio tridimensional. Sin embargo, para simplificar, lo que siempre es deseable, algunos elementos los ubicamos perpendicularmente a la página de forma que los vectores de interés y las trayectorias nos queden sobre la misma. Esto lo haremos siempre que podamos porque nos ayudará a ver el problema más claramente.

Cuando la carga penetra en la región en la que coexisten los campos eléctrico y magnético, sufre el efecto de ambos. Para que la carga penetre en esta región sin desviarse, la fuerza resultante sobre la misma debe ser nula.

$$\mathbf{F} = \mathbf{F}_E + \mathbf{F}_B$$

$$\mathbf{F}_E = q \cdot \mathbf{E} \text{ (dirección vertical hacia abajo)}$$

$$\mathbf{F}_B = q \cdot \mathbf{v} \times \mathbf{B} \text{ (dirección vertical hacia arriba)}$$

Cuando ambas fuerzas sean iguales en módulo la fuerza resultante será nula. Por tanto, igualando los módulos de ambas expresiones:

$$qE = qvB \rightarrow v = \frac{E}{B}$$

¿Cambia este resultado si la carga cambia de signo?

Algo que te recomiendo que hagas, además, es que imagines que uno de los dos campos no existe para compensar al otro. Verás que es fácil demostrar que la trayectoria de la carga producida si solo existiese E es una parábola, mientras que si solo existe B está es un arco de circunferencia. Curioso, ¿no?

Este problema nos ha enseñado que hay que pensar con cuidado cómo hacer el dibujo del planteamiento para poder interpretar bien lo que está pasando en 3D, que los campos E y B pueden actuar al mismo tiempo y que las trayectorias que producen estos campos por separado son diferentes.

FUERZA MAGNÉTICA SOBRE CONDUCTORES

Habiendo discutido ya las características de las fuerzas que el campo magnético produce sobre cargas en movimiento, podemos seguir avanzando. Resulta que alguien llamado Ampère se dio cuenta de que los campos magnéticos también ejercen fuerzas sobre un conductor rectilíneo de longitud L por el que circula una corriente eléctrica I ; y encontró la siguiente expresión para calcularla:

$$\mathbf{F} = I \cdot \mathbf{L} \times \mathbf{B}$$

Si la espira tiene más de una vuelta, sobre cada una de ellas aparecerá un momento igual; y el momento total será la suma de todos ellos. Por eso, para una espira con n vueltas, el momento se puede escribir como:

$$\mathbf{M} = -nIBAsen(\theta)\hat{i}$$

El uso de espiras es tan común en ingeniería que, para simplificar, esta expresión suele escribirse del siguiente modo:

$$\mathbf{M} = \boldsymbol{\mu} \times \mathbf{B}$$

donde el vector $\boldsymbol{\mu}$ se conoce como **Momento dipolar magnético**, solo depende de las características constructivas de la espira y tiene dirección perpendicular a la superficie de la misma ($\hat{n}$). El sentido de $\hat{n}$ nos lo da la regla de la mano derecha. Con los dedos de tu mano derecha en el sentido de la corriente el pulgar indica el sentido de $\hat{n}$.

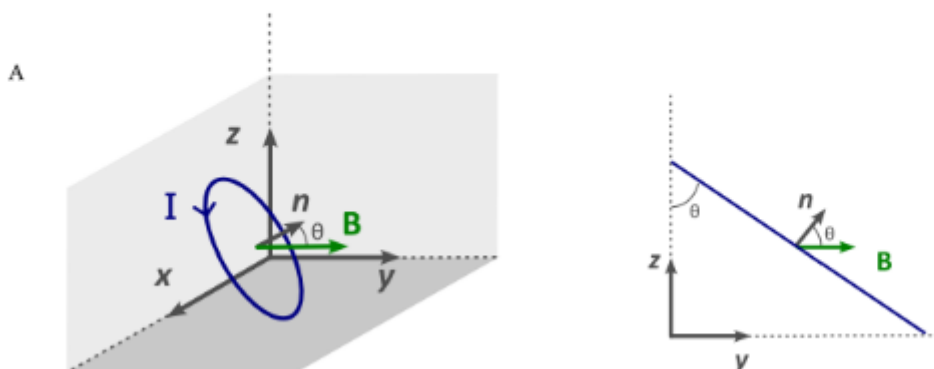
$$\boldsymbol{\mu} = nIA\hat{n}$$

Aunque no es objeto de este curso de física, no es muy difícil demostrar que esta expresión es válida independientemente de la forma que tenga la espira. Pongámosla en práctica resolviendo el siguiente problema.

Ejemplo 3. Una espira circular de radio R y masa m es recorrida por una intensidad constante I . La espira apoya en el suelo y en una pared formando con esta un ángulo θ . Un campo magnético B uniforme paralelo al suelo atraviesa dicha espira. Se pide calcular el valor del momento que sufre la espira y determinar el valor de B para que la espira se quede en su posición.

RESOLUCIÓN:

La imagen muestra una vista lateral de la espira. En ella se ha definido el vector normal ($\hat{n}$) a la superficie y se ha dibujado el campo B que la atraviesa.



Calculamos el momento dipolar magnético como:

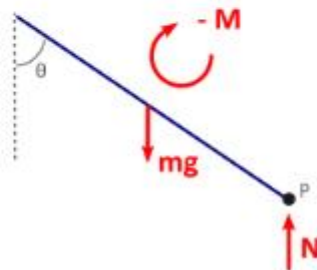
$$\mu = nIA\hat{n} = nI\pi R^2\hat{n}$$

Por tanto, el momento sobre la espira será:

$$\mathbf{M} = \mu \times \mathbf{B} = -nI\pi R^2 B \sin(\theta) \hat{i}$$

Para determinar el valor del campo magnético necesario para que la espira se quede en esa posición sin tener la pared como apoyo, dibujamos el diagrama de cuerpo libre de la espira y aplicamos la segunda ley de Newton para momentos. Para que la espira esté en equilibrio el sumatorio de momentos debe ser igual a cero:

$$\sum \mathbf{M} = 0$$



Sumamos momentos con respecto al punto en el que se la espira se apoya en el suelo:

$$mgR \sin(\theta) - I\pi R^2 B \sin(\theta) = 0 \rightarrow B = \frac{mg}{I\pi R}$$

LEY DE BIOT-SAVART

Hasta ahora hemos discutido los efectos del campo magnético, las fuerzas y momentos que es capaz de producir y las unidades en las que se mide. En este apartado vamos a entender cómo se producen los campos magnéticos; y la clave, tal y como lo describieron los científicos Jean-Baptiste Biot y Félix Savart en 1820, está en las cargas en movimiento.