MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

CREACIÓN DE UN CONJUNTO DE DATOS DE PREGUNTAS Y RESPUESTAS PARA EL AJUSTE FINO DE LLMS SOBRE DOCUMENTACIÓN PRIVADA

Autor
Mario Ripoll Domínguez

Director
Álvaro Jesús López López

Madrid

Julio de 2025

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

CREACIÓN DE UN CONJUNTO DE DATOS DE PREGUNTAS Y RESPUESTAS PARA EL AJUSTE FINO DE LLMS SOBRE DOCUMENTACIÓN PRIVADA en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2024-2025 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.: Mario Ripoll Domínguez Fecha: 15 / 07 / 2025

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Álvaro Jesús López López Fecha: 16 / 07 / 2025

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

CREACIÓN DE UN CONJUNTO DE DATOS DE PREGUNTAS Y RESPUESTAS PARA EL AJUSTE FINO DE LLMS SOBRE DOCUMENTACIÓN PRIVADA

Autor
Mario Ripoll Domínguez

Director
Álvaro Jesús López López

Madrid

Julio de 2025

CREACIÓN DE UN CONJUNTO DE DATOS DE PREGUNTAS Y RESPUESTAS PARA EL AJUSTE FINO DE LLMS SOBRE DOCUMENTACIÓN PRIVADA

Autor: **Ripoll Domínguez, Mario**

Director: López López, Álvaro Jesús

Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

1. Introducción

El rápido avance de la inteligencia artificial (IA) está siendo la principal cause de la evolución sustancial del procesamiento de lenguaje natural (PLN), en especial se ejemplifica este dato a través de la irrupción de los grandes modelos de lenguaje (LLM). Las herramientas que ya se emplean a diario más conocidas son GPT, LlaMa o Gemini, que destacan por su versatilidad para automatizar tareas rudimentarias y por su capacidad para comprender y sobre todo generar lenguaje natural en todos los registros[1][2]. No obstante, la aceptación e inclusión de estos nuevos agentes en entornos cerrados como pueden ser las empresas o instituciones no es tan directa, ya que surgen una serie de retos que deben ser abordados, desde la personalización de los modelos a los dominios privados en entornos locales hasta la protección de la información sensible frente a estructuras externas[3][4].

Este proyecto precisamente se enmarca en esa línea, con el principal objetivo de diseñar un proceso de trabajo semiautomatizado que permita la creación de conjuntos de datasetsets en formato de preguntas y respuestas (Q&A) sobre un corpus de documentación privada. La metodología que se describe en este proyecto abarca desde la segmentación de la información textual (*chunks*), pasando por la generación de los pares pregunta-respuesta realizando inferencia con el modelo Qwen2-7B-Instruct [5], hasta finalmente el proceso de validación cruzada implementado con técnicas de recuperación aumentada (RAG) con Mistral-7B[6]. De este modo, el flujo de trabajo propuesto minimiza las alucinaciones hasta eliminarlas y asegura que la información del corpus se cubre en su totalidad, representando de forma veraz la información.

Todo este proceso se caracteriza por estar diseñado para ser implementado en entornos locales, es decir, no solo sin acceso a la nube para preservar la confidencialidad de la información, sino que además con recursos computacionales limitados que son

precisamente el reactivo limitante a la hora de ejecutar tareas de PLN. Esta restricción ha sido abordada mediante técnicas de optimización de los modelos, como la cuantización [7][8] sin restringir el rendimiento de los mismos.

2. Metodología

El proceso diseñado se caracteriza por su naturaleza modular y secuencial, desarrollado precisamente para garantizar la reproducibilidad y escalabilidad de este. El proceso comienza con la definición cerrada de un corpus documental que debe cumplir con una serie de requisitos: actualidad de la información, coherencia estructural y sobre todo ausencia de los datos durante el entrenamiento de los modelos. Como el objetivo final es destinar el proceso a entornos corporativos y no se tiene acceso a un corpus de tal índole, se recurre a documentos de corte académico extraídos de arXiv con fecha posterior a la de salida de los modelos implicados, de este modo se asegura la inexistencia de *data leakeges* [9][5]. El primer reto del proceso es la conversión de los documentos a información textual segmentada y procesable, mientras se mantienen unidades semánticas relevantes y con suficiente coherencia contextual.

Una vez segmentado el corpus en *chunks*, se procede a generar el dataset de preguntas y respuestas mediante un proceso de inferencia con el modelo seleccionado tras un exhaustivo proceso de selección en el que se prioriza la eficiencia computacional y el rendimiento. Finalmente se optó por el Qwen2 frente a otros modelos evaluados como Mistral 7B, LlaMa2 y GPT-J, tras ser evaluados en métricas de precisión técnica, generación de *tokens*, eficiencia computacional (FLOPs) y calidad de la salida. Para realizar la inferencia se emplea la librería por antonomasia dentro del campo del PLN, la librería *transformers* de Hugging Face y a su vez técnicas de cuantización hasta representaciones de 4 bit en los pesos de los parámetros, reduciéndose así el consumo de GPU sin comprometer significativamente la salida [7][10]. Todo ello se consigue mediante un proceso iterativo de refinamiento del prompt (*prompt engineering)* [11][12] en el cual se contextualiza la información de entrada y se define la estructura de la salida.

Posteriormente y a pesar de las precauciones tomadas durante la elaboración del *prompt*, se realiza un post-procesado de la salida obtenida en formato JSON, ya que algunos de los pares pregunta-respuesta a pesar de ser correctos tanto técnica como gramaticalmente, carecen de valor para el posterior ajuste fino del modelo, que al fin y al cabo es la utilidad final que se le va a dar al dataset confeccionado.

Finalmente, se debe comprobar que el proceso genera un conjunto representativo del corpus. Es decir, se debe asegurar que el dataset realiza una cobertura completa de la información para evitar lagunas de conocimiento, y además también se debe garantizar que

las respuestas que proporciona el modelo son única y exclusivamente basadas en la información del corpus y no son producto de alucinaciones o información que no aparece en la documentación. Para ello se integra un sistema de validación cruzada basado en RAG, que mediante búsqueda semántica permite contrastar la información generada con los fragmentos originales. Este método se consigue a través de un índice FAISS construido sobre el modelo de embeddings BAAI/bge-large-en-v1.5, seleccionado siguiendo las directrices del proyecto y las métricas del MTEB benchmark [13][14].

El modelo Mistral 7B se mantiene ajeno al proceso de generación y únicamente es utilizado en la fase de validación para garantizar que cada respuesta se corresponde con el contexto recuperado. Además, para permitir la trazabilidad de los fallos, el propio modelo proporciona una justificación fundamentada explicando el origen del error.

3. Resultados

Como producto del proceso diseñado, el trabajo ha permitido generar un dataset de prueba que se compone de más de 40.000 pares pregunta-respuesta, obtenidos de un corpus técnico de 57 documentos en total y segmentados en 10.845 fragmentos textuales. El proceso de generación mediante inferencia con el Qwen2 y posterior validación da lugar a un archivo JSON que cumple con los objetivos planteados y asegura consistencia, trazabilidad y calidad suficiente para ser utilizado en un posterior proceso de fine-tuning supervisado (SFT) [15][5].

Los resultados han sido evaluados cuantitativamente después de la aplicación de la validación, en particular la precisión de los pares generados y la cobertura completa del corpus:

Métrica	Valor
Precisión	96,73%
Chunks accedidos	97,24%

Métricas de validación del proceso

Estos resultados indican una elevada concordancia de la información generada con la información de entrada al presentar una precisión del 96,73%, y por otro lado también aseguran que la cobertura de la información es completa, al acceder al 97,24% de los chunks [16][17].

Se debe destacar que las preguntas descartadas tanto en la limpieza del JSON como en el proceso de validación no son fruto de resultados incorrectos o de alucinaciones, sino que en su gran mayoría se deben a incoherencias semánticas entre los dos modelos utilizados y su incapacidad para recuperar con exactitud absoluta las palabras claves empleadas para

verificar la veracidad de la respuesta [11][18]. Asimismo, también se ha observado que los documentos que incluyen mayor cantidad de información en formato no textual dan lugar a una cantidad notablemente inferior de preguntas útiles debido a las constantes referencias a información no incluida en el proceso.

Finalmente, durante el proceso de selección del modelo de generación también se obtuvieron resultados destacable a través de benchmarks adaptados al caso de uso. Se demuestra que el Qwen 2 no sólo genera salidas de mayor precisión técnica, sino que también lo hace con una eficiencia computacional sustancialmente menor que sus homólogos evaluados [19].

4. Conclusiones

En conjunto, el sistema propuesto no solo es replicable y escalable, sino que también responde a las necesidades reales de instituciones que desean incorporar inteligencia artificial sin exponer su información. El proceso diseñado ha demostrado una competencia total para el caso de uso que se aplica, generando de forma semiautomatizada una generación vez y completa de un dataset sobre documentación de corte técnico y respetando las restricciones de computación para ser reproducible de forma local.

Además, de cumplir con los objetivos técnicos, su modularidad e integración facilitan su adaptación futura a entornos corporativos reales. Aun así, este proyecto sumado a la innegable evolución de los grandes modelos de lenguaje pone en evidencia la necesidad de continuar investigando líneas futuras que añadan valor y competencias al proceso. Los modelos multimodales y la búsqueda de la trazabilidad absoluta de la información son hoy en día el camino hacia los agentes de inteligencia artificial, que deben ser el producto final que persiga esta línea de investigación.

5. Referencias

[1] Kang, Y., Cai, Z., Tan, C. W., Huang, Q., & Liu, H. (2020). Natural language processing (NLP) in management research: A literature review. *Journal of Management Analytics*, *7*(2), 139–172. https://doi.org/10.1080/23270012.2020.1756939

[2] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving Language Understanding by Generative Pre-Training,” OpenAI, 2018. [Online]. Available: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

[3] M. R. J, K. VM, H. Warrier, and Y. Gupta, "Fine-tuning LLM for Enterprise: Practical Guidelines and Recommendations," *arXiv preprint* arXiv:2404.10779, 2024.

[4] Angels Balaguer, Vinamra Benara, Renato Luiz de Freitas Cunha, Roberto de M. Estevão Filho, Todd Hendry, Daniel Holstein, Jennifer Marsman, Nick Mecklenburg, Sara

Malvar, Leonardo O. Nunes, Rafael Padilha, Morris Sharp, Bruno Silva, Swati Sharma, Vijay Aski, and Ranveer Chandra. *Rag vs fine-tuning: Pipelines, tradeoffs, and a case study on agriculture*, 2024.

[5] A. Yang et al., "Qwen2 Technical Report," arXiv preprint arXiv:2407.10671, 2024.

[6] C. Jeong, "Domain-specialized LLM: Financial fine-tuning and utilization method using Mistral 7B," Journal of Intelligence and Information Systems, vol. 30, no. 1, pp. 93–120, Mar. 2024.

[7] Z. Han, C. Gao, J. Liu, J. Zhang, and S. Q. Zhang, "Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey," arXiv preprint arXiv:2403.14608, 2024.

[8] E. Hua, B. Qi, K. Zhang, Y. Yu, N. Ding, X. Lv, K. Tian, and B. Zhou, "Intuitive FineTuning: Towards Simplifying Alignment into a Single Process," arXiv preprint arXiv:2405.11870, 2024.

[9] Z. Gekhman, G. Yona, R. Aharoni, M. Eyal, A. Feder, R. Reichart, and J. Herzig, "Does Fine-Tuning LLMs on New Knowledge Encourage Hallucinations?" arXiv, 2024.

[10] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing," arXiv preprint arXiv:1808.06226, 2018. [Online]. Available: https://arXiv.org/abs/1808.06226.

[11] F. Liu, Y. Liu, L. Shi, H. Huang, R. Wang, Z. Yang, L. Zhang, Z. Li, and Y. Ma, "Exploring and Evaluating Hallucinations in LLM-Powered Code Generation," arXiv preprint arXiv:2404.00971, 2024.

[12] S. Vatsal and H. Dubey, "A survey of prompt engineering methods in large language models for different NLP tasks," arXiv preprint arXiv:2407.12994, 2024. [Online]. Available: https://arXiv.org/abs/2407.12994

[13] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, "MTEB: Massive Text Embedding Benchmark," arXiv preprint arXiv:2210.07316, 2023. [Online]. Available: https://arXiv.org/abs/2210.07316

[14] J. Amodei, "Inference: The next step in GPU-accelerated deep learning," NVIDIA Developer Blog, Aug. 2016. [Online]. Available: https://developer.nvidia.com/blog/inference-next-step-gpu-accelerated-deep-learning/

[15] J. Ye, Y. Yang, Q. Zhang, T. Gui, X. Huang, P. Wang, Z. Shi, and J. Fan, "Empirical Insights on Fine-Tuning Large Language Models for Question-Answering," arXiv preprint arXiv:2409.15825, 2024.

[16] Mistral AI, "Mixtral of Experts: A High-Quality Sparse Mixture-of-Experts Model," Mistral AI, Dec. 2023. [Online]. Available: https://mistral.ai/en/news/mixtral-ofexperts

[17] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, "MTEB: Massive Text Embedding Benchmark," arXiv preprint arXiv:2210.07316, 2023. [Online]. Available: https://arXiv.org/abs/2210.07316

[18] Z. Gekhman, G. Yona, R. Aharoni, M. Eyal, A. Feder, R. Reichart, and J. Herzig, "Does Fine-Tuning LLMs on New Knowledge Encourage Hallucinations?" arXiv, 2024.

[19] J. Hoffmann et al., "Training Compute-Optimal Large Language Models," arXiv preprint arXiv:2203.15556, Mar. 2022. [Online]. Available: https://arXiv.org/abs/2203.15556

CREACIÓN DE UN CONJUNTO DE DATOS DE PREGUNTAS Y RESPUESTAS PARA EL AJUSTE FINO DE LLMS SOBRE DOCUMENTACIÓN PRIVADA

Abstract

1. Introduction

The rapid advancement of artificial intelligence (AI) is the main driver behind the substantial evolution of natural language processing (NLP), particularly exemplified by the emergence of large language models (LLMs). The most well-known tools already used daily are GPT, LLaMA, and Gemini, which stand out for their versatility in automating rudimentary tasks and for their ability to understand and, above all, generate natural language across all registers [1][2]. However, the acceptance and integration of these new agents in closed environments, such as companies or institutions, is not so straightforward, as several challenges must be addressed — from the customization of models to private domains in local environments, to the protection of sensitive information from external infrastructures [3][4].

This project is precisely framed within that line of work, with the main objective of designing a semi-automated workflow that enables the creation of datasets in a question-and-answer (Q&A) format based on a corpus of private documentation. The methodology described in this project spans from the segmentation of textual information into chunks, through the generation of Q&A pairs via inference using the Qwen2-7B-Instruct model [5], to the final stage of cross-validation implemented through retrieval-augmented generation (RAG) techniques using Mistral-7B [6]. In this way, the proposed workflow minimizes hallucinations to the point of elimination and ensures that the corpus is fully covered, faithfully representing the information.

This entire process is characterized by its design for implementation in local environments — that is, not only without cloud access to preserve data confidentiality, but also with limited computational resources, which are precisely the limiting factor when executing NLP tasks. This constraint has been addressed through model optimization techniques such as quantization [7][8], without compromising their performance.

2. Methodology

The designed process is characterized by its modular and sequential nature, developed specifically to ensure its reproducibility and scalability. It begins with the strict definition

of a documentary corpus that must meet a series of requirements: recency of information, structural coherence, and above all, absence from the data used during the training of the models. Since the ultimate goal is to adapt this process for corporate environments, and no such private corpus is available, the project relies on academic documents extracted from arXiv with publication dates posterior to the release of the models involved. This ensures the absence of data leakages [9][5]. The first challenge of the process is the conversion of these documents into segmented, processable textual information, while preserving relevant semantic units with sufficient contextual coherence.

Once the corpus has been segmented into chunks, the question-and-answer dataset is generated via inference using the model selected after an exhaustive selection process that prioritized computational efficiency and performance. Qwen2 was ultimately chosen over other models such as Mistral 7B, LLaMA2, and GPT-J, based on technical precision, token generation, computational efficiency (FLOPs), and output quality. The inference is carried out using the reference library in the NLP field — Hugging Face's *transformers* — along with quantization techniques that reduce parameter weight representations to 4-bit, thereby minimizing GPU consumption without significantly compromising output quality [7][10]. This is achieved through an iterative prompt refinement process (*prompt engineering*) [11][12], in which the input is contextualized, and the output structure is clearly defined.

Subsequently, and despite the precautions taken during prompt design, a post-processing step is applied to the generated output in JSON format. This is necessary because some of the question-answer pairs, although technically and grammatically correct, lack value for the model's later fine-tuning — which is ultimately the intended purpose of the constructed dataset.

Finally, it is essential to verify that the process produces a representative dataset of the original corpus. That is, the dataset must ensure full coverage of the source information to avoid knowledge gaps, and it must also guarantee that the responses provided by the model are based solely on the corpus content, not on hallucinations or external knowledge. To this end, a cross-validation system based on Retrieval-Augmented Generation (RAG) is integrated, which uses semantic search to compare the generated content with the original document fragments. This method is implemented through a FAISS index built on top of the BAAI/bge-large-en-v1.5 embedding model, selected according to the project's criteria and the MTEB benchmark metrics [13][14].

The Mistral 7B model remains completely independent from the generation process and is used exclusively during the validation phase to ensure that each answer aligns with the retrieved context. Additionally, in order to enable traceability of errors, the model itself provides a reasoned explanation clarifying the origin of any discrepancy.

3. Results

As a result of the designed process, this work has enabled the generation of a test dataset composed of over 40,000 question-answer pairs, obtained from a technical corpus of 57 documents and segmented into 10,845 textual fragments. The generation process, carried out via inference using the Qwen2 model and followed by validation, yields a JSON file that meets the stated objectives and ensures consistency, traceability, and sufficient quality for use in a subsequent supervised fine-tuning (SFT) process [15][5].

The results have been quantitatively evaluated following the validation phase, particularly focusing on the accuracy of the generated pairs and the overall coverage of the corpus:

Metrics	Values
Accuracy	96,73%
Access chunks	97,24%

Métricas de validación del proceso

These results indicate a high degree of alignment between the generated information and the input data, with an achieved accuracy of 96.73%. Furthermore, they also confirm that information coverage is nearly complete, with 97.24% of the chunks being accessed during the process [16][17].

It is important to highlight that the questions discarded during both the JSON cleaning and validation stages were not the result of incorrect outputs or hallucinations. In the vast majority of cases, they stemmed from semantic inconsistencies between the two models used and their inability to recover with absolute precision the specific keywords required to verify the truthfulness of the responses [11][18]. Additionally, it was observed that documents containing a higher proportion of non-textual content led to a significantly lower number of usable questions, due to frequent references to information excluded from the processing pipeline.

Finally, during the model selection phase, notable results were also obtained through benchmarks tailored to the project's use case. It was demonstrated that Qwen2 not only produces outputs with higher technical precision but also does so with substantially lower computational cost compared to the other evaluated models [19].

4. Conclusion

Overall, the proposed system is not only replicable and scalable, but also addresses the real needs of institutions seeking to integrate artificial intelligence without compromising their

data. The designed process has demonstrated full competence for the applied use case, enabling the semi-automated and comprehensive generation of a dataset based on technical documentation, while respecting computational constraints to ensure reproducibility in local environments.

In addition to meeting the technical objectives, its modularity and integration facilitate future adaptation to real corporate settings. Nevertheless, this project — coupled with the undeniable evolution of large language models — highlights the need to continue exploring future research directions that add value and capabilities to the process. Multimodal models and the pursuit of complete traceability of information are, today, the pathway toward artificial intelligence agents, which should represent the ultimate outcome pursued by this line of research.

5. References

[1] Kang, Y., Cai, Z., Tan, C. W., Huang, Q., & Liu, H. (2020). Natural language processing (NLP) in management research: A literature review. *Journal of Management Analytics*, *7*(2), 139–172. https://doi.org/10.1080/23270012.2020.1756939

[2] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving Language Understanding by Generative Pre-Training," OpenAI, 2018. [Online]. Available: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

[3] M. R. J, K. VM, H. Warrier, and Y. Gupta, "Fine-tuning LLM for Enterprise: Practical Guidelines and Recommendations," *arXiv preprint* arXiv:2404.10779, 2024.

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

[5] A. Yang et al., "Qwen2 Technical Report," arXiv preprint arXiv:2407.10671, 2024.

[6] C. Jeong, "Domain-specialized LLM: Financial fine-tuning and utilization method using Mistral 7B," Journal of Intelligence and Information Systems, vol. 30, no. 1, pp. 93–120, Mar. 2024.

[7] Z. Han, C. Gao, J. Liu, J. Zhang, and S. Q. Zhang, "Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey," arXiv preprint arXiv:2403.14608, 2024.

[8] E. Hua, B. Qi, K. Zhang, Y. Yu, N. Ding, X. Lv, K. Tian, and B. Zhou, "Intuitive FineTuning: Towards Simplifying Alignment into a Single Process," arXiv preprint arXiv:2405.11870, 2024.

[9] Z. Gekhman, G. Yona, R. Aharoni, M. Eyal, A. Feder, R. Reichart, and J. Herzig, "Does Fine-Tuning LLMs on New Knowledge Encourage Hallucinations?" arXiv, 2024.

[10] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing," arXiv preprint arXiv:1808.06226, 2018. [Online]. Available: https://arXiv.org/abs/1808.06226.

[11] F. Liu, Y. Liu, L. Shi, H. Huang, R. Wang, Z. Yang, L. Zhang, Z. Li, and Y. Ma, "Exploring and Evaluating Hallucinations in LLM-Powered Code Generation," arXiv preprint arXiv:2404.00971, 2024.

[12] S. Vatsal and H. Dubey, "A survey of prompt engineering methods in large language models for different NLP tasks," arXiv preprint arXiv:2407.12994, 2024. [Online]. Available: https://arXiv.org/abs/2407.12994

[13] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, "MTEB: Massive Text Embedding Benchmark," arXiv preprint arXiv:2210.07316, 2023. [Online]. Available: https://arXiv.org/abs/2210.07316

[14] J. Amodei, "Inference: The next step in GPU-accelerated deep learning," NVIDIA Developer Blog, Aug. 2016. [Online]. Available: https://developer.nvidia.com/blog/inference-next-step-gpu-accelerated-deep-learning/

[15] J. Ye, Y. Yang, Q. Zhang, T. Gui, X. Huang, P. Wang, Z. Shi, and J. Fan, "Empirical Insights on Fine-Tuning Large Language Models for Question-Answering," arXiv preprint arXiv:2409.15825, 2024.

[16] Mistral AI, "Mixtral of Experts: A High-Quality Sparse Mixture-of-Experts Model," Mistral AI, Dec. 2023. [Online]. Available: https://mistral.ai/en/news/mixtral-ofexperts

[17] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, "MTEB: Massive Text Embedding Benchmark," arXiv preprint arXiv:2210.07316, 2023. [Online]. Available: https://arXiv.org/abs/2210.07316

[18] Z. Gekhman, G. Yona, R. Aharoni, M. Eyal, A. Feder, R. Reichart, and J. Herzig, "Does Fine-Tuning LLMs on New Knowledge Encourage Hallucinations?" arXiv, 2024.

[19] J. Hoffmann et al., "Training Compute-Optimal Large Language Models," arXiv preprint arXiv:2203.15556, Mar. 2022. [Online]. Available: https://arXiv.org/abs/2203.15556

Página intencionadamente en blanco

Tabla de contenido

Índice de Figuras

Índice de Tablas

1. Introducción

El avance de la Inteligencia Artificial (IA) en los últimos años ha abierto un mundo nuevo de posibilidades, entre las cuales destaca el procesamiento del lenguaje (NLP) a través de los grandes modelos de lenguaje (LLM). Estos modelos, como Gemini, LlaMa o GPT han demostrado tener una gran capacidad para comprender y generar información en lenguaje natural[15] . Uno de los ejemplos más claros es el de *ChatGPT*, la herramienta desarrollada por *OpenAI* que permite interactuar con un asistente virtual soportado por las versiones de GPT, en constante evolución. Sin embargo, la introducción de los LLM conlleva una serie de desafíos entre los que destaca el manejo seguro de la información, ya que en el ámbito corporativo hace tiempo que ya es imprescindible contar con estas herramientas a la vez que se mantiene la privacidad de la información.

Para que un LLM sea capaz de generar respuestas precisas, contextualizadas y completas, basadas en documentación privada y específica de una corporación, es necesario que dicho LLM sea entrenado por un conjunto de datos (documentos, en el contexto de este proyecto en particular) que representen fielmente la información sobre la que se precisan respuestas. La creación de este conjunto de datos no es trivial e implica un proceso complejo de recopilación, selección y estructuración de preguntas y respuestas derivadas de un *corpus* de documentos privados de una organización. Todo ello debe realizarse con la atención puesta en respetar la sensibilidad de la información y evitar *data leakages.*

Por ello, este proyecto tiene como principal objetivo desarrollar un flujo de trabajo semi automatizado a través del cual se permita la creación de un conjunto de datos compuesto por preguntas y respuestas (Q&A) a partir de un *corpus* de documentación privada. Este flujo de trabajo busca sistematizar el proceso de generación de preguntas y sus respectivas respuestas garantizando que la información contenida en los documentos es exhaustivamente cubierta en su totalidad. El diseño e implementación de este sistema no solo facilitará la adaptación de LLMs a contextos específicos y privados, sino que también incentivará la integración de asistentes virtuales en entornos privados sin que la seguridad e integridad de la información se vea comprometida.

Además, una característica diferencial de este proyecto es que el conjunto de datos generado será utilizado posteriormente para realizar un ajuste fino (*fine-tuning)* de un LLM, todo ello englobado en el marco de los entornos locales. Por lo tanto, debe tenerse presente en todo momento que este proceso semi automatizado persigue lograr un flujo de trabajo que sea realizable y replicable en un entorno local, con las limitaciones de recursos que ello implica. Por un lado, al restringir la conexión a servidores externos se garantiza la confidencialidad de los datos, pero, por otro lado, esta limitación dificulta la labor de garantizar la cobertura total

de la información y la calidad del conjunto de datos al estar los recursos limitados por la máquina utilizada.

La creación de este conjunto de datos no solo no es trivial, sino que además es única. Por ello muchas empresas optan por emplear un sistema RAG que no necesita de LLMs con ajuste fino, pero sin embargo están limitados por la calidad de las bases de datos empleadas [1] y sus capacidades de recuperación en lugar de las capacidades de los propios LLMs.

A pesar de que a priori los sistemas RAG y los ajustes finos de LLMs se consideraban dos soluciones alternativas [68] **,** se ha visto que nada más lejos de la realidad, son dos propuestas totalmente complementarias. El binomio que forman el fine-tuning y RAG se debe a que el ajuste fino de LLMs presenta una gran limitación: un proceso de entrenamiento costoso. Mientras que por otro lado RAG permite actualizaciones dinámicas del conocimiento del modelo sin necesidad de entrenamiento ya que el ajuste fino de un LLM se puede beneficiar de la. Por ello cada vez más se emplea el ajuste fino como punto de partida del modelo, dotándolo de estilo y caracterizando su comportamiento, mientras que RAG se emplea para nutrir al modelo de conocimiento novedoso no aprendido durante la fase de entrenamiento.

1.1. Motivación

Con la llegada de los asistentes virtuales impulsados por los LLMs, los entornos corporativos se enfrentan a dos principales desafíos: la gestión eficiente de su documentación interna y la extracción rápida y precisa de información relevante.

Estos retos se ven intensificados sobre todo en sectores altamente especializados, donde la información técnica, la normativa y la rápida evolución del sector, requieren de un proceso ágil en lo que a gestión de documentos se refiere. Sin embargo, las restricciones de seguridad y ética relacionadas con el uso de plataformas en servidores externos en la nube limitan enormemente su aplicación directa en estos contextos, además de que su naturaleza generalista ofrece un rendimiento inferior a lo demandado por parte de estos sectores. Por ello la necesidad de desarrollar soluciones adaptadas a las necesidades particulares de cada empresa se ha vuelto crítica hoy en día para poder procesar de forma segura y eficiente ese tipo de información. De este concepto se derivan otro dos claves:

I. Personalización y precisión: Al emplear el conjunto de datos formado por preguntas y respuestas para realizar el posterior *fine-tuning* del LLM, el proceso tendrá la capacidad de generar respuestas más precisas y menos generalistas, mejorándose así la eficiencia en el acceso a la información.

II. Seguridad y privacidad: Tanto la elaboración del conjunto de datos, como la ejecución del propio asistente en un entorno local sin necesidad de depender de plataformas en la nube, es particularmente relevante en el contexto de información sensible. La gestión del corpus de documentos es de gran importancia en el contexto del proyecto ya que el riesgo de exposición de la información pasa por ejecutar la lectura y clasificación de esta en entornos seguros. A pesar de que el proyecto se desarrolla en Google Colab, es únicamente para la parte exploratoria, ya que la implementación puede luego darse con una GPU en un entorno local (idealmente no en un ordenador portátil convencional)

1.2. Alcance y Objetivos

Para validar el proyecto se seguirá una metodología ampliamente utilizada en el mundo del ajusto fino de LLMs y en la creación de datasets sintéticos, que consiste en comparar mediante Benchmarks los modelos que se emplearán a lo largo del proyecto y también se llevará a cabo un proceso de evaluación para verificar que se cumplen los objetivos. A pesar de que la técnica más expandida de verificación de dataset reside en la comparación pura, como A. Cvetanović y P. Tadić en [32] se centran en modelos de entre 110 y 270 millones de parámetros para validar el desempeño de su dataset en serbio basado en el dataset más utilizado en modelos de IA, SQuAD de la universidad de Stanford [33] . En esta caso no se puede recurrir a esa técnica ya que no se tiene el corpus necesario, y el objetivo es desarrollar un proceso que pueda generar datasets y no que el dataset per se.

Para conseguir alcanzar con éxito la meta presentada anteriormente, el proyecto primero debe cumplir con los siguientes objetivos:

I. Definir y obtener un *corpus* relevante de documentos

Se debe definir un corpus y la fuente de donde se va a extraer el mismo. Para la selección del corpus de documentos se pretende ejecutar un *web scrapping* con el fin de captar documentos sobre una misma temática. Para ello deberá decidirse si se tienen en consideración todos los formatos (incluyendo código) o únicamente aquellos archivos de texto que cumplan con los requisitos necesarios para realizar la división de información en *chunks*.

Como se deriva de [20], al obtener el *corpus* de información, también se debe valorar C(onditioned)-RLFT, es decir, dar pesos diferentes a la información según la calidad de la fuente de la que provengan. A pesar de ser una optimización más eficiente, también es de mayor complejidad, por lo que se deberá sopesar la viabilidad de gestionar diferentes pesos según la fuente (si se obtienen documentos de bases de datos varias y no únicamente de arXiv).

II. Validar el conjunto de datos generado por medio de Benchmarks

Una vez obtenido el conjunto de datos y realizado el fine-tuning, este debe ser comparado frente a otros conjuntos y bajo las mismas métricas. Un buen punto de partida son los ejemplos de [18] UltraChat, desarrollado para el ajuste fino de modelos en abierto en su capacidad de responder a comandos de instrucciones, y [19] UltraFeedback, que se ha llevado a cabo con el objetivo de mitigar la presencia de sesgos en el aprendizaje por refuerzo humano y reforzar la idea de que los conjuntos de datos son más eficientes y objetivos con *feedback* de IA en lugar de *feedback* humano.[19]

III. Compartimentar la información del corpus en *chunks*

Para que el modelo sea capaz de captar toda la información, esta debe ser previamente dividida en *chunks*. Para ello se debe tener en consideración el uso de memoria que implica el proceso, ya que como se deriva de [12] , la aplicación de técnicas automática de obtención de *chunks* (como AutoChunk, un sistema de compilación automático) puede reducir hasta un 80% de la memoria de activación sin perder más del 10% de velocidad en el proceso. Es importante iterar la división de *chunks* por medio de varias técnicas, ya sean basadas en semántica o eficiencia de memoria.

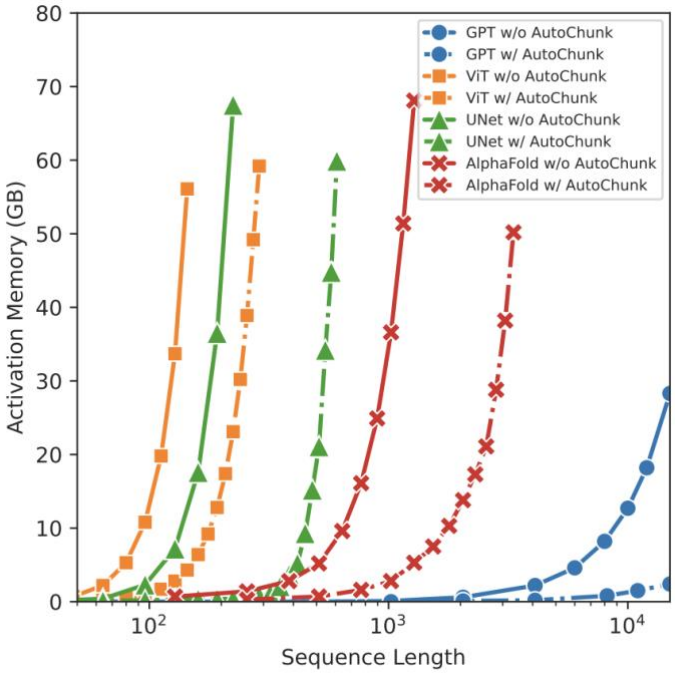

Figura 1. Comparación de la memoria de activación al emplear AutoChunk. Fuente: [12]

Finalmente se debe corroborar que el dataset obtenido realiza una cobertura completa de la información de entrada, ya que de no ser así el proceso no puede considerarse como válido, pues la característica principal que se persigue es la de generar un dataset completo. Para ello,

se deberá demostrar que porcentaje de la información ha sido utilizado durante la generación del dataset.

IV. Comparar el desempeño de los diferentes modelos

Como se ha comentado con anterioridad, el Q&A será generado en un archivo tipo json realizando inferencia con un LLM. Para ello se deberá comparar el comportamiento de varios modelos en diferentes configuraciones que impliquen variaciones de la cuantización, la temperatura e incluso las versiones de estos. Además, también deberá ser evaluado el rendimiento de los modelos en función del corpus de documentos seleccionado y a su vez deben ser comparados con los resultados obtenidos en otros estudios, como por ejemplo con [5] , [7] [8] .

Dichas comparativas se podría realizar, por ejemplo, por medio de *benchmarks* establecidos o por la comparativa directa de algunas métricas como: F1-Score, precisión (*accuracy*), sensibilidad (*Recall*), BLEU (*Bilingual Evaluation Understudy*)[11] , ROUGE, etc. Aunque estas métricas carecen de sentido si no se tuviese un dataset de referencia contra el que compararlas.

2. Descripción de las Tecnologías

Este apartado pretende contextualizar los modelos y técnicas que se emplearán a lo largo de proyecto. Debe tenerse en cuenta que el estado del arte del procesamiento de lenguaje natural (PLN) avanza a una velocidad inusualmente rápida con la democratización de las herramientas de inteligencia artificial. Por ello, a pesar de que el proyecto se inició en mayo de 2024, hay muchas técnicas y modelos que se mencionan que pueden resultar subóptimos para el desarrollo de este caso de uso.

El ejemplo más claro se encuentra en el proceso de validación del proyecto (véase el apartado 4.4.6 del documento), ya que los modelos mencionados están demostrando tener rendimientos inferiores a modelos más modernos y ligeros, especializados en tareas de extracción de información.

No obstante, los modelos y técnicas que se presentan a continuación son perfectamente integrables dentro del proyecto y siguen siendo ampliamente utilizados por la comunidad de Hugging Face ya que su rendimiento permite establecer un benchmark para los modelos venideros y también facilitan la integración en códigos de mayor complejidad debido a la modularidad de estos.

2.1. Modelos utilizados

Durante el desarrollo del sistema se evaluaron varios modelos de lenguaje con arquitecturas diversas, escalas distintas y configuraciones técnicas específicas. La elección de estos modelos responde tanto a criterios teóricos (como su capacidad de generalización, tamaño del contexto o arquitectura interna) como a restricciones prácticas impuestas por el entorno de ejecución (capacidad de la GPU, eficiencia en inferencia, y compatibilidad con herramientas como Hugging Face y Langchain). La Tabla 1 muestra las características principales de los modelos con los que se inició la investigación para la generación de pares pregunta-respuesta:

Modelo	Tokens	Parámetros	MoE
Mixtral 8x7B	32768	~56 Billones	Sí - 2 activas
Qwen2	32768	~7 Billones	No
Mistral 7B	32768	~7 Billones	No
GPT-J	2048	~6 Billones	No
Llama2	4096	~7 Billones	No

Tabla 1. Especificaciones técnicas de los modelos empleados

Como se puede apreciar en la Tabla 1, los modelos permiten una ventana contextual de 32,768 tokens. Esta cantidad de tokens permite procesar largos corpus de documentación de forma relativamente ágil. La excepción de Llama2 se debe a que s una versión destinada a ejecutar procesos en inferencia de forma más rápida, pero con menor capacidad de atención al contexto global. Por otro lado, se aprecia también como los parámetros de los modelos se toman en torno a 7 Billones, que es la cantidad mínima estándar en largos modelos de lenguaje.

Existen también otros tamaños estandarizados, como 13B o 70B, pero las primeras pruebas exploratorias determinaron que son tamaños excesivamente complejos y difíciles de manejar para este tipo de tareas. Cabe destacar que Mixtral 8x7B realmente emplea únicamente 16 billones de parámetros por ejecución, ya que como se explica en el apartado 4.3.5 de este documento, el Mixture of Experts en este caso solo activa 2 instancias de 7 billones cada una y en ningún caso activa las 8 instancias que suman un total de 56 billones de parámetros.

3. Estado de la Cuestión

Para contextualizar el proyecto, se deben tener presentes los avances realizados en relación con el caso de estudio que, en este caso en particular, son especialmente importantes debido a la naturaleza del proyecto y la rápida evolución que están teniendo los LLMs tanto en el mundo académico como corporativo.

La generación de conjuntos de datos de preguntas y respuestas (Q&A) ha cobrado relevancia en el desarrollo y ajuste fino de LLMs, especialmente en entornos donde es necesario adaptar los modelos a contextos específicos y sectores industriales. Este apartado revisa los enfoques y tecnologías actuales en este ámbito, centrándose en la creación de datos de Q&A, las técnicas de ajuste fino, los sistemas de protección de la información en el uso de datos sensibles y la cantidad de datos necesaria para poder realizar un ajuste fino lo suficientemente bueno.

3.1. Conceptos clave y definiciones previas

Los conceptos y definiciones que se explican en esta sección son de gran relevancia para la comprensión y contextualización del proyecto en su totalidad, ya que, si bien es verdad, no se tratan técnicamente a todos los niveles, están implícitos en todos los pasos de selección, comparación y validación de los modelos y datasets. Para poder trabajar de forma fundamentada es imprescindible tener presente estas características que se presentan.

3.1.1. Tokens

En el contexto del procesamiento de lenguaje natural (NLP), como bien definen Thawani et al. [44] los tokens representan la unidad básica de representación de texto y no están limitados a una frase, palabra o sílaba ya que son característicos de cada tokenización. Existen varios enfoques a la hora de realizar la tokenización de texto, puede ser basándose en palabras, subpalabras, caracteres e incluso por tokenización semántica o contextual.

Por ejemplo, en 2018 Google publicó *SentencePiece* [36] de forma abierta, un tokenizador de sub-palabras independiente del idioma que fue uno de los componentes clave de la creación de modelos multilingües al habilitar una tokenización sin necesidad de preprocesamiento en ningún idioma en específico.

Debido al rápido crecimiento de los grandes modelos de lenguaje, para que la comunidad pueda usar los modelos libremente, estos se suelen cargar con un tokenizador específico de cada modelo preentrenado. Por ejemplo, modelos más antiguos como pueden ser BERT[21] y *DestilBERT* [37] emplean *WordPiece* con un vocabulario de 30,000 tokens [38] , sin embargo,

modelos más nuevos como GPT-2, GPT-3, Qwen2 o Mixtral8x7B emplean un tokenizador basado en *Byte Pair Encoding* (BPE)[39] .

BPE entró en el mundo de los LLMs en 2016 debido a la necesidad la traducción automática neuronal mediante la descomposición de las palabras en unidades más manejables y frecuentes[40] , pero realmente debe sus raíces a Philip Gage quien en 1994 presentó BPE [41] como un algoritmo competitivo rente a otros más tradicionales en términos de eficiencia y simplicidad, tal y como se deriva de la Figura 2. Precisamente estas características son las que han permitido que el algoritmo gane popularidad en la tokenización para LLMs ya que:

- Permite la representación de palabras desconocidas como combinación de subpalabras frecuentes
- Reduce la longitud de las entradas, mejorando así la eficiencia computacional
- Es compatible con la arquitectura Transformer, que requiere entradas tokenizadas en secuencias discretas [11] , es decir, en vectores de números enteros que representan unidades lingüísticas permitiendo al modelo operar sobre datos estructurados de forma finita y ordenada
- Soportado en ecosistemas NLP como Hugging Face, lo que ha facilitado su estandarización

	12 bit LZW	14 bit LZW	Default BPE	Small BPE	Fast BPE
Original file size (bytes)	544,789	544,789	544,789	544,789	544,789
Compressed file size (bytes)	299,118	292,588	276,955	293,520	295,729
Compression time (secs)	28	28	55	41	30
Expansion time (secs)	27	25	20	21	19
Compression data size (bytes)	25,100	90,200	17,800	4,400	17,800
Expansion data size (bytes)	20,000	72,200	550	550	550

Figura 2. Evaluación comparativa del rendimiento de BPE frente a LZW en términos de compresión, velocidad y eficiencia. Fuente: [41]

3.1.2. Embeddings

Los Embeddings son representaciones vectoriales densas de múltiples dimensiones de elementos o unidades lingüísticas [43] , generalmente estos embeddings están formados por tokens dentro de una enorme matriz de vocabulario [44] . A diferencia de otras representaciones, como por ejemplo *One-hot* encoding, los embeddings permiten capturar relaciones semánticas y sintácticas entre grupos de elementos lingüísticos.

A pesar de ser un representación vectorial no deben confundirse con otro tipo de técnicas de su misma naturaleza. Por un lado, y siguiendo con el ejemplo anterior, One-hot encoding permite

segmentar en múltiples variables numéricas otra de origen categórico. Pero esta representación vectorial no va más allá del análisis de datos y no es capaz de capturar ningún tipo de relación semántica entre sus valores. Por otro lado, los embeddings reflejan relaciones representando unidades lingüísticas similares en posiciones cercanas o alcanzables fácilmente con operaciones algebraicas simples en el espacio n-vectorial en el que se representen.

Es precisamente es esta capacidad de caracterizar semántica de forma distribuida y organizada lo que permite a los embeddings realizar tareas avanzadas como la generación de texto, clasificación contextual y recuperación de información.

La evolución del Procesamiento de Lenguaje Natural se ha visto fuertemente influenciada por los hitos que han ido marcando las nuevas técnicas y modelos de embeddings:

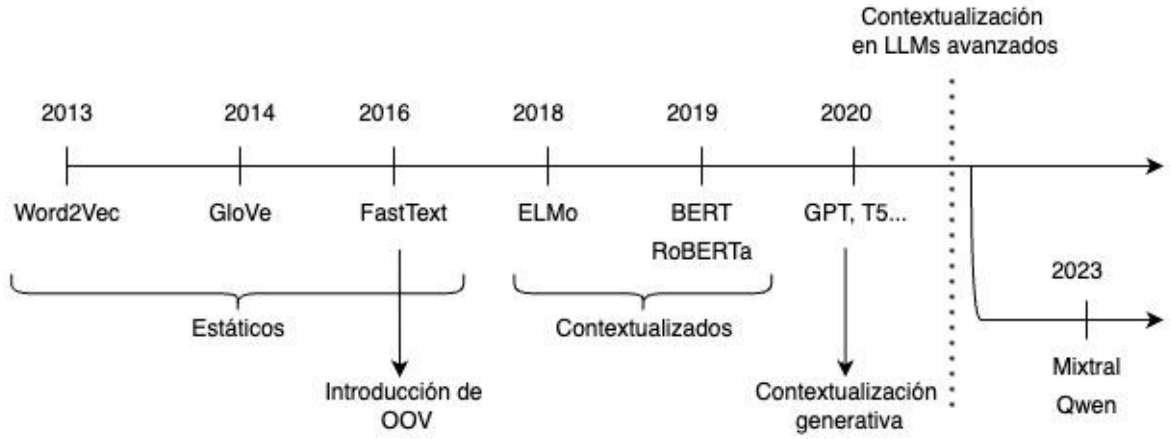

Figura 3. Evolución de los Embeddings en PLN

Siguiendo la línea temporal de la Figura 3, partiendo desde el año 2013, Word2Vec fue el primer método en aprender a representar palabras en función de su contexto. Mikolov et al. [45] propusieron una arquitectura basada en otras dos:

- *Continuos Bag of Words* (CBOW); predice la palabra correspondiente basándose en el contexto.
- *Skip-gram;* predice el contexto o las palabras de alrededor basándose en la palabra actual.

Sin embargo, estas representaciones vectoriales son estáticas, es decir, que cada palabra tiene siempre la misma representación independientemente del contexto en el que se encuentre. El

hecho de que sean representaciones vectoriales estáticas da lugar a dos limitaciones respecto a los embeddings contextuales:

- Falta de entendimiento en el contexto: por ejemplo, la palabra "banco" tendrá siempre el mismo vector, ya sea que se refiera a un "banco para sentarse" o a un "banco como institución financiera". Esto genera ambigüedad semántica. Si se comparan frases como "me senté en el banco del parque" y "deposité dinero en el banco", los modelos basados en embeddings estáticos no distinguirán el significado contextual de "banco", ya que no adaptan la representación a la oración. Esta limitación impide una comprensión precisa del lenguaje en tareas como desambiguación semántica o traducción automática.
- *Out-of-Vocabulary:* aquellas palabras que no hayan sido vistas durante el entrenamiento del modelo no tendrán representación vectorial en el espacio de embeddings y por lo tanto serán desconocidas. Este factor esto implica que el entrenamiento debe ser extenso para poder abarcar todo el lenguaje. Aun así, resulta prácticamente imposible captar todas las palabras, sobre todo en contextos dinámicos con nombres propios, neologismos o términos técnicos. Esta carencia se acentúa más en modelos multilingües ya que el entrenamiento implica un corpus inmenso y cuidadosamente equilibrado.

En 2014 GloVe, introdujo las ventanas de contexto locales [46] y en 2016 FastText [47] el manejo de palabras desconocidas, *Out-of-vocabulary* (OOV), que permitiría representar palabras no vistas durante el entrenamiento como suma de otras subpalabras o tokens más frecuentes. A pesar de estos avances, no fue hasta la aparición de los embeddings contextualizados en 2018, que se pasó de representaciones estáticas a embeddings contextualizados, es decir, una palabra puede tener múltiples representaciones vectoriales en función de su contexto. El primer modelo en implementar esta evolución fue ELMo, utilizando redes LSTM bidireccionales (*Bidirecctional Language Model*, BiLM) [48], con un modelo hacia delante que predice la siguiente palabra dado el contexto previo y otro modelo hacia atrás que predice la palabra anterior dado el contexto posterior. Poco después surgiría BERT que introduciría el modelo de Transformer bidireccional [21] .

Esta transición a embeddings contextualizados representa el inicio de modelos de lenguaje más avanzados capaces de adaptarse dinámicamente a diferentes contextos, dominios y tareas,

dando lugar en a la llegada disruptiva de *Generative Pretrained Transformer* más conocido como GPT [22] y *Text-to-Text Transfer Transformer* también conocido como T5 [23] .

- GPT se basa en la arquitectura tradicional de Transformer [11] , concretamente en la parte de decoder, donde cada token única y exclusivamente puede atender a anteriores de la secuencia
- T5 en lugar de basarse solo en la parte decoder de la arquitectura Transformer, está construido en full Transformer, es decir, en encoder-decoder explicados en la sección 3.1.4 del documento.

Además de continuar con las representaciones vectoriales contextualizadas también incluyen capacidades de generación secuencial de texto para la resolución de múltiples tareas de NLP, como bien puede ser ChatGPT.

La etapa evolutiva más reciente los embeddings viene marcada por los grandes modelos escalables y especializados como Mixtral [49] y *Quantum Well for Enhanced NLP* - Qwen [24] , quienes mantienen e incluso superan el rendimiento de arquitecturas monolíticas (todos los parámetros del modelo activos durante lo pasos sucesivos de inferencia) más grandes, pero manteniendo un coste muy inferior. Estos modelos no solo obtienen representaciones vectoriales contextualizadas como modelos anteriores, si no que construyen embeddings con más características:

- Técnicas como *Top-k routing* permite que para cada token de entrada solo se activen lo *k* expertos con mayor puntuación [50] , en el caso de Mixtral que emplea una arquitectura *Mixture-of-Experts* solo utiliza *top-2 routing* activándose así 2 de los 8 expertos disponibles en Mixtral 8x7B. Cuando *k* es menor que el número de expertos involucrado también recibe el nombre de *Sparse mixture of Experts (SMoE)*.
- Ajustados al dominio, permitiendo ser utilizados en entornos limitados a través de técnicas como la cuantización, que será mencionada más en profundidad en la sección 3.1.4 del documento.
- Adaptados a la tarea que desarrollan, tal y como explican Bommasani et al. en [51] , los embeddings de los LLMs más complejos ya no solo capturan proximidad semántica como los fundacionales anteriores (por ejemplo, Word2Vec o GloVe), sino que son capaces de modelar relaciones conceptuales profundas evitando ligar su rendimiento a la tarea que desempeñan.

Finalmente, además de la diferenciación entre embeddings contextualizados y estáticos que se ha hecho con anterioridad, se debe destacar que existen múltiples formas de clasificar los embeddings. Como se ha visto en este apartado, son muchas las características que

pueden tener los modelos y las técnicas empleadas, por ello, basándonos en [42] podemos encontrar la siguiente clasificación según el enfoque, arquitectura e inferencia de los modelos para generar embeddings:

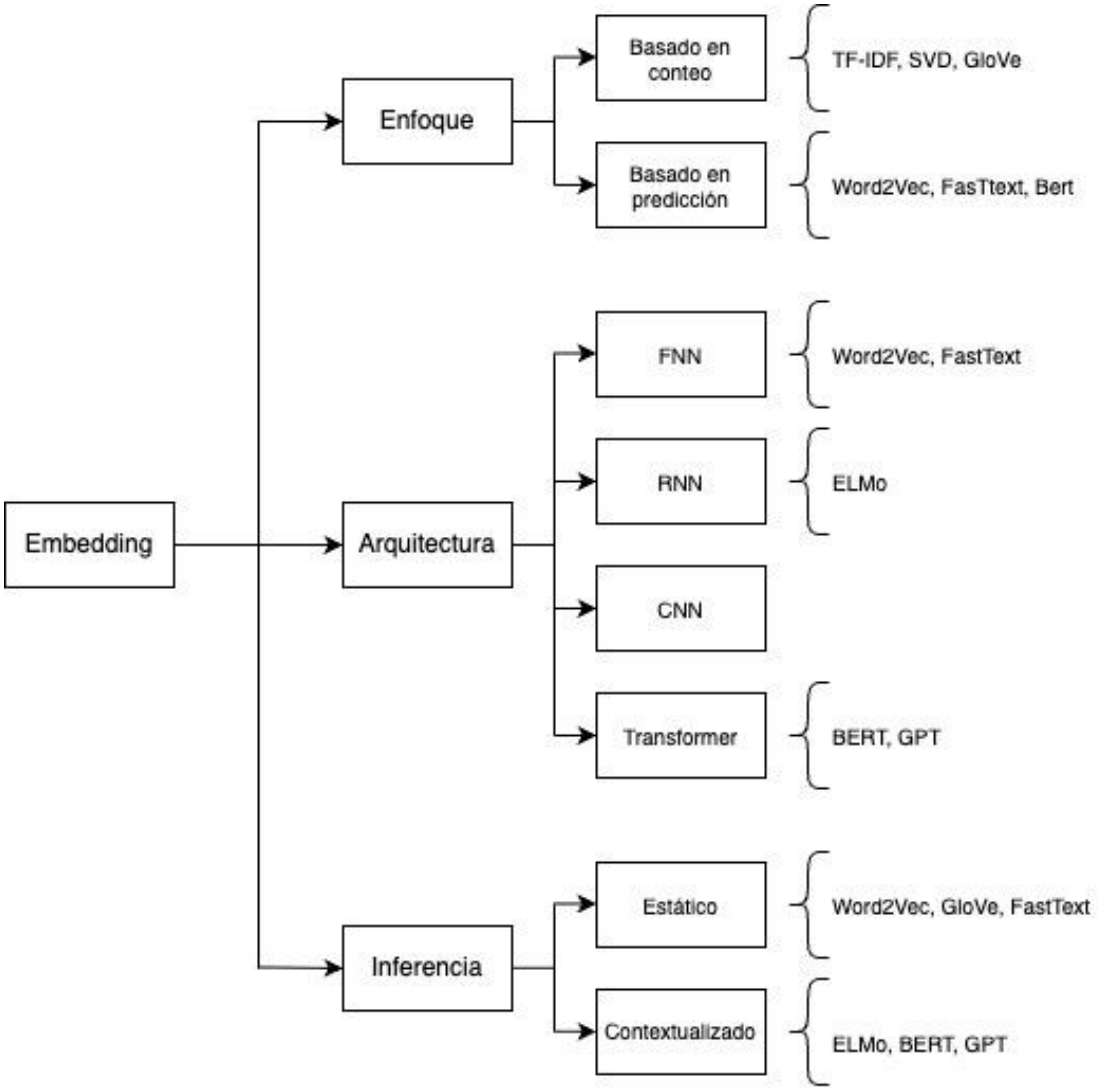

Figura 4. Dimensiones de los embeddings

3.1.3. Mecanismo de atención

El mecanismo de atención (presente en la arquitectura de Transformer) define que no es estrictamente necesario recorrer secuencialmente la entrada, si no que se pueden establecer varios niveles de atención dentro de una misma entrada a tokens que sean más relevantes en cada paso de la generación [26]. En los casos de traducciones, por ejemplo, mejora el desempeño del modelo y facilita la captura de relaciones a corto plazo.

Los mecanismos de atención varían según la función final que debe desempeñar un modelo, por ello se pueden clasificar siguiendo 4 criterios, tal y como se procede en [27] :

Mecanismos de atención	*Suavidad del mecanismo de atención*	*Suave / dura* *Global / local*
	Formas de las características de entrada	*Por elemento* *Por ubicación*
	Representaciones de entrada	*Distintiva* *Autoatención* *Co-atención* *Jerárquica*
	Representaciones de salida	*Salida única* *Múltiples cabezas* *Multidimensional*

Tabla 2. Clasificación de los mecanismos de atención

El mecanismo de atención que se menciona a lo largo del proyecto consiste principalmente de tres matrices: Query (Q), Key (K) y Value (V). Estos valores se calculan a partir de las entradas de la secuencia y son los que establecen la importancia relativa que se le da a cada token dentro de la secuencia.

La atención se computa como $Q \cdot K^T$, cuyos valores se conocen como *logits*. Estos valores se escalan y posteriormente se suavizan con la función de activación Softmax dando lugar a una distribución de atención probabilística.

Finalmente, la distribución de atención es utilizada en combinación con la matriz de valores V para producir la representación que incluye la información contextual que permite al modelo centrarse en las partes más relevantes. De esta forma, cada token puede introducir conocimiento que se encuentra en otros tramos de la secuencia, consiguiendo capturar relaciones complejas y la estructura semántica global de la secuencia de entrada.

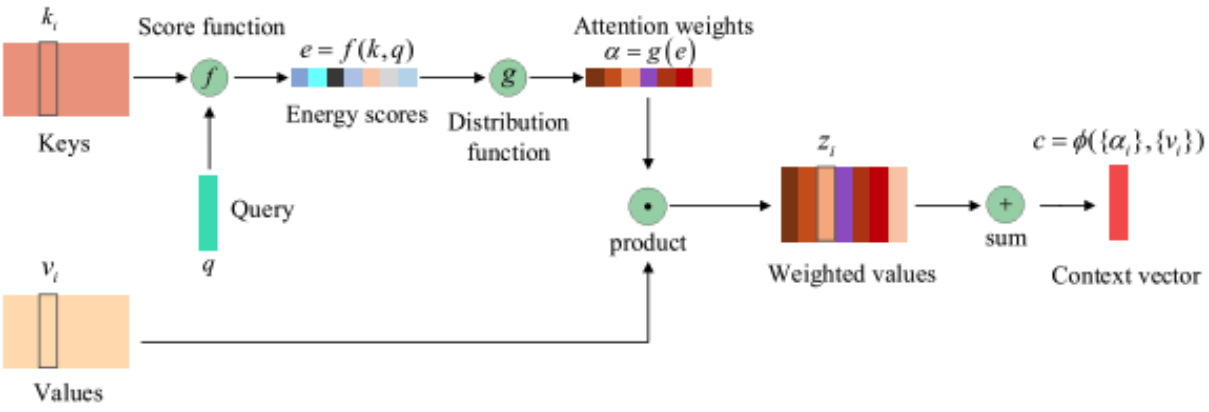

Figura 5. Diagrama de funcionamiento del mecanismo de atención. Fuente:[27]

De todos modos, el límite de elección de un token puede modularse cambiando el valor de la temperatura (variable que puede tomar valores entre [0,1]), de forma que una temperatura de 0 dará un resultado determinista y a medida que incrementa su valor, el resultado de la salida del modelo de lenguaje pierde el determinismo y da lugar a salidas más creativas.

A pesar de que esto puede ser un factor favorable en muchas aplicaciones (como escritura asistida o generación de texto sin referencia), no interesa alejarse del resultado determinista en muchas otras, como puede ser en la creación de datasets sintéticos controlados donde se persigue la rigurosidad respecto al corpus de documentos base para asegurar la consistencia semántica.

3.1.4. Arquitectura de Transformers

La arquitectura de Transfomer fue introducida al mundo académico en 2017 a través de uno de los artículos más influyentes en lo que a LLMs se refiere [11] , en este trabajo titulado "*Attention isa ll you need*", Vaswani et al. Revolucionan el campo de NLP y establecen los cimientos de los primeros LLM que seguirían evolucionando basados en esta misma arquitectura [25] como BERT [21] , GPT [22] , T5 [23] o Qwen2 [24] .

La arquitectura Transformer se diferencia esencialmente de arquitecturas de deep learning previas como RNN, LSTM, GRU Y CNN en su capacidad para procesar secuencias de datos en paralelo mediante mecanismos de atención. De esta forma se elimina la necesidad de recurrencia (como las RNN) o convolución (como las CNN) para modelar dependencias entre elementos de una secuencia, es decir, la capacidad de establecer relaciones entre secciones de la entrada [26]. De esta forma se permite analizar todas las dependencias, por ejemplo, una palabra con el resto sin necesidad de recorrer todos los tokens de manera secuencial.

A diferencia de las redes neuronales previamente mencionadas (dependientes de la secuencialidad), los Transformer utilizan mecanismos de atención auto-regresiva y de múltiples cabezas de atención, asignando así dinámicamente la importancia de diferentes partes de las entradas. De esta forma se pierda la limitación de la paralelización que venía impuesta por arquitecturas previas y se pierda la dificultad de capturar relaciones a largo plazo, un problema que a pesar de que ya fue abordado las LSTM, no deja de limitar ese tipo de redes neuronales en cuanto a paralelización.

A continuación, se presenta la arquitectura del modelo Transformer, formada por el encoder y decoder principalmente, de los que derivan otros componentes.

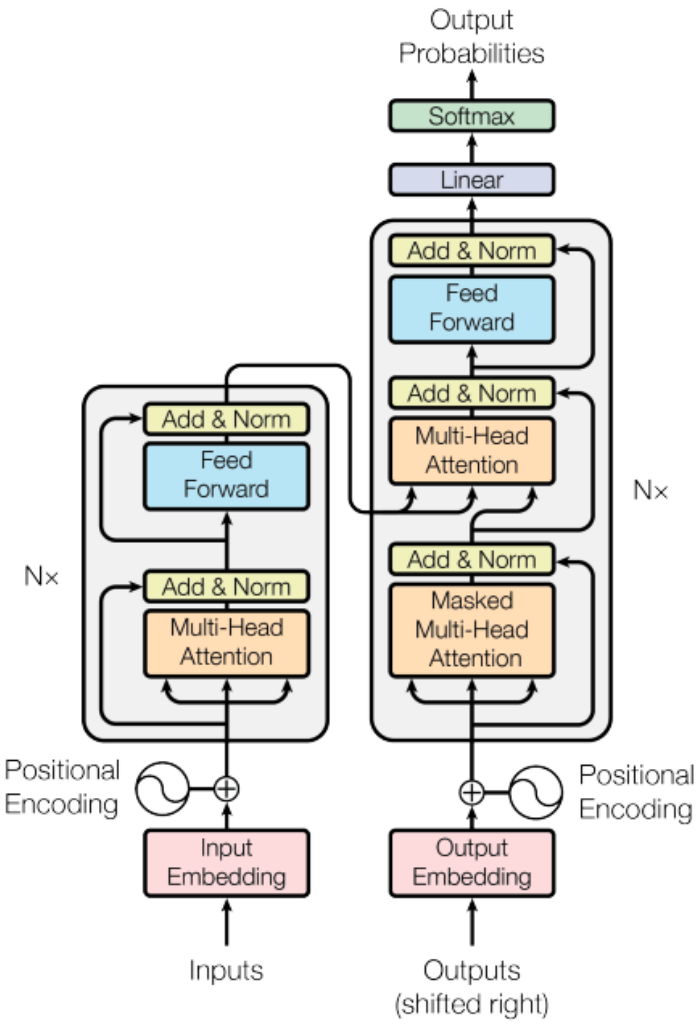

Figura 6. Transformer - Arquitectura del modelo. Fuente:[11]

- *Encoder:* compuesto de dos subcomponentes, el mecanismo de atención multi-cabeza (que son los que precisamente permiten prescindir de la atención secuencial de tokens), seguido de una capa *feedforward* completamente conectada en la que se aplica una función de activación no lineal.
- *Decoder:* similar al encoder, pero en este caso se añade un mecanismo de atención cruzada (*Masked Multi-Head Attention*) para generar la salida teniendo en cuenta la entrada en su totalidad.

Como se puede observar en la Figura 6, los módulos previamente mencionados de la arquitectura de Transformer tienen normalización de capa y residuos (Add & Norm). Por un lado, la conexión residual, permite sumar la entrada original de cada submódulo a su salida, no solo para preservar la información original, sino que también permite combatir el problema del *vanishing gradient.* Por otro lado, después de esta suma residual se aplica

una normalización para estabilizar y acelerar el entrenamiento, al menos en la versión original de la arquitectura Transformer, a pesar de que se ha observado que esta normalización no es estrictamente necesaria

3.1.5. Ingeniería del prompt

Al generar un Q&A por medio de inferencia con un LLM, se le deben indicar claramente las instrucciones que debe seguir este para evitar resultados que se desvíen de las intenciones del usuario, como por ejemplo, las alucinaciones [5] . Las alucinaciones *(hallucinations)* son aquellas generaciones de texto derivadas de la fuente original de texto [9] , estas pueden ser intrínsecas (en caso de que el texto generado contradiga al texto original), o extrínsecas (son aquellas que no pueden ser verificadas por el texto original).

Estas alucinaciones pueden darse por varios factores. Principalmente son debidas a *prompts* insuficientemente precisos que son interpretados por el modelo de forma errónea en determinadas ocasiones, por ejemplo, ante una estructura de *prompt* de otro modelo [4] .

Otra de las razones es un valor del hiperparámetro de la *temperatura* inadecuado. A pesar de que una temperatura elevada ofrece resultados menos deterministas (suaviza las diferencias entre los *logits* de las palabras), facilita la generación de alucinaciones [6] ya que la distribución de probabilidad de las palabras que el modelo elige para generar respuestas incluye una gama más amplia de palabras (tokens para el caso que nos ocupa) que no son necesariamente consideradas como aceptables. Además, un valor de la temperatura más elevado facilita el éxito de los famosos *jailbreaking prompts* [6] .

```
<s>[INST] <<SYS>>\n{system prompt}\n<</SYS>>\n{user prompt} [/INST] {model
answer} </s>
```

Al realizar inferencia con modelos LLM se debe tener presente que el formato que emplea cada modelo para interpretar el *prompt* correctamente es diferente. Por ejemplo, LlaMa 2 emplea un formato más estructurado y basado en etiquetas para delimitar el *system prompt*, la entrada del usuario y la consecuente respuesta [4] . Este formato es muy similar al de Mistral 7B (Mistral 8x7B en su configuración particular para realizar inferencia) ya que es un modelo que se basa en la arquitectura de LLaMa [7] , se utiliza el siguiente formato de prompt:

Además, como se puede apreciar en [8] , cada modelo requiere de una configuración particular para estructurar la interacción entre usuario y modelo, en su formato de inferencia. De esta forma quedan formateados tanto el texto de entrada como el de salida, lo que facilita su interpretación por el LLM.

Modelos	Templates
LLaMA-2-7B	<s>[INST]{Question}[/INST]{Answer}</s>
LLaMA-22-13B	<s>[INST]{Question}[/INST]{Answer}</s>
LLaMA-3-8B	< begin_of_text|><|start_header_id |>user<end_header_id |> {Question}< |eot_id|><| start_header_id |>assistant< |end_header_id|> {Answer}< |eot_id|>< |end_of_text |>
Qwen-2-7B	< |im_start|>user {Question< |im_end|> < |im_start|>assistant {Answer}< |im_end|>

Tabla 3.Plantilla para fine-tuning SFT de cada LLM. Fuente:[8]

Teniendo esto en cuenta se puede identificar que cada modelo tiene un funcionamiento particular a pesar de las generalizaciones que se realizan al usar cada uno de ellos. Por ejmplo, tal y como menciona Greg Brockman (presidente de OpenAI), se deben diferenciar 4 zonas en el *o1 prompt* de entrada al interactuar con la IA de OpenAI:

1. *Objetivo*: Describe el objetivo principal de la tarea a ejecutar por la IA y el resultado que el usuario pretende obtener.
2. *Formato de respuesta*: En esta sección el usuario debe establecer las directrices y límites que definan la estructura y formato en el que la respuesta de la IA debe ser presentada. Es de vital importancia que al usar LLMs o herramientas similares de NLP, el usuario sea conocedor de las limitaciones y prestaciones de esta, con el objetivo de evitar alucinaciones o respuestas incoherentes.
3. *Warnings*: A pesar de la eficiencia y optimización de las IA para evitar fallas, se deben esclarecer los aspectos más delicados de la petición del usuario para que sean abordados con especial atención.
4. *Contexto*: Finalmente se recomienda aportar cierto grado de contextualización al prompt para facilitar la búsqueda de la respuesta. A pesar de que las IA no son puramente RAGs, se basan en un espacio n-vectorial de embeddings, y al aportar contexto en el prompt el usuario facilita la búsqueda y generación de aquellos embeddings más relevantes.

A diferencia de los grandes modelos de lenguaje, el refinamiento del prompt no requiere de extensos procesos de entrenamiento donde se deban modificar los pesos de los parámetros para conseguir una mejora de rendimiento, sino que opera única y exclusivamente sobre el

conocimiento latente de la información almacenada en los embeddings del mismo, tal y como se deriva de [54] . No se debe olvidar que esta estructura es la presentada como "óptima" para el modelo particular de Open AI, y a pesar de que el funcionamiento del resto de modelos es similar, no se puede tomar como estructura única de prompt para interactuar con el resto ya que podría no ser tan eficiente.

3.2. Large Language Models

La evolución del lenguaje de procesamiento natural (NLP) y modelos de IA ha sufrido una profunda y agresiva transformación desde principios de los años 2000 con la evolución de las RNN [29] , contribuyendo así junto con la arquitectura Transformer a aumentar la presencia e importancia de los Grandes Modelos Preentrenados desde finales de 2010. En estos últimos años, los LLMs han ido cobrando mayor importancia debido a su superior capacidad para entender, generar y aprender texto con las características que emulan el comportamiento humano [28] , por medio del entrenamiento de grandes cantidades de datos y parámetros. Estos modelos no solo se quedan en las tareas de *chatbots*, también incluyen a asistentes humanos en todos los sectores profesionales ya sean de sanidad, banca o educación, desempeñando una labor superior [30] .

En este sentido, los LLMs se han consolidado como una herramienta de apoyo que permite a profesionales e investigadores centrarse en tareas de mayor valor añadido. Por ejemplo, pueden automatizar procesos laboriosos de menor complejidad, como la revisión bibliográfica, el resumen de casos clínicos o la búsqueda de jurisprudencia [31] , liberando así tiempo para el análisis crítico, la toma de decisiones o la innovación.

Los LLM se basan fundamentalmente en la arquitectura Transformer mencionaba previamente, lo que les permite capturar relaciones semánticas entre los tokens de entrada de forma paralela y no secuencial. Para llevar a cabo sus funciones siguen un flujo que se presenta de la siguiente manera:

3.2.1. Ajuste fino de LLMs

A la hora de hacer un ajuste fino de LLMs para cumplir con la tarea, por ejemplo, de servir como asistente de Q&A, existe un creciente interés en la optimización de la base de datos que se utiliza [8] . Por ejemplo, en [13] se recopiló un conjunto de preguntas con opción de múltiple respuesta, y procedieron a dividir los datos según cada modelo de lenguaje (pre-entrenado), filtrando por la precisión de estos y a posteriori condujeron un estudio sobre el ajuste fino con

instrucciones de estos modelos. La conclusión fue que, para lograr un ajuste fino exitoso, se debe conseguir mantener un conocimiento consistente de los parámetros de cada modelo tanto antes como después del ajuste. Sus conclusiones apoyan la idea de que se puede utilizar un modelo menos preciso y de menor capacidad para guiar a un modelo con mayores prestaciones en el ajuste fino IFT (*Instruction Fine-tuning*) [13] , que sigue la estructura que se presenta a continuación con GPT4 como ejemplo:

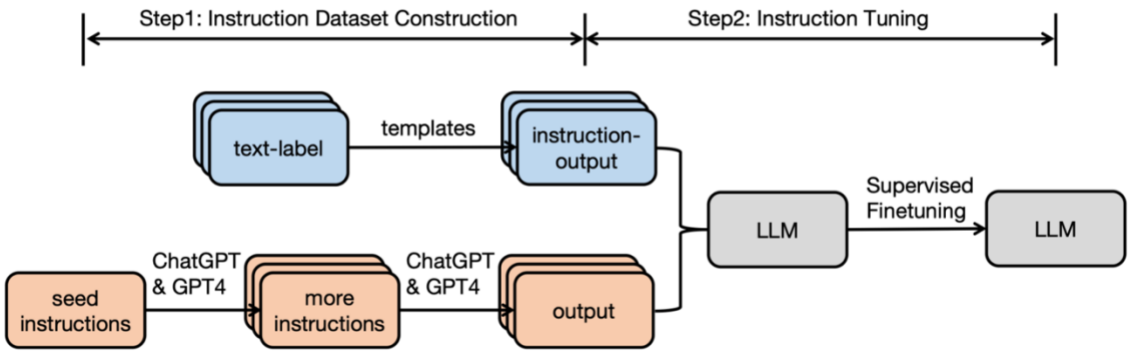

Figura 7. Pipeline general para IFT

Esta técnica de segmentar la información por la precisión de los modelos pre-entrenados es bastante común a la hora de hacer ajustes finos, no solo en IFT, también en SFT *(Supervised Fine-tuning)*. En [14] se toma la base de datos basada en un corpus de Wikipedia y se concluye que entrenar LLMs con bases de datos obtenidas de un corpus no riguroso, aumenta la cantidad y agresividad de las alucinaciones, haciendo que estas sean menos perceptibles y por ende más significativas.

Dichas conclusiones, [13] y [14] , demuestran como una variación del conjunto de datos tiene una gran influencia en el resultado final del ajuste fino, independientemente de la técnica utilizada (IFT o SFT). Por lo tanto, es conveniente seleccionar el modelo pre-entrenado según el criterio que se vaya a seguir para filtrar los datos, además de un corpus riguroso y significativo para el estudio, teniendo siempre presente la importancia de que el modelo sea capaz de aprender de la nueva base de datos sin olvidar aquello que ya sabía en su preentrenamiento.

También se debe prestar especial atención a la función final que va a desempeñar el LLM, ya que, aunque los datos hayan sido preprocesado y cautelosamente seleccionados, la función final se ve también afectada por el posterior proceso de ajuste fino. Como en [16] , donde se aprecia como IFT presenta notables ventajas en las acciones de generación de información a partir de

un corpus, pero empeora respecto a DPO *(Direct Preference Optimization)* en labores de respuesta con opción múltiple.

3.2.2. Inferencia con LLMs

En el campo de los grandes modelos de lenguaje el proceso de inferencia hace referencia al proceso a través del cual un modelo que ha sido previamente entrenado para realizar una serie de tareas concretas generando una salida a partir de una entrada. A pesar de la similitud conceptual, no debe confundirse el proceso de inferencia con el de entrenamiento. En este último, los pesos de los parámetros del modelo son ajustados por medio de la exposición a grandes volúmenes de datos independientemente de toda índole (numéricos, alfabéticos, categóricos…). Por otro lado, la inferencia solo se produce una vez se ha realizado este entrenamiento previo, y durante el proceso los pesos de los parámetros quedan inalterados. En la Figura 8, se puede apreciar visualmente esta diferencia, mientras que durante el entrenamiento el modelo ajusta sus parámetros mediante *backpropagation* del error, en la inferencia estos parámetros permanecen congelados, limitándose únicamente a producir una salida a partir de una entrada:

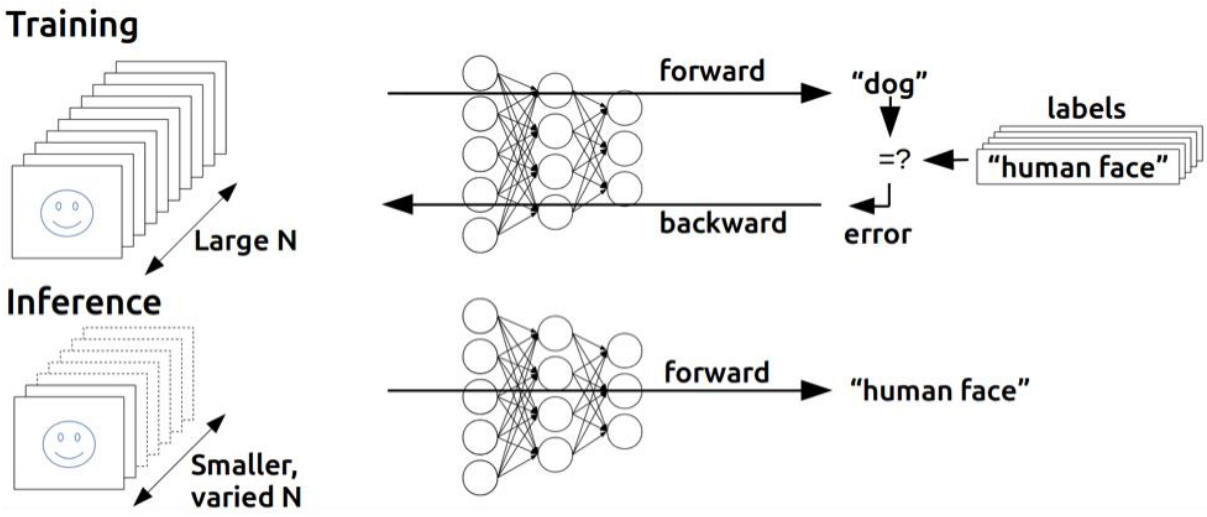

Figura 8. Comparación de entrenamiento e inferencia en Deep learning. Fuente: [70]

Los conceptos de inferencia y entrenamiento son de vital importancia para comprender las limitaciones de los LLMs. Como ya se ha mencionado en reiteradas ocasiones, el estado del arte de los grandes modelos de lenguaje está creciendo a un ratio de casi diez veces por año, y

con ello, el consumo computacional y la emisiones de carbono[53] . Es por ello por lo que resulta crucial conseguir disminuir los costos de computación y almacenamiento de los modelos sin comprometer sus capacidades [52] . Este objetivo se puede resumir en conseguir realizar de forma eficiente el proceso de inferencia sin sacrificar las predicciones generadas, ya sea mediante técnicas de destilación del conocimiento, optimización del uso de GPU, paralelismo o reestructuración de la arquitectura de redes neuronales.

La calidad de la inferencia de un LLM se mide en su capacidad para generar predicciones de tokens secuenciales basándose en la información de entrada y en los tokens previamente generados, y a pesar de que existen múltiples técnicas de inferencia, en el alcance de este proyecto solo se emplean tres de ellas:

- *Retrieved-Augmented Generation (RAG):* en el proceso de validación para verificar que la información de las preguntas y respuestas refleja la información que aparece en los documentos
- *Ingeniería del prompt:* para darle al modelo instrucciones claras sobre la tarea a desempeñar, pero sin necesidad de volver a entrenar o modificar pesos en los parámetros [54]
- *In-context learning*: junto con el prompt y las estructura que se le pide al modelo de salida, permite que el modelo aprenda el patrón de la tarea

3.3. Técnicas de generación de Conjuntos de Datos para fine-tuning

A la hora de mejorar el rendimiento de un LLM existen tres enfoques que reflejan su versatilidad, los cuales se explican brevemente a continuación:

- Aprendizaje de cero muestras *(Zero-shot learning):* permite medir la calidad de los modelos en tareas en las que no han sido específicamente entrenados con anterioridad, basándose únicamente en la información adquirida del contexto durante el entrenamiento. Según Pourpanah et al. [34] , persigue detectar instancias de clases no vistas durante el entrenamiento utilizando otro tipo de información latente como pueden ser descripciones textuales o atributos.
- Aprendizaje de pocas muestras (*Few shot learning*): este enfoque se emplea para realizar aprendizajes en contexto, y es especialmente útil para replicar la capacidad humana de aprender basándose en una pequeña muestra de ejemplos [35] .
- Ajuste fino (*fine-tuning)*: se basa en la continuación del entrenamiento de un modelo, es decir, a partir de modelos preentrenados, se vuelve a hacer otra iteración enfocándose en un campo de conocimiento o una tarea específica a la que se va a dedicar el modelo.

Para ello se modifican los valores de algunos de los pesos de los parámetros y es por ello por lo que es computacionalmente más costoso[3] que los dos casos anteriores. Es imprescindible realizar este entrenamiento con un dataset depurado y asegurarse de que el modelo "olvida" la menor información posible [22] de la que ya conocía después del primer entrenamiento.

Esta última, entre otras cosas, permite a los usuarios entrenar a los modelos con bases de datos adicionales refinando parámetros concretos del modelo dejando el resto inalterados [3] , esta técnica se denomina PEFT (*Parameter-Efficient-Fine-tuning*).

También se debe destacar la importancia del RAG a la hora de optimizar la generación que proporciona el propio LLM [4] . Hoy en día RAG ya no solo es una alternativa de mejora de los LLM, sino que también se combina con el proceso de fine-tuning para evitar el reentrenamiento constante del modelo en casos en los que la información se vaya actualizando con frecuencia.

Es decir, por un lado, el proceso de fine-tuning modifica los pesos de algunos de los parámetros mientras que RAG permite mejorar las respuestas de los modelos en tiempo real, integrando información externa sin modificar los parámetros internos. Un estudio realizado por Angels Balaguer, et al. [10] , destaca los significativamente mejores resultados obtenidos por un LLM al que se le ha aplicado un ajuste fino frente a la alternativa de RAG. Sin embargo, también hay una notoria diferencia en el costo inicial de ambas alternativas, haciendo que RAG no solo sea una opción más sencilla de implementar y con resultado competitivos, sino que además es menos costosa.

Por ejemplo, en la Figura 9 se puede apreciar el flujo de trabajo que sigue una implementación de un sistema de respuesta de preguntas donde se combina el fine-tuning y RAG para mejorar las capacidades del modelo.

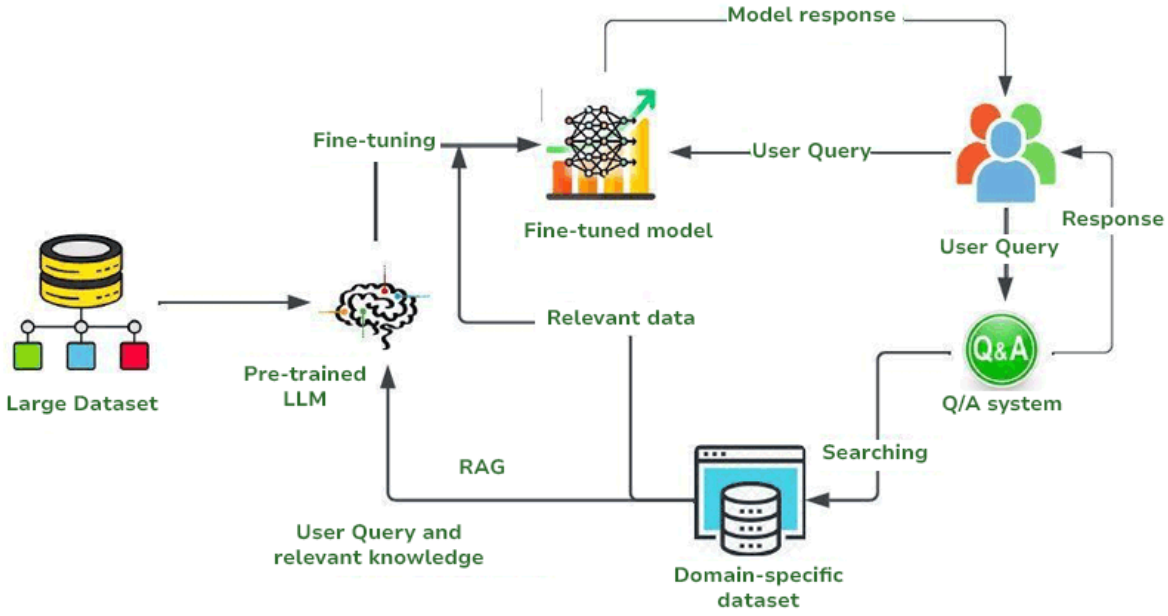

Figura 9. Arquitectura de un sistema de preguntas y respuesta con ajuste fino y RAG. Fuente:[71]

3.4. Benchmark de validación

En la elección de modelos radica la parte más crítica del proyecto, ya que si se eligiese uno sin limitaciones impuestas por los entornos locales se obtendrían resultados mejores y con menores esfuerzos a pesar del coste computacional extra. Por el otro lado, si se eligen modelos muy antiguos, el output puede no ser el deseado. Para encontrar el balance, se debe retroceder en la arquitectura de los modelos hasta su representación de los embeddings, ya que una buena representación es la clave para tener un modelo adaptable, flexible y a la vez robusto, tal y como explican Wang et al. en [72] .

3.4.1. MTEB Benchmark

La representación de los embeddings está ganando mayor atención a medida que se avanza en el mundo de NLP, ya que se ha demostrado que las bases para poder mejorar el desempeño de los modelos en tareas específicas radican en una buena representación de estos embeddings. A pesar de que los embeddings estáticos como FastText o Word2Vec carecen de contextualización y por ende tienen un peor rendimiento, presentan una gran ventaja frente a los novedosos modelos de embeddings contextualizados: el tamaño. Y es que realizar cualquier tarea con este tipo de emebeddings agiliza los procesos de entrenamiento e inferencia de los modelos y por ello se está empezando a ver una tónica generalizada respecto a los embeddings,

cada vez más, se están empleando embeddings estáticos como punto de partida y mejorándolos para aproximarse a las representaciones de los embeddings contextualizados.

Una forma de ver los resultados de la constante evolución de estas representaciones es MTEB (Massive Text Embedding Benchmark), que consiste en un conjunto estandarizado de pruebas diseñadas para evaluar el rendimiento de estos modelos en una variedad de pruebas de procesamiento del lenguaje natural. Desde análisis de sentimientos hasta tareas de *clustering*, clasificación y extracción de conocimiento (*Retrieval*).

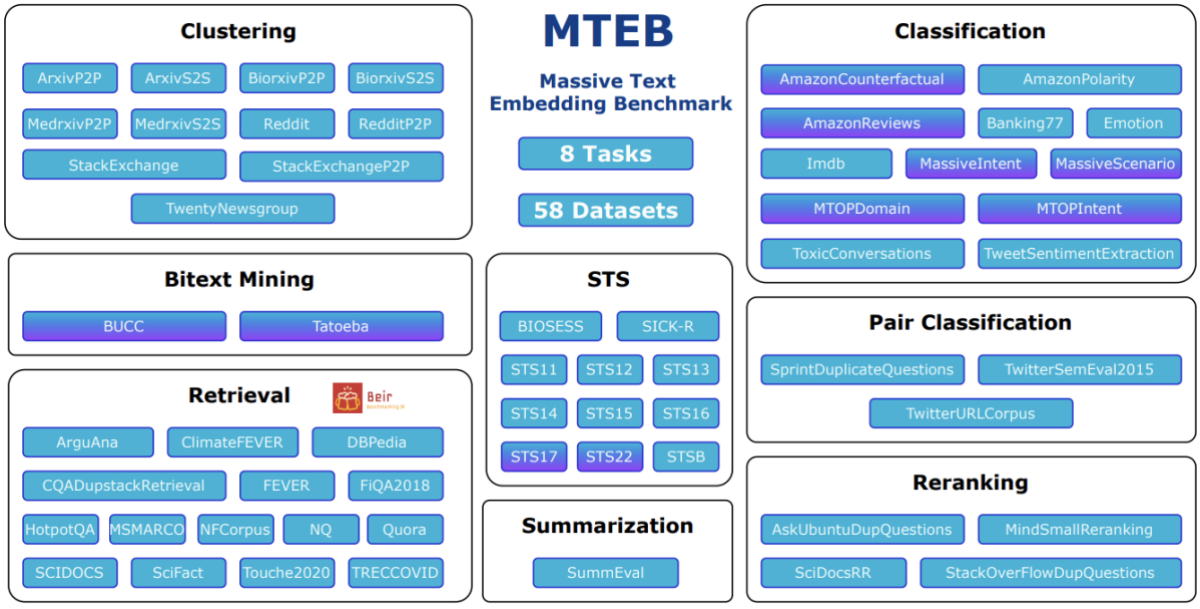

Figura 10. Tares iniciales de MTEB (las tareas multilingues están sombreadas en morado). Fuente: [69]

A pesar de ser un benchmark que lleva en uso varios años, no ha dejado de evolucionar constante y actualmente cuenta con más de 131 tareas en 20 dominios diferentes y contribuya activamente a la comunidad global evaluando sus tareas en más de 250 idiomas. Sin embargo, para el caso de uso que ocupa este proyecto se ha empleado la versión inglesa de MTEB en lugar de la multilingüe. Cabe destacar que un claro ejemplo que apoya la evolución del benchmark es su extensión MMTEB (Massive Multilingal & Multimodel Text Embedding Benchmark), que expande el enfoque inicial de MTEB para abarcar también modelos multimodales. Aunque los modelos multimodales se escapan del alcance de este proyecto, MMTEB debe ser una herramienta para considerar en caso de expandir el proceso de generación en iteraciones futuras.

Por todo lo expuesto anteriormente, MTEB se ha considerado como el estándar de referencia para la elección de los modelos utilizados en la generación del dataset, en especial a la hora de

elegir el modelo para la validación. Los modelos de *sentence embedding* que se evaluaron para la aplicación del RAG de validación fueron aquellas con:

- Mayor capacidad en tareas de recuperación semántica y clasificación, ya que son clave para la extracción de contexto relevante
- Tamaño limitado para que sea reproducible en entornos locales

Como se aprecia a continuación, con los datos obtenidos directamente de [69] , los modelos evaluados presentan un rendimiento que sigue un patrón identificable. En general los modelos evaluado destacan por sus capacidades en clasificación y recuperación, mientras que tareas como *clustering* y *reranking* son notablemente peores:

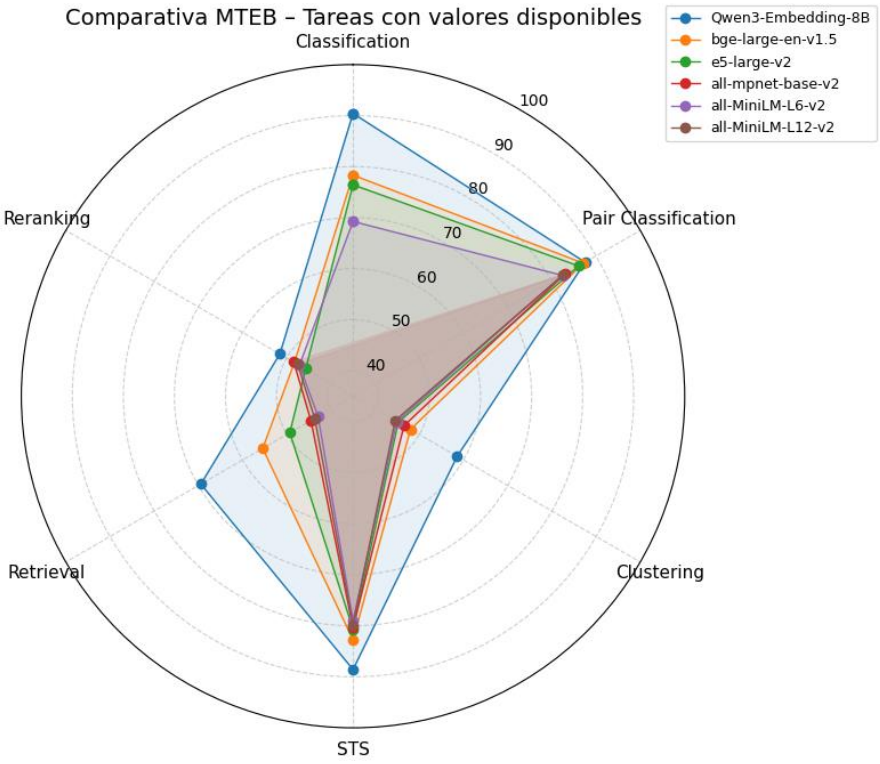

Figura 11. Comparativa de los modelos evaluados en MTEB

En esta comparativa se aprecia como también se ha incluido el Qwen3-Embeding-8B, que es un modelo mucho más grande que cuenta con una arquitectura de LLM, lo que permite capturar representaciones semánticas más complejas y contextualizadas. Sin embargo, su tamaño conlleva un mayor coste computacional, pero el hecho de compararlo con modelos generalistas permite contextualizar el proyecto. Los modelos elegidos para realizar el análisis son de carácter generalista (a excepción del mencionado Qwen3 que simplemente pretende exponer la diferencia que existe entre modelos más sencillos y modelos basados en LLMs) en su totalidad, y algunos de ellos de embeddings estáticos: all-MiniLM-L6-v2, all-MiniLM-L12-v2 y all-mpnet-base-v2.

Este comportamiento que se aprecia en la Figura 11 permite identificar que los modelos generalistas que son entrenados para tener representaciones vectoriales útiles en múltiples tareas, para seleccionar los modelos de acuerdo al ajuste que se les pretende realizar o simplemente eligiendo el modelo que mejor desempeño tenga para una tarea específica. A pesar de ello destaca el bge-large-en-v1.5, ya que mantienen unas capacidades relativamente estables y ofrece un rendimiento balanceado.

4. Metodología

En esta sección tiene por objetivo, explicar detalladamente los diferentes pasos que se han seguido en la creación del dataset sintético. Aun así, conviene definir el flujo de trabajo completo del proyecto para tener un mejor entendimiento de lo que se ha llevado a cabo. A pesar de las simplificaciones hechas en la Figura 12, pretende representar con claridad los rasgos principales del proyecto:

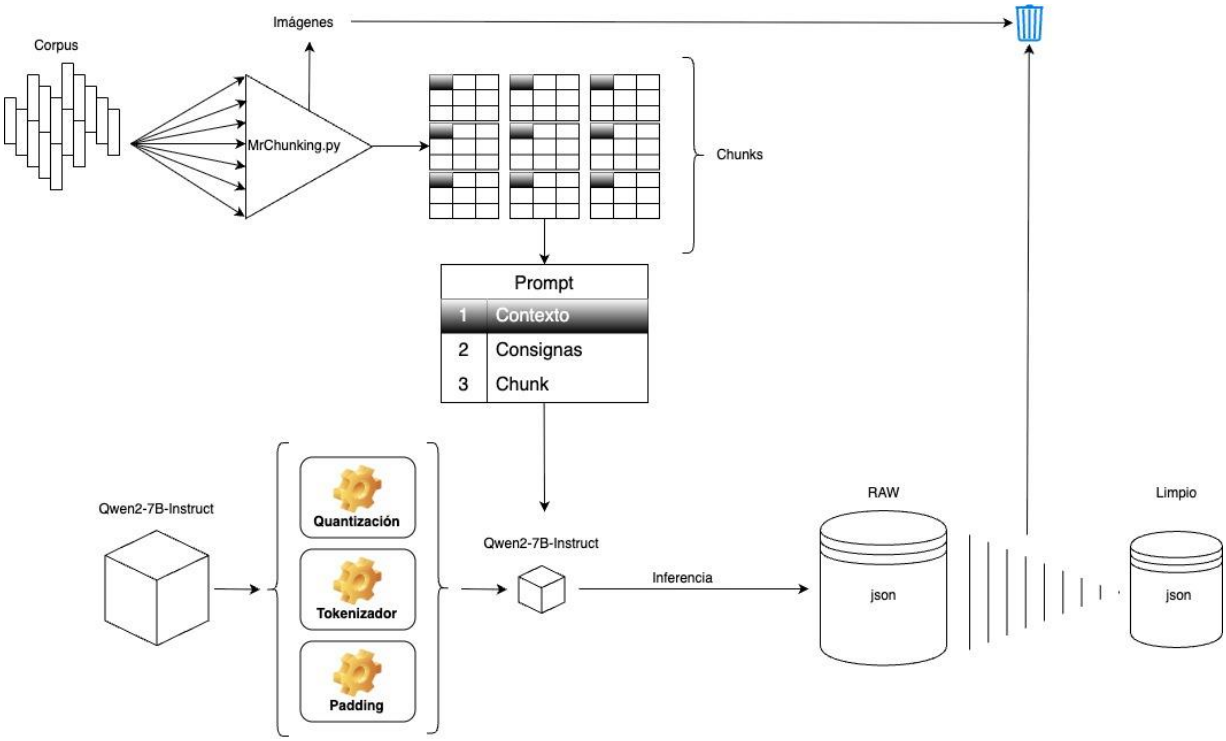

Figura 12. Generación - Flujo de trabajo del proyecto

Como se presenta en la Figura 12, se pueden diferenciar varios bloques relativamente independientes que forman el proyecto en su totalidad. En primer lugar, hay una fase de selección, filtrado y preparación de los documentos, seguida de una fase donde se procede a la segmentación de los textos en unidades textuales manejables y con información autocontenida a las que se le denominará como chunks. Una vez obtenidos estos fragmentos de texto, se clasificarán de forma que siempre uno de los chunks de cada documento será el utilizado como contexto para realizar la inferencia, este fragmento de código se corresponde con el *abstract* de cada documento. Con el chunk de contexto y los demás fragmentos de los documentos, se diseña un prompt con las consignas pertinentes para generar las preguntas y respuestas basadas en la información de los documentos.

En paralelo, se carga y configura el LLM, en este caso el Qwen2 7B en su versión instruct, al que se le enviará como entrada el prompt diseñado anteriormente. Finalmente, se cargan todos los pares pregunta-respuesta en un archivo json al que se le realiza un postproceso para limpiar las preguntas y respuestas que no representan información relevante para el posterior ajuste fino, ya sea por falta de claridad o por referenciar partes de documentos de forma poco específica.

Una vez se ha obtenido el dataset sintético, se procede a la validación de este, en este caso aplicando un RAG para verificar que la información que ha generado el LLM en el proceso anterior, concuerda con la información que aparece en los documentos, y más importante aún, verificar que toda la información ha sido procesada:

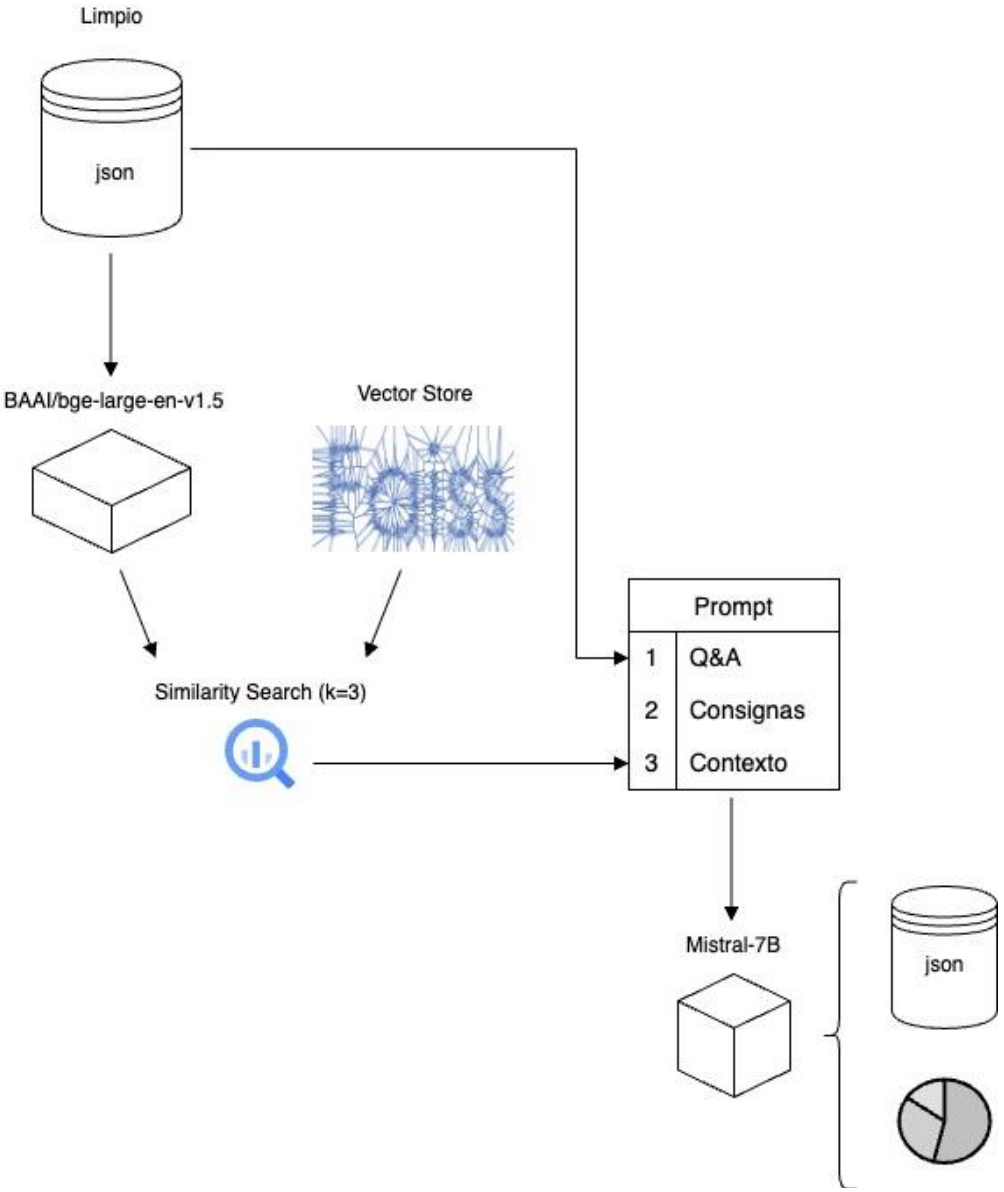

Figura 13. Validación - Flujo de trabajo del proyecto

Una vez más, el flujo de trabajo del RAG consiste en segmentar de la misma forma que se ha hecho en la primera parte del proyecto los documentos en fragmentos textuales más pequeños. Esta división es exactamente la misma que la realizada en la primera parte del proyecto, de esta forma se conserva la estructura de cada chunk y facilita el trabajo de validación de todos el proceso.

Con el modelo *BAII/bge-large-en-v1.5* se convierten todos los fragmentos de texto en vectores de embeddings y paralelamente se construye un índice FAISS (*Facebook AI Similarity Search)* para realizar búsquedas en el espacio vectorial de embeddings de forma eficiente. Sobre estas herramientas, se evalúan todos y cada uno de los pares pregunta-respuestas generados con anterioridad por el modelo Qwen2. Para ello, se emplea el modelo Mistral 7B, que no ha participado en la generación del dataset, y se le proporcionan como entradas las preguntas con sus respectivas respuestas y los chunks (si es que hay) que se consideran relevantes para el par pregunta-respuesta evaluado. De esta forma se está realizando una técnica de validación cruzada entre LLMs para asegurar que la información generada por el modelo Qwen2 proviene estrictamente del texto y no es fruto de alucinaciones o información previa que el modelo ya conocía.

Finalmente, se almacenan todos los chunks a los que ha accedido el Mistral 7B para justificar cada una de las líneas del dataset y el total de preguntas y respuestas de las cuales no se ha podido encontrar un chunk que justifique el contexto bajo el cual han sido generadas. Además, el Mistral 7B también proporciona una explicación de porque el par pregunta-respuesta ha sido o no considerado como correcto por el modelo, de este modo garantizamos una validación tanto cuantitativa como cualitativa.

4.1. Justificación

4.1.1. Modelos Multimodales

En primer lugar, se debe hacer referencia a los modelos utilizados, en este caso Qwen2, Mixtral 8x7B y Mistral 7B. Como bien se puede apreciar ninguno de ellos se trata de un modelo multimodal, es decir, son estrictamente textuales (al menos en las versiones en las que han sido utilizados), diseñados para procesar y generar información en formato de tokens lingüísticos.

Los modelos multimodales, como el reciente GPT-4o, permiten procesar no solo información escrita en forma de textos que se fragmentan para poder ser debidamente embebidos, sino que además permiten el procesamiento de imágenes, figuras, audio e incluso vídeo. Estas facetas los convierten en herramientas muy potentes en el análisis de datos donde el preprocesamiento es mínimo o inexistente, como pueden ser documentos escaneados o vídeos de conferencias en crudo.

Por ello se debe destacar, que en el alcance de este proyecto no se han considerado este tipo de modelos por varias razones:

〉 Corpus documental: los documentos de los cuales se ha partido para generar el dataset han sido exclusivamente extraídos de la plataforma arXiv o de plataformas derivadas de Google scholar. Esto implica que el texto que se ha procesado es semiestructurado (o estructurado en su minoría, en los casos donde se procesan tablas) y la cantidad de imágenes o figuras visuales que aparecen en los artículos científicos es despreciable en comparación con la cantidad de texto. Concretamente, para los documentos utilizados en la creación de dataset, tan solo un 3,73% de la información de estos aparecía en un formato diferente al texto. En este cálculo no se han tenido en cuenta referencias externas como pueden ser repositorios de GitHub u otras fuentes citadas en los textos.

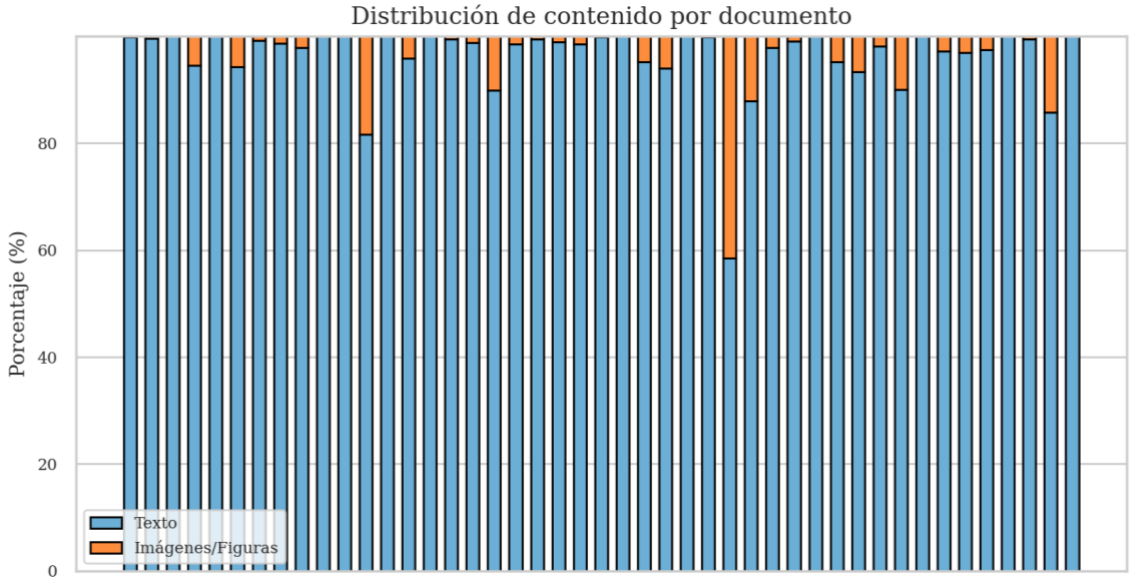

Figura 14. Distribución porcentual de información en texto y en imágenes

En la Figura 14, se representa una muestra de los documentos que se han utilizado como corpus para la realización del dataset sintético, y como se puede apreciar, en su vasta mayoría los documentos tienen una cantidad mínima de información almacenada en formato de imágenes, justificándose así el 3,73% de información en imágenes sobre el total de la información procesada en todo el corpus. Este cálculo se ha realizado teniendo en cuenta el área que ocupan proporcionalmente el texto y las imágenes de cada documento junto con el porcentaje de preguntas descartadas.

- Limitaciones técnicas: a pesar de que los modelos multimodales son prometedores en su desempeño, aun presentan limitaciones respecto a los modelos puramente textuales en tareas estrictamente textuales, como podría ser en este caso la generación de preguntas y respuesta sobre una imagen o un diagrama. Además, tal y como se deriva de [55] ,una vez aplicado *MultiBench*, se observó que los modelos unimodales generalmente resultan ser más robustos en tareas puramente textuales que los multimodales.

4.1.2. Corpus de documentos

Una de las principales cuestiones a plantear al inicio del proyecto ha sido el corpus de documentos sobre el cual iban a basarse las preguntas y respuestas que constituirían el dataset sintético. Esta selección de documentos no es trivial pues se deben cumplir una serie de condiciones para garantizar la veracidad del dataset:

En primer lugar, los documentos seleccionados debían ser invisibles para los LLMs que se iban a utilizar en el posterior fine-tuning y para los modelos empleados durante el proceso de inferencia en la generación de preguntas y respuestas, ya que, si los modelos hubiesen tenido acceso previo a dicho contenido, se podrían introducir sesgos no deseados derivados del conocimiento latente del propio modelo en lugar de generar información basada exclusivamente en el contenido de entrada. Esto implica que se deben usar documentos suficientemente novedosos debido a la velocidad evolutiva del estado del arte de la Inteligencia Artificial Generativa.

En segundo lugar, esta información contenida en los documentos debía ser lo suficientemente rica y variada para permitir un fine-tuning significativo. A pesar de no disponer de grandes unidades de computación no se debe perder de vista que el objetivo primero del proyecto es el de desarrollar e implementar un proceso completo con unidades de computación limitadas en un entorno local.

Finalmente, el corpus de documentos debía contar con cierta estructura unificada, ya que se presupone que la implementación del flujo de trabajo se realizaría sobre una serie de documentos privados de una misma entidad y por ende las directrices que marcan la estructura de los mismo debe ser consistente. Además, el propósito de realizar un ajuste fine es conseguir que el modelo aprenda en profundidad a cerca de un tema en concreto, y por ello no tendría sentido seleccionar documentos que no versan alrededor de una misma temática, como puede ser una entidad privada.

Con estos argumentos presentes, la opción más lógica y efectiva seria contar con los documentos de una entidad privada para poder aplicar directamente el proyecto. No obstante, durante el inicio del proyecto, no se consiguió acceder a documentación privada de ninguna empresa debido a la dificultad de firmar NDAs. Por ello finalmente se ha optado por realizar el dataset sintético a cerca de las propias referencias de este proyecto posteriores al lanzamiento de los modelos utilizados y que se encuentren en la base de datos de ArXiv. Con este enfoque se consigue:

- *Información desconocida:* Garantiza que los modelos no han sido entrenados con esa información, pues no existía cuando estos fueron entrenados, de forma que no se introducen sesgos ni *data leakeges*.
- *Estructura consistente:* asegura que los documentos siguen una estructura unificada, ya que se obtienen todos de ArXiv y aunque no sigan todos las mismas directrices en cuanto a estructura de documentos, no dejan de ser artículos académicos que siguen una serie de normas preestablecida, como puede ser empezar por el título, autores y abstract; para acabar con una sección de referencias.
- *Control del contenido:* al tratarse de material revisado y conocido por el autor, facilita la labor de revisión y validación manual. Además, el contenido de los documentos versa sobre un mismo tema que engloba todo lo relacionado a este proyecto.
- *Volumen de datos:* la principal dificultad era encontrar suficientes datos sobre un mismo tema y relativamente nuevos en el tiempo, por ello, agrupar todos los documentos que reunían las características previamente explicadas era la única forma que se contempló de lograr conseguir un volumen de datos suficientemente grande y a la vez procesable por la capacidad de computación con la que se contaba.

Al revisar trabajos de similar índole, se observa que varios de ellos parten de datasets estructurados y ya creados, por ejemplo, en otro idioma, como puede ser el caso de [32] , que parte del famoso dataset de la universidad de Stanford, SQuAD [33] en la versión del idioma serbio. No obstante, esta técnica no sería una buena práctica en el alcance de este proyecto, pues el objetivo es crear el dataset desde cero sin contemplar la traducción o técnicas de *data augmentation* de otros datasets a pesar de su efectividad, como se aprecia en trabajos como [56] y [57] .

4.2. Recursos empleados

El proyecto ha sido desarrollado casi en su totalidad en Google Colab Pro, debido a los requerimientos de hardware para poder realizar inferencia con LLMs y descargar los pesos de los parámetros de los modelos. La principal limitación es la VRAM (Memoria de vídeo) que

es donde se almacenan los pesos de los parámetros para que la inferencia se pueda realizar de forma razonable, ya que en caso de que los pesos no quepan en su totalidad en la VRAM, se produce una severa degradación del rendimiento y por ende unas prestaciones menores que no reflejan las capacidades del modelo.

En primer lugar, se hicieron varias iteraciones con LM Studio, una herramienta que permite ejecutar modelos LLM en local, pero no permitía la libertad y flexibilidad que requería el proyecto cuando se inició, a pesar de que es una herramienta que ha evolucionado mucho y ya cuenta con más funcionalidades.

Como no se tenía disponibilidad absoluta de un ordenador con suficiente VRAM como para ejecutar los modelos de lenguaje con los que se estaba iterando (Mixtral8x7B que finalmente fue descartado precisamente por esa razón), se decidió hacer uso de una cuenta de Google Colab pro. Esta cuenta permite acceder a GPU como la A100:

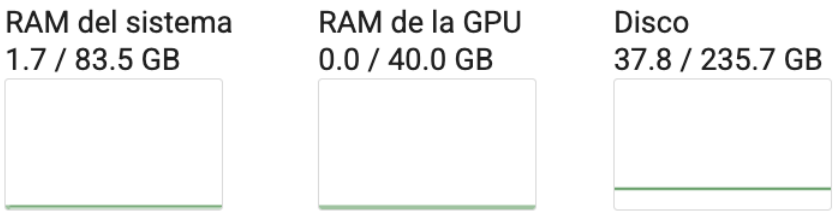

Figura 15. Prestaciones de la A100 en Google Colab

A pesar de la ventaja que presenta el acceso a unidades de computación potentes, también suponen ciertas correcciones respecto a los entornos locales. En primer lugar, se debe destacar que la cuenta era compartida por dos alumnos (uno de los alumnos desarrollaba el proceso de generación del dataset y otro el posterior ajuste fino del modelo), por lo tanto, se debía organizar el entorno de Google drive y coordinarse en las ejecuciones para no interrumpir el trabajo del otro. En segundo lugar, las dependencias debían ser instaladas cada vez que se accedía al entorno ya que no se guardan indefinidamente. Finalmente, aunque la cuenta de pago era el plan *pro,* tiene un límite de tiempo según el consumo de recursos (consumo elevado en este caso ya que se realizaba inferencia con LLMs), lo que dificultaba mucho el guardado de información.

4.2.1. Librerías y repositorios

A continuación, se presentan una seria de librerías, módulos y recursos que han sido utilizados durante el desarrollo del proyecto, todos ellos englobados en el ecosistema Python. Cabe

destacar que no todos los elementos de esta sección han sido incluidos finalmente, pero sí que han sido utilizados en varias de las iteraciones y por ende se consideran importantes ya que han permitido la consecución del proyecto.

- Embeddings y segmentación de texto
  - **pdfminer.six**: librería de Python para extraer información de archivos PDF, especialmente texto, metadatos y estructura del documento.
- Procesamiento de Lenguaje natural y LLMs
  - **transformers**: es la librería por antonomasia para la carga de grandes modelos de lenguaje y tokenizadores desde Hugging Face. Ha sido utilizada tanto en la fase de generación (Qwen-2) como en la de validación (Mistral-7B). Ha sido especialmente útil por su integración nativa con PyTorch, TensorFlow y bitsandbytes, además para las tareas de generación se ha implementado el método pipeline simplificando considerablemente la gestión de los LLMs.
  - **torch**: se trata de un framework empleado en proyectos de *deep learning* ya que permite gestionar tensores y ejecución en la GPU. A pesar de que en este proyecto no se ha realizado entrenamiento de modelos (una de las funcionalidades más relevantes de PyTorch), su uso ha sido fundamental en la carga y ejecución de la inferencia en los modelos.
  - **tensorflow**: este framework fue considerado en fases exploratorias iniciales del proyecto para la carga y ejecución de modelos alternativos, pero sin embargo debido a la elección de modelos del proyecto se optó por PyTorch.
  - **bitsandbytes:** permite cargar modelos grandes en menor memoria mediante cuantización eficiente en 8 bits, 4 bits y otras cuantizaciones reducidas, que como ya se ha explicado con anterioridad, permite disminuir el consumo de memoria de GPU. Esta última parte fue especialmente importante durante fases iniciales del proyecto ya que se contaba únicamente con la versión gratuita de Google Colab con recursos computacionales y temporales limitados.
  - **huggingface_hub:** permite la autenticación, carga, descarga y gestión de modelos y archivos de Hugging Face.
- Embeddings y recuperación semántica
  - **langchain-huggingface** y **langchain-community**
  - **faiss-cpu**: se trata de la biblioteca para búsqueda eficiente de vectores (en este caso la versión de CPU ya que se pretende liberar GPU para la inferencia.
- Manipulación de datos
  - **numpy:** librería base para operaciones vectoriales y numéricas.
  - **pandas:** empleada para la gestión y manipulación de estructuras de datos tabulares.

- **scikit-learn:** utilizado en fases exploratorias del modelo para el análisis de los datos del corpus y la validación de resultados
- **polars**: utilizado al final del proyecto como alternativa a pandas para el procesamiento de datos a gran escala y de forma paralela, a pesar de que finalmente no fue incluida en el proyecto. Se debe destacar que en casa de tener que escalar el proyecto sería conveniente realizar una migración a esta librería por ser más eficiente y no estar limitada por la capacidad de memoria RAM.

〉 Automatización y control de flujo

- **os:** permite la navegación, lectura y escritura de archivos y directorios.
- **json**: empleado para leer, escribir y manipular archivos en formato JSON, que es el formato que se ha empleado para almacenar todas las iteraciones del dataset.
- **sys:** utilizado para el manejo de errores y controles de flujo durante las ejecuciones.
- **re:** aplicado para expresiones regulares (patrones para buscar, coincidir o manipular texto). Esta librería es de vital importancia para filtrar los pares pregunta-respuesta no deseados durante la generación.

〉 Repositorios de datos

- **arXiv y wikipediadatabase:** son las fuentes documentales sobre las que se sustenta el corpus del proyecto ya que, como se menciona con anterioridad, no se ha podido acceder a un corpus de documentación privada. Wikipediadatabase ha sido utilizado en iteraciones intermedias, ya que la información se obtenía de forma ya estructurada en un json.

〉 Interfaz y entorno de ejecución

- **IPython.display:** utilizado en el entorno de Google Colab para limpiar y controlar las salidas de las celdas, ya que la generación continua de preguntas y respuestas dificulta la trazabilidad de la ejecución.

4.3. Técnicas Utilizadas en la Elección de LLM

4.3.1. Comparación y selección de modelos

Uno de los elementos más críticos durante la creación del dataset sintético ha sido la selección del LLM que se iba a utilizar para realizar la inferencia. No solo por la cantidad de modelos que existen en abierto, sino que también por la complejidad técnica de implementarlos y que sean adecuados para el lenguaje y tarea que se iba a desarrollar.

En primer lugar, por volumen de datos y por simplicidad de implementación del proyecto, tanto los documentos empleados como los prompts de entrada de los modelos se han realizado en

inglés. A pesar de que la gran mayoría de modelos están adaptados a varios idiomas (configuraciones multilingües), generalmente han sido entrenados en una primera instancia con datos de entrada en inglés, esto resulta muy relevante ya que como se ha mencionado con anterioridad en este documento, el ajuste de los pesos de los parámetros se realiza durante el entrenamiento y si este ha sido en una lengua en concreto, las connotaciones y relaciones lingüísticas entre tokens serán más precisas en ese idioma que en cualquier otro, por muy extenso que sea el diccionario de tokens de cada modelo.

Teniendo estos conceptos presentes, la elección del modelo a utilizar está sujeta a cuatro principales restricciones:

- Comprensión del lenguaje técnica, ya que se procesan documentos académicos de ArXiv.
- Capacidad de generación relevante basado en contexto limitado (en este caso, el chunk y la metainformación del documento en forma de contexto).
- Mantener tiempos de procesamiento razonables durante el proceso de inferencia y requerimientos de memoria limitados.
- Ser accesibles desde Google Colab y permitir flexibilidad de adaptación de características bajo los que opera el modelo

Debido a las consignas con las que se estaba trabajando, se barajaron principalmente 4 modelos, a pesar de que muchos otros fueron tenidos en cuenta, no pasaron a la fase final de decisión. También cabe destacar que muchos modelos han sido desarrollados desde el inicio del proyecto y que por ello no han podido ser utilizados como la evolución de Qwen2, Qwen3 [58] con ventanas de contexto de 128K tokens o Gemma 3 con configuraciones desde 1 a 27 billones de parámetros [59] . Como se ha indicado, estos modelos no han sido utilizados ya que, a su salida, el proceso de generación con LLMs integrados ya estaba demasiado avanzado. Se tiene como límite de selección de modelos enero del 2025.

4.3.2. Creación y aplicación de benchmark

Para la selección del modelo de lenguaje más adecuado para la generación de pares de preguntas y respuestas, se llevó a cabo un benchmark comparativo de los modelos que se podían emplear para el proyecto. El objetivo principal de este benchmark de evaluación es el de identificar el modelo que ofrezca un equilibro lo más cercano al óptimo en términos de calidad de la salida, eficiencia computacional y tiempo de inferencia

Debe tenerse en cuenta, que esta comparativa se hizo con el único objetivo de seleccionar el modelo con el que realizar la inferencia para generar los pares de preguntas y respuestas, pero para la etapa de validación donde se implementó un sistema RAG no se llevó a cabo esta comparativa, ya que con la experiencia de uso que se había obtenido en la primera etapa directamente se seleccionó el que se cree era el mejor modelo para desarrollar la tarea.

La comparativa se realzó sobre una muestra representativa de fragmentos de texto (chunks) y se midieron los siguientes indicadores de rendimiento:

〉 *FLOPs (Floating Point Operations)*: este indicador está muy extendido en el contexto del entrenamiento de LLMs ya que hace referencia al número de operaciones en coma flotante que un modelo realiza durante su ejecución y permite obtener un valor del coste computacional independientemente del hardware que se utilice. El propósito de medir los FLOPs como el número de operaciones en coma flotante, es debido a que este tipo de operaciones son la base de cálculo de las redes neuronales en la multiplicación de matrices [61] .

Además, la escalabilidad de este parámetro es relativamente predecible con la cantidad de parámetros del modelo, la longitud de entrada y de salida y el número de capas, tal y como se aprecia en la *Tabla 4* del estudio de Hoffman et al. acerca del tamaño óptimo que debe tener un modelo según la cantidad de parámetros, tokens y presupuesto [61] . En definitiva, esta variable nos permitirá evaluar la eficiencia de cada modelo para entradas iguales, y será especialmente interesante ya que el número de parámetros de los modelos evaluados también es igual en todos ellos.

Parameters	FLOPs	FLOPs (in *Gopher* unit)	Tokens
400 Million	1.92e+19	1/29,968	8.0 Billion
1 Billion	1.21e+20	1/4,761	20.2 Billion
10 Billion	1.23e+22	1/46	205.1 Billion
67 Billion	5.76e+23	1	1.5 Trillion
175 Billion	3.85e+24	6.7	3.7 Trillion
280 Billion	9.90e+24	17.2	5.9 Trillion
520 Billion	3.43e+25	59.5	11.0 Trillion
1 Trillion	1.27e+26	221.3	21.2 Trillion
10 Trillion	1.30e+28	22515.9	216.2 Trillion

Tabla 4. Estimación del óptimo de FLOPs y tokens de entrenamiento para diferentes tamaños de modelos

Finalmente, cabe destacar que en esta evaluación solo se han tenido en cuenta los tokens generados para la estimación del cálculo de los FLOPs, por la particular característica de los modelos ensayados, ya que todos son de arquitectura *decoder-only* (Mistral, Qwen y Llama). A pesar de que el procesamiento del prompt tiene cierto peso, es la generación autorregresiva la que constituye la mayor parte del coste computacional de este tipo de modelos [60] , y como al ejecutar el benchmark el prompt final del proyecto aun no estaba definido, se ha decidido dejar fuera del cálculo de este indicador los tokens de entrada.

- *Velocidad en A100*: se valora la velocidad en una A100 ya que es la que se emplea durante el proyecto. A pesar de que ya se está contabilizando la eficiencia computacional de cada modelo con los FLOPs, es conveniente evaluar el tiempo que tarda cada modelo en realizar el benchmark. Esto permitirá tener una idea clara y bien definida del tiempo total para conseguir el dataset y además tiene en cuenta también el tiempo de procesamiento de los tokens de entrada. En definitiva, permitirá vislumbrar diferencias significativas entre modelos de forma cuantitativa (tiempo total en realizar la tarea) que sobre todo cobrará relevancia cuando se compare con la calidad de la salidas generadas.

- *Tokens*: al igual que sucede con los otros dos indicadores, los tokens generados ayudarán a comprender mejor las diferencias de rendimiento entre modelos y las diferencias entre los tiempos de generación.

- *Pares Pregunta-respuesta*: finalmente, también se debe evaluar el número de salidas que genera cada modelo por prompt, ya que el tiempo de generación, los FLOPs y los tokens que generan, estas directamente relacionados con la cantidad de pares de pregunta-respuesta que genera el modelo para cada entrada de texto.

  Además, también se evaluará de forma no cuantitativa la calidad técnica de cada modelo. A pesar de no contabilizar numéricamente este indicador, resulta el más importante de todos para el caso de aplicación, ya que el desempeño de un modelo puede ser extremadamente eficiente y veloz, pero a la vez inservible si la calidad de pares pregunta-respuesta es insuficiente.

- *Calidad técnica:* se refiere a la capacidad del modelo para generar pares de preguntas y respuestas relevantes, precisas, veraces y con suficiente profundidad como para representar toda la información del documento.

4.3.3. Arquitectura del benchmark

El benchmark técnico diseñado para esta tarea, se realiza sobre sobre una muestra de 25 párrafos provenientes de documentos científicos reales que se utilizaron como parte del corpus documental final, y fueron seleccionados tratando de alcanzar temas relativamente distantes a pesar de que versaran sobre la temática del proyecto estos 25 párrafos además provienen estrictamente de 5 documentos diferentes. Cada modelo generó pares pregunta-respuesta bajo condiciones controladas en Google Colab Pro (GPU A100).

Además, el prompt utilizado fue uniforme en todos los modelos de forma que las entradas eran uniformes e inalteradas en cada iteración, con el objetivo de conseguir repetibilidad y objetividad en la evaluación. A pesar de ello y como es normal, la métrica del tiempo transcurrido varía ligeramente en cada iteración, pero no resulta una variación significativa ya que es constantemente inferior al 0,5% en cada chunk procesado. Esta ligera variación se debe principalmente a que la ejecución se realiza en un entorno compartido, aun así, para asegurar la precisión del proceso, se ha tomado los resultados de la media de 5 iteraciones completas del benchmarks sobre las mismas entradas.

El prompt utilizado en el benchmark es el que aparece a continuación:

```
Act as an expert assistant in academic language. Read the
following paragraph and generate as many question-and-answer
pairs as you consider useful to understand its content.



Paragraph:

""" {paragraph} """



Format:

Question: ...

Answer: ...

(If there are more questions, continue)



Begin:
```

Tabla 5. Prompt empleado en el benchmark de selección

Como se puede apreciar, el prompt empleado no cuenta con contexto en la entrada ya que el objetivo principal es el de medir el rendimiento de generación ante entradas lo más simples y uniformas posibles para luego escalarlo al proyecto global.

Por ejemplo, en la siguiente *Tabla 6* se muestra el resultado del benchmark en uno de los 25 párrafos, concretamente el *abstract* de [33] :

> *"We present the Stanford Question Answer-ing Dataset (SQuAD), a new reading comprehension dataset consisting of 100,000+ ques-tions posed by crowd workers on a set of Wikipedia articles, where the answer to each question is a segment of text from the cor-responding reading passage. We analyze the dataset to understand the types of reason-ing required to answer the questions, lean-ing heavily on dependency and constituency trees. We build a strong logistic regression model, which achieves an F1 score of 51.0%, a significant improvement over a simple baseline (20%). However, human performance (86.8%) is much higher, indicating that the dataset presents a good challenge problem for future research. The dataset is freely available at https://stanford-qa.com."*

En esta tabla únicamente se muestran los pares de pregunta-respuesta que ha generado el modelo, es decir, lo que se empleará para realizar la evaluación cualitativa:

Modelo	Pares Q&A
Qwen/Qwen2-7B-Instruct	Question: What is the Stanford Question Answering Dataset (SQuAD)? Answer: The Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset consisting of 100,000+ questions posed by crowd workers on a set of Wikipedia articles, where the answers are segments of text from the corresponding passages.
	Question: How are the questions in SQuAD related to the Wikipedia articles? Answer: Each question in SQuAD is related to a specific Wikipedia article, and the answer to each question is a segment of text taken directly from the corresponding reading passage.
	Question: What does the analysis of the dataset reveal about the types of reasoning needed to answer the questions? Answer: The analysis of the dataset indicates the types of reasoning required to answer the questions, which heavily relies on dependency and constituency trees.
	Question: What is the performance of the logistic regression model built for answering questions in SQuAD? Answer: The logistic regression model achieved an F1 score of 51.0%, showing a significant improvement over a simple baseline (20%).
	Question: How does human performance compare to the model's performance on SQuAD? Answer: Human performance on SQuAD is much higher than the model's performance, with a score of 86.8%, suggesting that the dataset poses a challenging problem for future research.
	Question: Where can one access the Stanford Question Answering Dataset (SQuAD)? Answer: The Stanford Question Answering Dataset (SQuAD) is freely available at https://stanford-qa.com.

Tabla 6. Resultados del modelo Qwen de los pares pregunta-respuesta frente a un chunk

Finalmente, cabe señalar que el parámetro de la temperatura ha sido configurado a 0 ya que se persigue diseñar un proceso que de por resultado un dataset sintético y por ende se quieren obtener preguntas y respuestas lo más deterministas posibles sin dar lugar al modelo a alucinar en el proceso de inferencia.

4.3.4. Resultados y discusión

Los modelos que se evalúan en el benchmark son los siguientes:

- Qwen2-7B-Instruct
- Mistral-7B-Instruct-v0.2
- Llama-2-7b-chat-hf (a pesar de no ser instruct y estar ajustado a diálogo, funcionalmente opera de forma similar al instruct)
- GPT-J-6B
- *Mixtral 8x7B * se encuentra en la* Figura 20 *de este documento al explicar MoE*

Se han seleccionado estos modelos para tener suficiente variedad, ya que, evaluándolos, se cubre con suficiente amplitud una ventana temporal de 4 años y también de autores. Además, se ha contemplado la inclusión de modelos con diferentes estilos de entrenamientos, como se puede apreciar en la *Tabla 7,* pero manteniendo tamaños de modelos siempre comparables (6-7B de parámetros). A pesar de que también se hicieron iteraciones posteriormente con modelos 13B, el rendimiento que ofrecían era prácticamente igual a modelos más pequeños y por ende se decidió mantener los modelos en rangos más bajos de tamaño también para favorecer la repetibilidad en entornos locales de baja capacidad computacional.

Modelo	Año	Autor	Entrenamiento
Qwen2-7B-Instruct	2024	Alibaba	Instruct-tuned
Mistral-7B-Instruct-v0.2	2023	Mistral AI	Instruct-tuned
Llama-2-7b-chat-hf	2023	Meta	Chat-tuned
GPT-J-6B	2021	EleutherAI	Base Autoregressive

Tabla 7. Modelos considerados para el benchmark

En primer lugar, se presentan los resultados del tiempo medio de cada modelo en realizar la inferencia sobre las 25 entradas del benchmark. Esta métrica es particularmente interesante ya que el proyecto se realiza con la visión de escalabilidad siempre presente y pequeñas diferencias en este parámetro pueden eventualmente resultar en grandes costes computacionales.

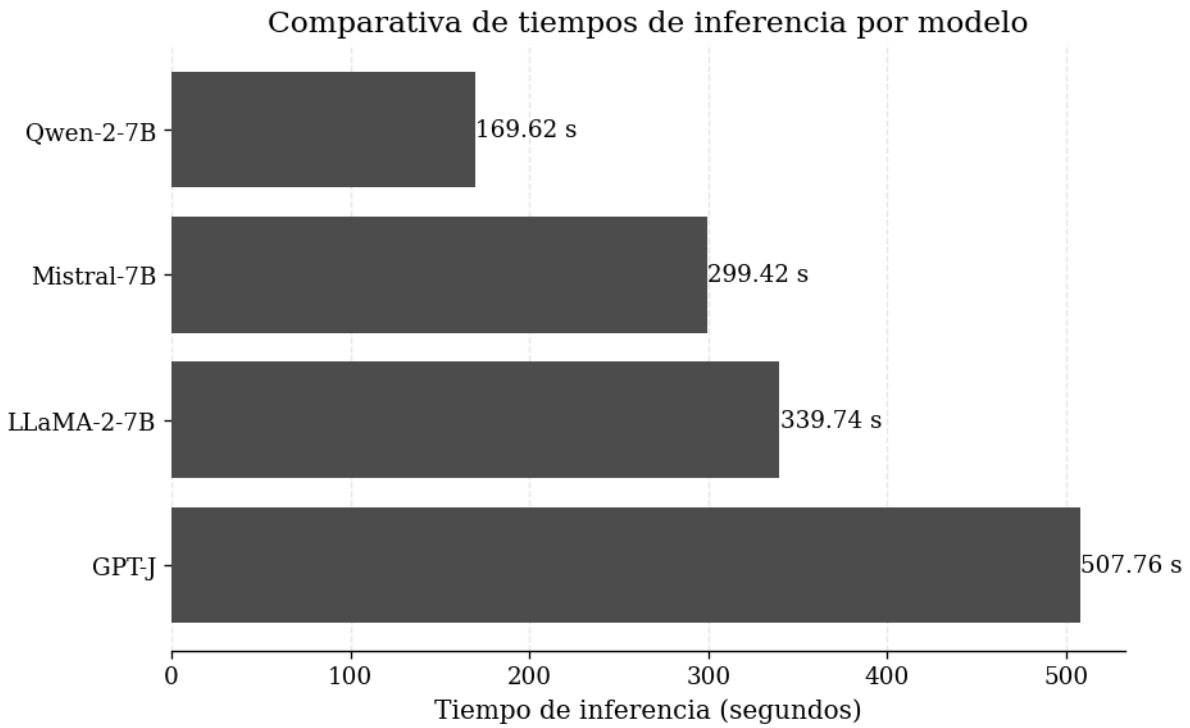

Figura 16. Tiempos de inferencia de los modelos en el benchmark de selección

De este primer indicador se deben destacar varios aspectos relevantes, siempre teniendo en cuenta los resultados cualitativos de los pares pregunta-respuesta. En primer lugar, el pobre rendimiento de GPT-J en este aspecto, porque no solo ha sido el modelo que con diferencia ha tardado más en realizar la inferencia de las 25 entradas del benchmark, sino que además la mayor parte de las salidas están vacías o son alucinaciones del modelo. Lo más probable es que se debe a una pobre adaptación del prompt a la arquitectura del modelo y no tan solo a que el modelo sea más pequeño que el resto (el GPT-J utilizado únicamente dispone de 6B de parámetros). Por otro lado, entre los modelos de Llama y Mistral no hay demasiada diferencia en cuanto a tiempo de ejecución ni en calidad de las salidas generadas, pero al compararlo con el modelo Qwen sí que se observa un salto sustancial en ambos campos. No solo es el modelo que más rápido ha ejecutado la inferencia, sino que además ha proporcionado pares de pregunta-respuesta estructurados y completos cubriendo gran parte de la información del texto y refiriéndose a él de forma concreta y no genérica como se aprecia en la *Tabla 6* .

Es interesante observar como el modelo Qwen es el único que se refería a la información por su nombre y sin contar con un contexto definido. Por ejemplo, tomando como referencia el párrafo anteriormente utilizado de [33] , si miramos a la pregunta que han generado los tres modelos a cerca del rendimiento del modelo de regresión para contestar preguntas sobre el dataset SQuAD observamos este patrón que se ha mencionado:

Modelo	Par pregunta-respuesta
Qwen2-7B-Instruct	Question: What is the performance of the logistic regression model built for answering questions in SQuAD? Answer: The logistic regression model achieved an F1 score of 51.0%, showing a significant improvement over a simple baseline (20%).
Mistral-7B-Instruct-v0.2	Question: What is the F1 score of the logistic regression model built for the dataset? Answer: The logistic regression model achieved an F1 score of 51.0%.
Llama-2-7b-chat-hf	Question: What was the result of the logistic regression model built by the authors? Answer: The model achieved an F1 score of 51.0%, which is a significant improvement over a simple baseline (20%).

Tabla 8. Par pregunta-respuesta común al mismo chunk en los tres modelos

El único modelo que sin contexto en la entrada ha sido capaz de realizar una pregunta clara y contextualizada ha sido el Qwen consistentemente a lo largo de todo el benchmark, indicando así que la calidad técnica para esta aplicación es en principio superior a los otros modelos de similares características. Ya que los otros dos modelos hacen referencias genéricas como "*the authors*" o "*the dataset*" lo que puede generar confusiones a posteriori al emplear el dataset para realizar el ajuste fino de otro modelo.

Por otro lado, gran parte de la diferencia en los tiempos de inferencia se debe a la diferencia de tokens generados en la salida por cada modelo. Es precisamente ahí dónde reside la conclusión del benchmark, para cubrir la información de una misma entrada, el modelo Qwen necesita generar menos pares de pregunta-respuesta ya que al ser estos de mayor calidad técnica y precisión, no necesita continuar con la generación para cubrir la información en casi su totalidad. Finalmente, en términos de eficiencia computacional esto supone una gran brecha entre modelos, ya que como se puede observar a continuación, la generación de tokens está estrechamente relacionada con la cantidad de FLOPs, que como se ha explicado con anterioridad es la medida objetiva que refleja el consumo computacional. Concretamente el modelo de regresión lineal es capaz de explicar la variabilidad de los FLOPs a partir de los tokens generados en un 90,5%.

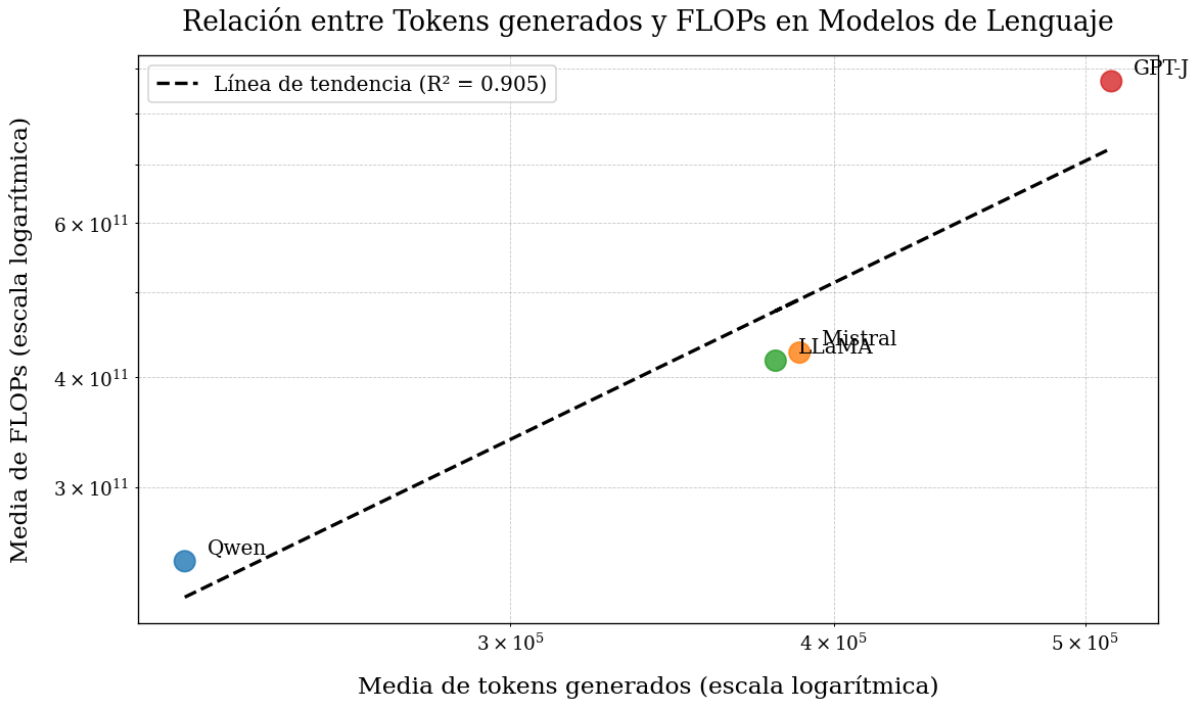

Figura 17. Relación entre Toknes generados y FLOPs de los modelos evaluados en el benchmark

Como resultado del diseño y aplicación del benchmark se ha optado finalmente por hacer uso del modelo Qwen ya que ofrece el mejor rendimiento y eficiencia respecto a los demás, por lo menos en las configuraciones de cuantización y parámetros que se han empleado. A continuación, se presenta la tabla de resultados obtenida del benchmark, como la media de todas las entradas, a excepción del tiempo, que representa el total de la ejecución:

Qwen	Tiempo	169,63 s
	Tokens	224,619
	FLOPs	2,47E+11
Mistral	Tiempo	299,42 s
	Tokens	387,90
	FLOPs	4,27E+11
Llama	Tiempo	339,74
	Tokens	379,8
	FLOPs	4,18E+11
GPT-J	Tiempo	507,48 s
	Tokens	511,76
	FLOPs	8,70E+11

Tabla 9. Resultado de la aplicación del benchmark

4.3.5. Optimización para el uso de las Unidades de computación

Uno de los aspectos más importantes al realizar procesos de inferencia o entrenamiento con grandes modelos de lenguaje es la cantidad de memoria de vídeo, conocida como VRAM, disponible en la GPU del ordenador. Pues es precisamente la VRAM la que delimita la capacidad del sistema para cargar y soportar el modelo además del tamaño del contexto que se puede procesar.

Si la VRAM con la que se trabaja tiene una capacidad limitada se puede trabajar con estos modelos, pero no a su máxima precisión (16 o 32 bits), dando lugar a técnicas de optimización como puede ser la cuantización. Este técnica permite acceder a los pesos de todos los parámetros, pero con una precisión notablemente menor, por ejemplo, en lugar de representar los pesos con una precisión de 32 bits se compactan a 4 bits y se reduce el consumo de memoria facilitando la ejecución en entornos más accesibles. A pesar de la efectividad de esta técnica para reducir el consumo de memoria, debe realizarse con sumo cuidado, pues puede afectar a la calidad de las salidas generadas y acabar generando alucinaciones al modelo.

Para seleccionar la mejor configuración con la que realizar el proyecto, se decidió evaluar al modelo en diferentes configuraciones de precisión: precisión completa de 16 bits (FP16), cuantizado a 8 bits y cuantizado a 4 bits. Cada configuración sería evaluada en los dos aspectos relevantes en el contexto del proyecto y del hardware utilizado, que son el tiempo de inferencia y el uso de memoria. Primero de todo se carga el modelo desde la librería de Transformers y posteriormente, se aplica la cuantización. En el caso de la cuantización más agresiva (4-bit) se aplica también una cuantización doble para preservar la precisión y conseguir una mínima pérdida semántica. Este tipo de técnicas no son necesarias en la cuantización a 8-bit ya que el riesgo de degradación es mucho menor y es una cuantización estandarizada en este modelo. A continuación, se presenta una imagen ilustrativa del proceso de cuantización, donde se pasa de una representación de 32 bit a una de 4 bit donde cada valor tiene su equivalente en una representación binaria de 4 dígitos:

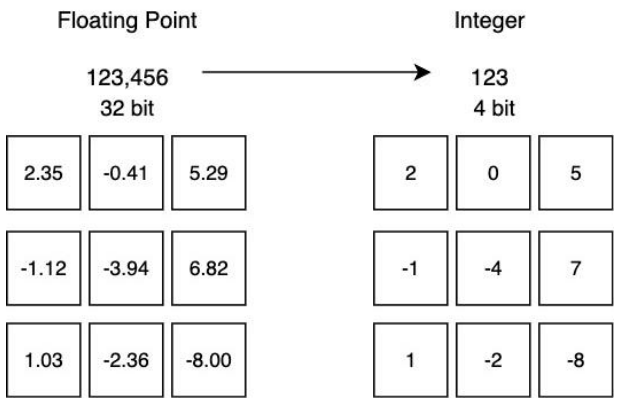

Figura 18. Representación del proceso de cuantización de 32 bit a 4 bit

Una vez cargados los modelos se procede a medir el tiempo total de inferencia con un prompt de generación sencillo y limitando la cantidad de tokens generados a 50, en este caso se ha empleado el siguiente prompt:

"Explain the importance of language models in modern AI systems."

Por otro lado, para medir la cantidad de uso de memoria de la GPU tras la carga y ejecución del modelo, se emplea *torch.cuda.empty_cache(),* que se transforma a GB para tener una representación más sencilla. Para asegurar que las siguientes mediciones no estén contaminadas por residuos de memoria ocupada de esta forma se mantiene la validez experimental del benchmarking.

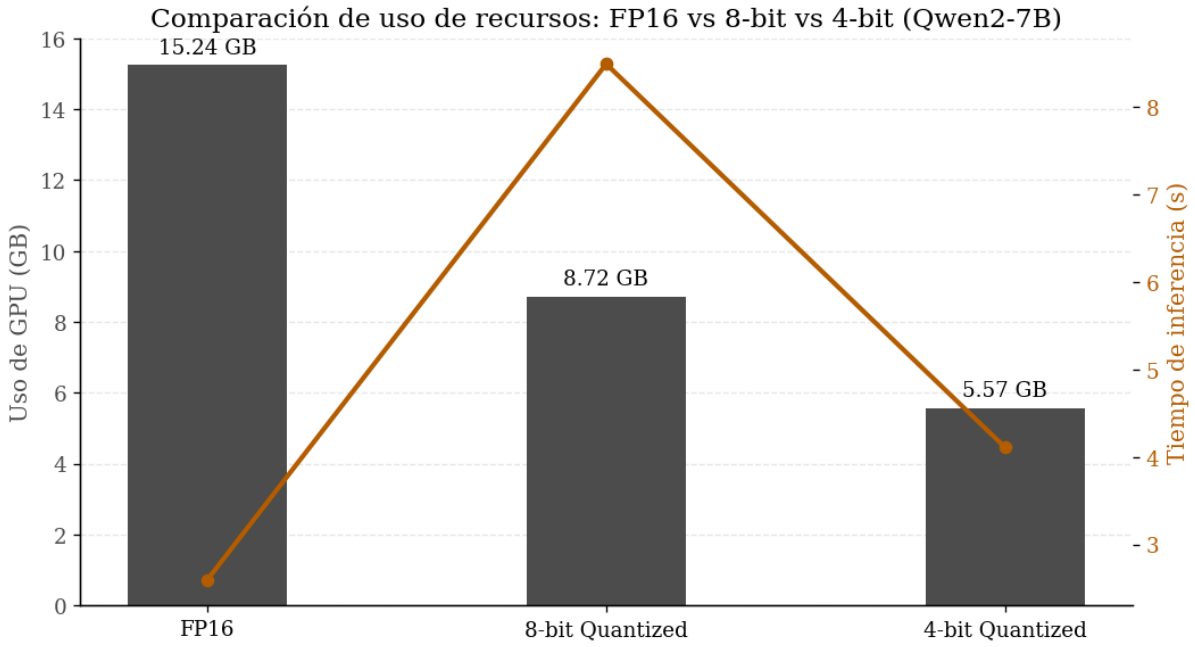

Figura 19. Comparación del uso de recursos en diferentes niveles de configuración de Qwen 2-7B

Los resultados obtenidos tras la aplicación del benchmark son realmente interesantes en cuanto se comparan entre sí, ya que, por un lado, el consumo de GPU sí que se reduce gradualmente como era de esperar al bajar la precisión de los pesos de los parámetros, tal y como se puede comprobar teóricamente. Siendo P el número de parámetros (invariante en los 3 casos en 7 Billones) y b el número de bits utilizados para almacenar los pesos de los parámetros:

$$GPU\ (GB) = \frac{P \cdot b}{8 \cdot 10^9}$$

En FP16, que es la configuración inicial del modelo:

〉 $GPU\ (GB) = \frac{7 \cdot 10^9 \cdot 16}{8 \cdot 10^9} \approx 14\ GB$

En 8-bit y sin cuantización doble:

〉 $GPU\ (GB) = \frac{7 \cdot 10^9 \cdot 8}{8 \cdot 10^9} \approx 7\ GB$

En 4-bit y con cuantización doble:

〉 $GPU\ (GB) = \frac{7 \cdot 10^9 \cdot 4}{8 \cdot 10^9} \approx 3{,}5\ GB$

Aunque la estimación teórica no se corresponde enteramente con los resultados empíricos, esto se debe a que durante la ejecución no solo se almacenan los pesos de los parámetros, también debe tenerse en cuenta los tensores intermedios (que suelen operar en float16 y no están cuantizados), buffers temporales y operaciones de cuantización. Cabe destacar que la mayor diferencia entre el valor teórico y el empírico obtenido se da en la cuantización 4-bit ya que no se almacenan escalas de cuantización de primer nivel, sino que también de segundo nivel debido a la acción doble.

Finalmente, atendiendo a los recursos computacionales con los que se inició el proyecto, se optó por seleccionar la configuración 4-bit, ya que el consumo de GPU se reducía a casi una tercera parte del inicial (35% de media en las primeras iteraciones con documentos), y el tiempo de inferencia a pesar de que se duplica, no es una desventaja notable en el contexto del proyecto.

```
# Configuración de la cuantización
quantization_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_use_double_quant=True,
    bnb_4bit_compute_dtype=torch.bfloat16,
)
```

Tabla 10. Configuración de la cuantización 4-bit para Qwen2-7b

Finalmente, a pesar de que se optó por emplear el modelo Qwen2 cuantizado para realizar la inferencia, también se tuvo en consideración el Mixtral 8x7B [49] que emplea MoE *(Mixture of Experts)*. Esta estrategia permite a los modelos incrementar sus prestaciones a la hora de realizar inferencia, pero sin un aumento lineal del coste computacional. A diferencia de los

modelos que se han presentado con anterioridad en este proyecto, Mixtral 8x7B y los modelos que emplean MoE, no activan todos los pesos de todos los parámetros para realizar la inferencia, sino que en función de la entrada del modelo se activan dinámicamente un subconjunto de parámetros específico. Esta acción limita considerablemente el consumo de VRAM, pues la información no ha de ser procesada por tantos nodos dentro de la red.

En arquitecturas MoE, cada "experto" o instancia (Mixtral 8x7B está formado por 8 expertos o instancias de 7 Billones de parámetros cada uno) normalmente se corresponde con una subred *feed-forward* independiente de las demás y por ende invisible a ellas. Cada una de estas subredes se activa y desactiva mediante un mecanismo de enrutamiento (*gating mechanism).* Este mecanismo determina los expertos más relevantes para la entrada proporcionada, y puede ser configurado [64] . Por ejemplo, la configuración *top-2* empleada con Mixtral 8x7B implica la activación de 2 de los 8 expertos disponibles, tal y como se observa en la figura que aparece a continuación, representando la capa de MoE en el Mixtral 8x7B:

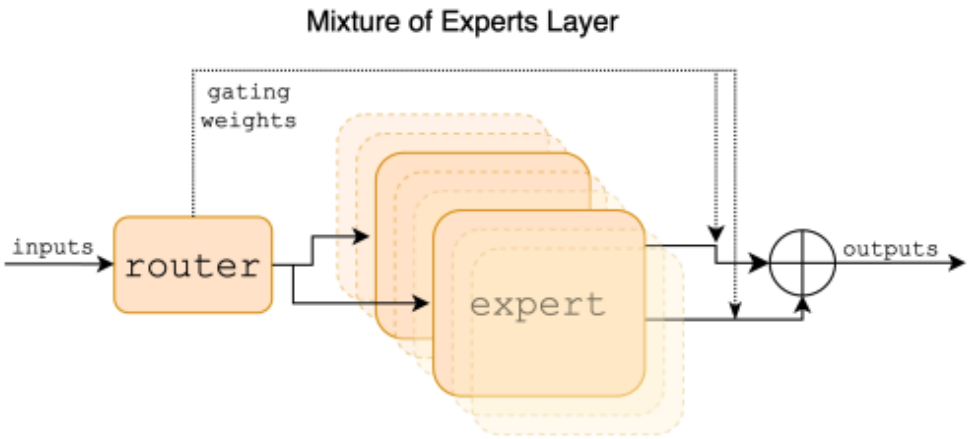

Figura 20. Capa de MoE en Mixtral 8x7B. Fuente:[66]

Se debe destacar que este enfoque presenta numerosas ventajas frente a las arquitecturas tradicionales, ya que permite escalar eficientemente la capacidad de los modelos permitiendo a los diferentes expertos que lo forman aprender tareas distintas y especializarse en ellas. A pesar de ello, también presentan algunos desafíos como el desequilibrio entre expertos (puede dar lugar a *overfitting* en expertos poco utilizados) y la difícil implementación de los modelos [65] .

No obstante, a pesar de las ventajas que pueda presentar, tras evaluarlo en el benchmark de selección frente a la configuración seleccionada previamente de Qwen2, se observa como el tiempo de inferencia se duplica y la cantidad de memoria utilizada aumenta de 8,94 GB hasta 31,25 GB, una capacidad que ya exigiría una unidad de computación avanzada que no está al

alcance de la mayoría en sus ordenadores de diario. Por ello, a pesar de que se utilizó el modelo Mixtral8x7B para iterar sobre la calidad de los pares pregunta-respuesta (la calidad de la generación es notablemente superior) no se empleó para desarrollar el dataset al completo.

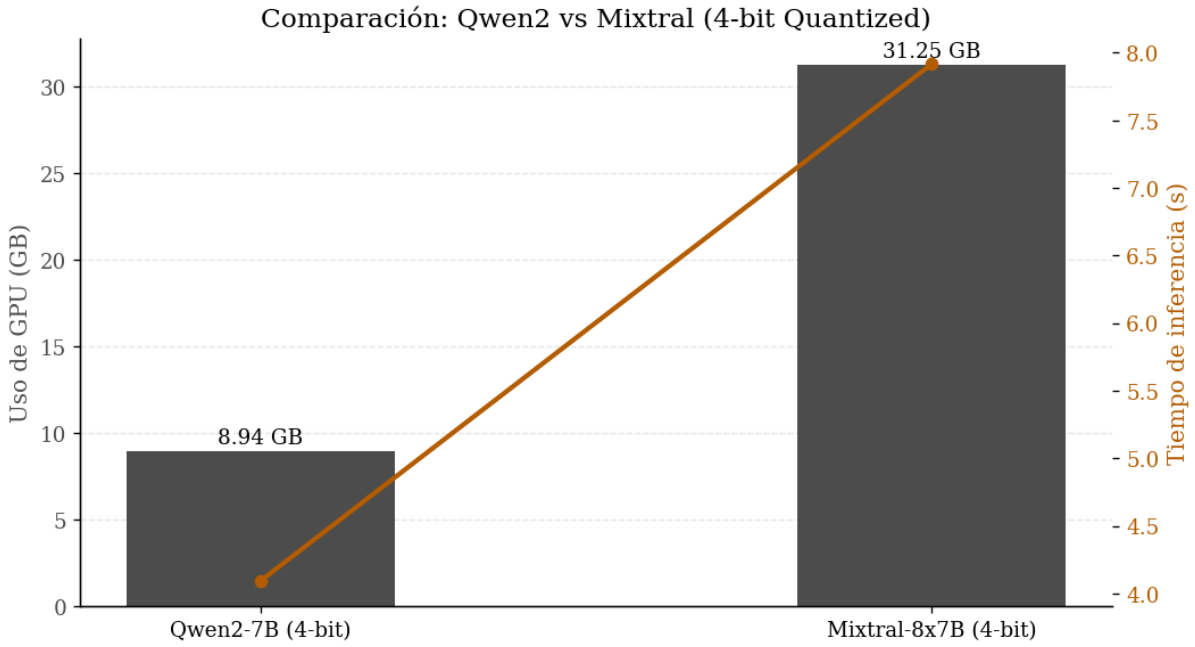

Figura 21. Comparación de Qwen2 y Mixtral8x7b en el benchmark de selección

4.4. Técnicas utilizadas en la Generación del Dataset

4.4.1. División de la información en Chunks

Como se explica en la sección 4.1.2 de este documento, debido a la peculiar naturaleza del corpus documental, se han seleccionado los documentos que se incluyen en las referencias del proyecto y que están en la base de datos de arXiv para generar el dataset.

En las tareas relacionadas con el PLN como la que se lleva a cabo en este proyecto a la hora de generar preguntas y respuestas, es una práctica habitual la de dividir en fragmentos manejables la información. En primer lugar, para poder procesar la información de los documentos seleccionados que en primera instancia están en formato PDF, se deben pasar a texto limpio y segmentado. En este proceso, los fragmentos manejables obtenidos se conocen como chunks e influyen directamente en la calidad de las salidas generadas. Estos chunks pueden obtenerse de varias maneras:

- *Segmentación por longitud fija*: Consiste en la división del texto en cadenas de longitud fija. Estas cadenas pueden ser de caracteres, palabras o tokens. Esta técnica es comúnmente empleada en sistemas automáticos por su simplicidad y

determinismo, ya que es una técnica cuyo resultado es completamente predecible. No obstante, la rigidez de su aplicación facilita la incoherencia semántica [21] .

- *Segmentación estructural*: Se basa en la estructura lógica del texto en su totalidad, empleando delimitadores como párrafos, secciones, títulos o cualquier otro elemento repetitivo que permita diferenciar las partes de un documento. A diferencia del anterior, este enfoque prima la coherencia y consistencia contextual y es más adecuado en tareas que requieren una comprensión semántica más compleja. A pesar de ello, presenta fragmentos de longitud variable que deben ser gestionados para no comprometer las entradas de los modelos [11] .
- *Segmentación por límites sintácticos:* De forma similar a la segmentación estructural, emplea delimitadores de puntuación o de análisis gramatical (como, por ejemplo, oraciones). Esta técnica prioriza mantener las unidades lingüísticas intactas a pesar de que implica un procesamiento posterior que en algunos casos supone realizar análisis sintáctico [62] .
- *Ventanas deslizantes (sliding windows):* Este enfoque sugiere la división del texto en bloques de longitud fija, pero con solapamiento entre ellos para evitar la pérdida de contexto e información entre chunks. A pesar de que mejora la consistencia semántica y minimiza el postproceso, aumenta considerablemente el coste computacional[63] .

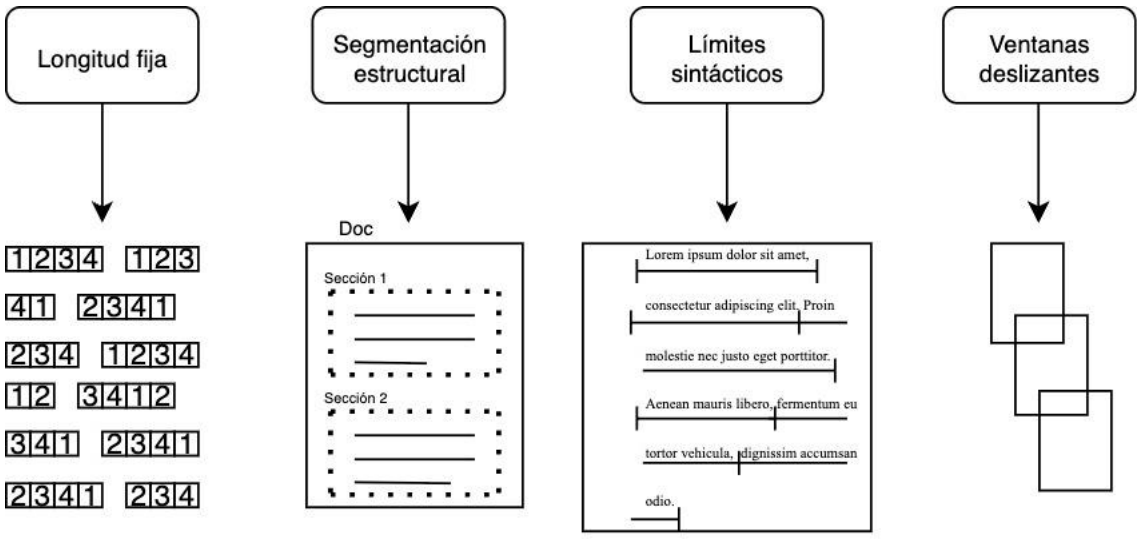

Figura 22. Diferentes tipos de segmentación textual

Para realizar esta división de la información en chunks se deben tener en cuenta varios aspectos:

- *Cantidad máxima de tokens de entrada*: los LLMs tienen predefinida una cantidad limitada de tokens por cada proceso de inferencia que se realiza. Al segmentar el texto

en varios chunks se garantiza que cada fragmento se pueda procesar en una sola ventana de contexto sin perder información. De todos modos, es importante conocer la cantidad máxima que soporta el modelo, ya que, si suponemos una cantidad ilimitada o exageradamente grande de tokens, resultaría más sencillo y lógico hacer inferencia en cada ventana de contexto con los documentos completos.

No deben confundirse los tokens que puede procesar el modelo con los tokens que soporta el tokenizador. Por ejemplo, en el caso del Qwen2, se pueden procesar 32,768 tokens en cada proceso de inferencia, tal y como aparece en la Tabla 1 y estos tokens son la combinación de los empleados en la entrada más los generados en la salida. En cambio, el tokenizador cuenta en su vocabulario con 151,643 tokens diferentes que puede emplear para codificar cualquier palabra como combinación de algunas de las 15,643 componentes.

- *Estructura original del texto*: la segmentación de texto depende profundamente del tipo de información con la que se trata. Por ejemplo, si se trabaja con la API de Wikipedia se puede obtener la información de todas las páginas de Wikipedia en un formato JSON y con secciones completamente limpio y organizado. En cambio, por ejemplo, al segmentar libros antiguos que han sido escaneados, se debe tratar la información para discernir la información en texto y la información en imágenes para posteriormente tratarlas cada una por separado y como es debido.
- *Ruido:* durante de la segmentación del texto es de vital relevancia limpiar los textos antes de que estos sean procesados, ya que la división automatizada de documentos de toda índole, por muy robusto que sea el proceso, siempre se van a generar particularidades que se salen de la norma, como pueden ser párrafos excesivamente cortos, pies de página, referencias, etc.

Finalmente, se optó por realizar la división en chunks por párrafos atendiendo a las ventajas que esto presentaba:

- *Coherencia semántica:* a diferencia de otras unidades contextuales como pueden ser las frases aisladas, los párrafos suelen ser desarrollos argumentativos cerrados y engloban información de principio a fin alrededor de un tema concreto o de parte de él
- *Tamaño manejable:* por lo general los párrafos por sí solos no exceden los límites de tokens de los modelos y esto permite tener cierta flexibilidad en el prompt y más aún si se pretende incluir también texto en forma de contexto.
- *Compatibilidad con tareas de Q&A:* para conseguir generar suficientes preguntas y respuestas de forma que se cubra toda la información de la entrada al modelo, es necesario que el input sea denso informativamente a la vez que relativamente autónomo, lo suficiente como para poder auto contextualizarse.

〉 *Coste computacional:* a pesar de que el método de ventanas deslizantes también presentaba las características necesarias para la fragmentación del texto, el coste computacional que añadía era considerable, y teniendo en mente la escalabilidad del proyecto se ha optado finalmente por fragmentar el texto de forma que en el futuro no comprometa la creación de otros datasets.

Para llevar a cabo la segmentación primero de todo se examinó el modelo para corroborar que los tokens del prompt y de un párrafo podían ser procesados sin problema. En este caso, como ya se ha mencionado anteriormente, el modelo Qwen2 utilizado puede procesar 32,768 tokens, de forma que se deben conocer específicamente las longitudes de los chunks que se han creado. Después de la división, se observó que el chunk más extenso en cuanto a número de tokens, ocupa 1,035 tokens. Además, se deben sumar los tokens que ocupa el propio prompt, que en este caso finalmente fueron 73. Por último, al incluirse un chunk de contexto (explicado en detalle en el apartado 4.4.4 de este documento) en la entrada de la inferencia del modelo, también debe de tenerse en cuenta su longitud, que por seguridad y para garantizar holgura se considera que el chunk más extenso que se incluye en el contexto puede ser como máximo igual al chunk más extenso después de la división de chunks, es decir, 1,035 tokens. Estas partes que forman la entrada finalmente ocuparán un máximo de 2,143 tokens, dejando espacio para la generación de al menos 30,625 tokens, los cuales son muchos más de los necesarios para el contexto del proyecto.

A continuación, se presentan los pasos seguidos para la división del texto en fragmentos. Cabe destacar que dicha división ha sido realizada y organizada en el código como un archivo .py independiente para facilitar el proceso de actualización de documentos, además

4.4.2. Inferencia

El modelo utilizado durante la generación del dataset, el Qwen2-7B-Instruct, fue cargado en su configuración en 4 bits, tal y como se detalla en el apartado 4.3.5 de este documento. La cuantización siempre está estrechamente relacionada con el rendimiento del modelo y más en aquellos casos en los que el proceso de inferencia abarca la parte más delicada del proceso.

Para ejecutar la generación de forma eficiente y modular, se empleó el pipeline *text-generation* de Hugging Face, que permite realizar inferencia causal a partir de un prompt estructurado . Sin embargo, todas estas características y configuraciones de ejecución deben de ser entendidas en el contexto de la versión que haya sido utilizada, pues el rendimiento en una determinada tarea no depende únicamente de la precisión de los pesos de los parámetros, sino también depende del tipo de ajuste que haya recibido el modelo con anterioridad.

En este caso de uso, se ha empleado la versión Instruct del modelo, es decir, la versión ajustada (por técnicas de fine-tuning supervisado) para seguir instrucciones específicas del usuario. Es decir, este tipo de versiones permiten obtener salidas controladas y definidas a partir de un prompt sea o no contextualizado. La principal diferencia de este tipo de modelos frente a los modelos "base" o los modelos adaptados a estructuras de chatbots, radica en que las versiones instruct priorizan la precisión y mantener una estructura coherente de salida ciñéndose a los requerimientos del usuario.

Este tipo de ajustes tienen una implicación directa tanto en la calidad como en la flexibilidad de la generación del dataset, ya que el prompt puede ser tan cerrado como el usuario quiera definirlo. Además, mediante un proceso iterativo se puede encaminar la salida del modelo hacia el objetivo final, evitándose así salidas ambiguas e inconsistentes en el largo plazo. De este modo, la utilización de los modelos de lenguaje en su versión instruct, no únicamente mejoran la coherencia de los pares pregunta-respuesta, sino que también favorecen la trazabilidad del proceso de generación, una de las tareas más desafiantes al trabajar con LLMs. A continuación, se presenta un diagrama ilustrativo de algunos de los ajustes más comunes en los LLMs [79] :

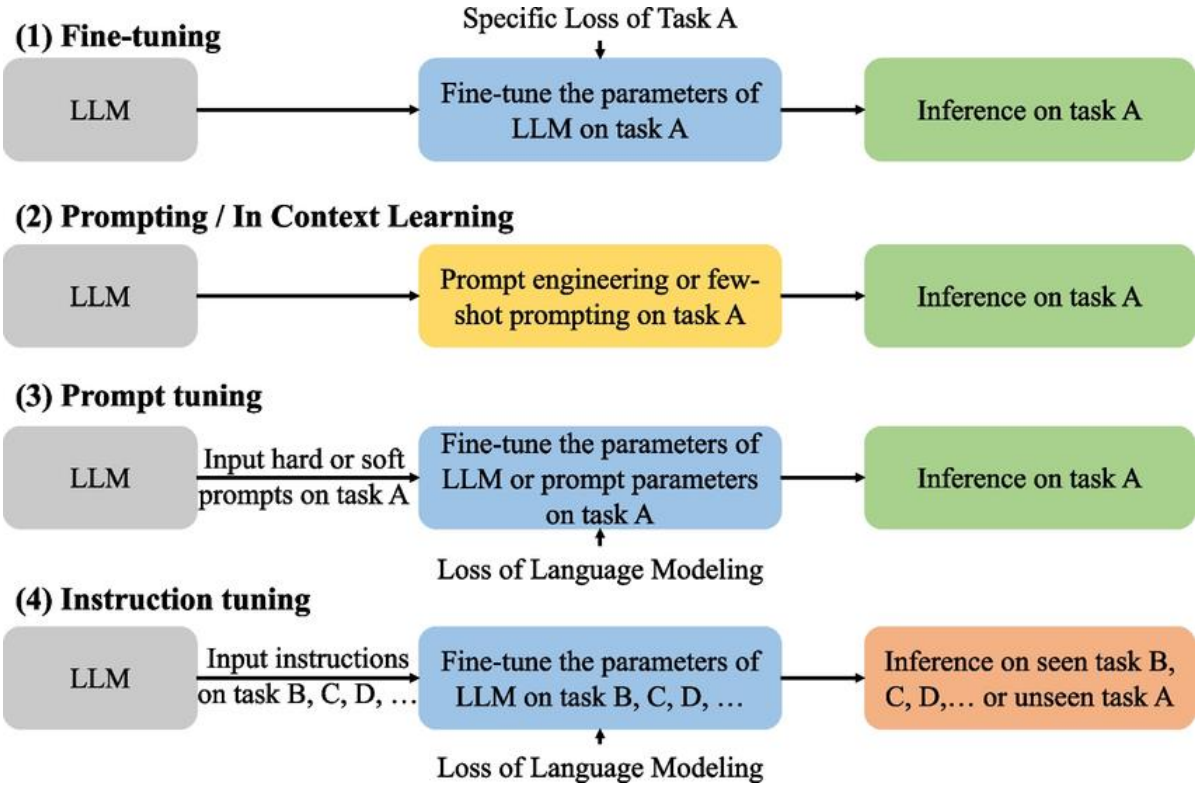

Figura 23. Explicación detallada de cinco métodos diferentes de entrenamiento (adaptación de dominio) para recomendaciones basadas en modelos de lenguaje grande (LLM). Fuente: [79]

4.4.3. Ingeniería del Prompt

El factor clave del proyecto para conseguir generar pares de pregunta-respuesta coherentes, relevantes y sobre todo basadas en el texto del corpus documental, es el prompt utilizado como input del modelo. Este proceso de refinamiento conocido como *Ingeniería del Prompt,* fue la etapa más iterativa y sensible del proyecto, ya que los LLM funcionan como un mecanismo de pesos encadenando operaciones secuenciales de uno a otro para obtener el output, pero este proceso es una caja negra que hasta el momento no se ha conseguido descifrar. Esto significa que pequeñas variaciones en el prompt de entrada, se traducen en grandes cambios del output. Además, este comportamiento se ve fuertemente condicionado por la configuración del modelo y por la versión que se esté utilizando (en este caso la versión *instruct*).

Las iteraciones se realizaron basándose en dos pilares, el primero siendo las recomendaciones formuladas por Greg Brockman mencionadas anteriormente en el apartado 3.1.5 de este documento, y en segundo lugar los principios del aprendizaje por refuerzo con retroalimentación humana (*RLHF*). A pesar de que no se aplicó RLHF de forma directa, sí que se estableció un protocolo a través del cual se realizaban pequeñas modificaciones del prompt en busca de una mejora basando estas modificaciones en la observación manual de los outputs generados, ya que sí que se tenía una clara idea del output objetivo.

A lo largo del proyecto se mantuvo un historial de versiones de *prompts* probados, que permitió entender cómo distintas variaciones afectaban la precisión, la completitud o la naturalidad de las preguntas generadas. Este registro fue fundamental para identificar *hallazgos empíricos*, como, por ejemplo:

- La necesidad de especificar la longitud o formato de las respuestas.
- La importancia de indicar que las preguntas debían basarse exclusivamente en el contexto del chunk.
- El uso de ejemplos explícitos en el prompt para guiar el comportamiento del modelo (pocos-shot prompting).
- La supresión de ambigüedad mediante instrucciones negativas (por ejemplo, “no inventes información fuera del contexto”).

Finalmente, se adoptó una versión optimizada del prompt que maximizó la relevancia de las preguntas y minimizó la generación de respuestas incorrectas o genéricas. Este prompt final fue utilizado en el proceso de inferencia a gran escala con el modelo Qwen-2.

4.4.4. Gestión del json

La cuenta de Google Colab Pro, tiene un límite temporal en función del uso de recursos que se esté haciendo. Para este caso de uso se trata de un consumo inusualmente elevado ya que realizar inferencia con grandes modelos de lenguaje es sumamente costoso. Esto presenta un gran reto a la hora de generar un dataset extenso, ya que si la ejecución se ve interrumpida no es posible saber en qué punto de la generación se ha quedado el modelo.

Por otro lado, también se debe tener en cuenta el uso de GPU ya que si se utiliza por encima de su capacidad y no se libera espacio después de cada iteración puede dar lugar al error más común en los proyectos de NLP: *CUDA out of memory.*

Para subsanar estas dificultades se optó por una técnica de subida continua de la información. Cada vez que el flujo de trabajo finaliza la ejecución de un lote de archivos, accede al dataset de Hugging Face por medio del token de autenticación (véase el apartado 4.4.9 de este documento) y se descarga el archivo para posteriormente concatenar la información nueva que ha generado y volver a subir el JSON al dataset de origen. De esta forma, cuando se produce un error (ya sea por una conexión inestable o un consumo excesivo de recursos), no se pierde la evolución de la información generada y se tiene una trazabilidad total del punto en el que se encuentra el modelo en todo momento.

4.4.5. Depuración del conjunto de preguntas y respuestas

Una vez obtenido el JSON con todo el conjunto de pares pregunta-respuesta, se debe hacer una depuración del resultado. Esto decisión responde a la observación de que, a pesar de realizar numerables refinamientos e iteraciones del prompt (incluso a pesar de darle contexto al modelo) este tiende a referenciar información del texto que carece de significado si no se tiene el artículo en su totalidad. Principalmente, este problema aparece cuando el modelo procesa algún chunk en el que se menciona una figura del artículo, ya que, como se explica en el apartado 4.1.1 de este documento, al no utilizar VLMs, no se procesan las imágenes o figuras que no contengan información en formato de texto.

Cabe señalar que esto no significa que el output generado por el modelo sea incorrecto o impreciso, pero no se debe perder de vista que el proyecto tiene como objetivo final obtener un dataset que sea utilizable en un posterior proceso de fine-tuning. Es por ello por lo que estos pares pregunta-respuesta deben ser eliminados del dataset, para no comprometer la calidad del posterior proceso y evitar la dependencia de elementos visuales que no se han tenido en consideración.

A continuación, se puede observar uno de los ejemplos que ha sido depurados del dataset original donde en efecto se observa como la respuesta es precisa y coherente con el contexto

del cual se ha obtenido, pero como ya se ha mencionado, el dataset no puede tener una depender de elementos visuales que no van a ser vistos por el posterior modelo en el proceso de fine-tuning. Esta coherencia se debe a que el modelo contesta en base a la información de entrada, y aunque se referencia contenido visual, como en este caso la "*Figure 1.3*", también hay más información acerca de esa figura, que es precisamente la que emplea el modelo para la generación:


```
        "source": "2408.13296v3.pdf",

        "chunk": "Figure 1.3: Mind map depicting various dimensions of
Large Language Models (LLMs), covering aspects\nfrom pre-training and
fine-tuning methodologies to efficiency, evaluation, inference, and
application do-\nmains. Each dimension is linked to specific techniques,
challenges, and examples of models that exemplify\nthe discussed
characteristics. This diagram serves as an overview of the multifaceted
considerations in\nthe development and deployment of LLMs. (adapted from
[13])",

        "question": "What aspect of LLMs does the dimension of
\"efficiency\" in Figure 1.3 focus on?",

        "answer": "The dimension of \"efficiency\" in Figure 1.3 focuses
on aspects such as computational resources required, training time, and
model size."
```


Tabla 11. Ejemplo de pares pregunta-respuesta depuradas del dataset RAW

Para hacer esta limpieza, se empleó un proceso de filtrado automático en el cual se definió una lista de palabras clave que inhabilitaban la validez del output generado por el modelo, entre las que se incluyen: "*Figure*", "*Table*", "*Appendix*", "*Row*", "*Column*" e "*image*". Estas referencias fueron detectadas mediante expresiones regulares aplicadas a las preguntas generadas. De esta forma si la pregunta contenía alguna de estas palabras citadas en la lista, seguida de algún patrón de numeración, se descartaba el par y se almacenaba en un json. Esta última acción no tiene otro propósito que el de facilitar la depuración y asegurar la trazabilidad de la generación del dataset.

4.4.6. RAG con Mistral 7B

En el proceso de validación se ejecuta un modelo basado en *Retrieved Augmented generation* (RAG) para verificar que se cubre todo el espacio vectorial de embeddings y que realmente la información sale del mismo

Para ello en primer lugar se importa el token de HuggingFace de la misma manera que se había hecho en el proceso de generación, para poder acceder a los modelos de HuggingFace. Una vez configurado en entorno con las librerías seleccionadas (explicadas con detenimiento en el apartado 4.2.1 de este documento), se procede con la carga del modelo.

En este caso se ha seleccionado el Mistral-7B-Instruct (optimizado para GPU A100). Le elección de este modelo, al igual que en el proceso anterior está profundamente fundamentada, teniendo en consideración que se sigue valorando los mismo aspectos de eficiencia computacional y limitaciones de unidades de computación. Es decir, se pretendía encontrar el modelo que ofrecía el mayor equilibrio entre desempeño, eficiencia y flexibilidad de integración en el *workflow* del proyecto en su totalidad:

- MMLU (*Massive Multitask Language Understanding*): Mistral-7B-Instruct supera a Llama-2-13B y es comparable a Llama-2-70B en tareas de comprensión y razonamiento multitarea, lo que lo hace adecuado para validación de Q&A basada en contexto. Este último es relativamente importante ya que su desempeño se está equiparando al de un modelo con 10 veces más parámetros.

Para continuar con el proceso de validación se diseña un pipeline de generación de texto que permitirá validar las respuestas dado un contexto.

La parte más relevante del proceso de validación del dataset trata de la preparación de los embeddings que se van a utilizar para realizar la búsqueda contextual que habilita la asignación de cada par pregunta-respuesta con los chunks del corpus documental. Antes de realizar la vectorización de los chunks, se segmenta toda la información de exactamente la misma forma que en el proceso de generación, para asegurar que se está trabajando sobre los mismos datos.

A pesar de que se podría optar por otra segmentación, no se aseguraría trazabilidad, coherencia ni validez. Al replicar la segmentación se asegura que en caso de que la pregunta-respuesta se corresponda con alguno de los chunks utilizados durante la generación, este pueda ser encontrado durante el proceso de validación permitiéndose de esta forma comprobar de forma precisa que la información generada realmente está presente en el corpus y no es fruto de alucinaciones del modelo.

Además, como durante la validación también se pretende comprobar que se ha realizado una cobertura completa de la información, si se alterasen los chunks respecto a los obtenidos durante la generación, podría dar lugar a *falsos positivos* al considerar alguno de los nuevos chunks suficientemente similares a los originales como para concluir que las respuesta han sido obtenidas del corpus original.

Una vez segmentado el texto se debe elegir el modelo de embeddings que se va a utilizar. Para ello se hizo una selección de modelos según el benchmark MTEB [67] , explicado en el apartado 3.4.1 de este documento. Los modelos evaluados fueron los siguientes:

- all-MiniLM-L6-v2
- all-MiniLM-L12-v2
- all-mpnet-base-v2
- bge-m3
- e5-large-v2
- bge-large-en-v1.5

Ya se ha documentado en la literatura que el modelo BAAI/bge-large-en-v1.5 presenta un rendimiento destacado en tareas de recuperación semántica a nivel general y resulta especialmente adecuado para aplicaciones del tipo RAG (Retrieval-Augmented Generation).

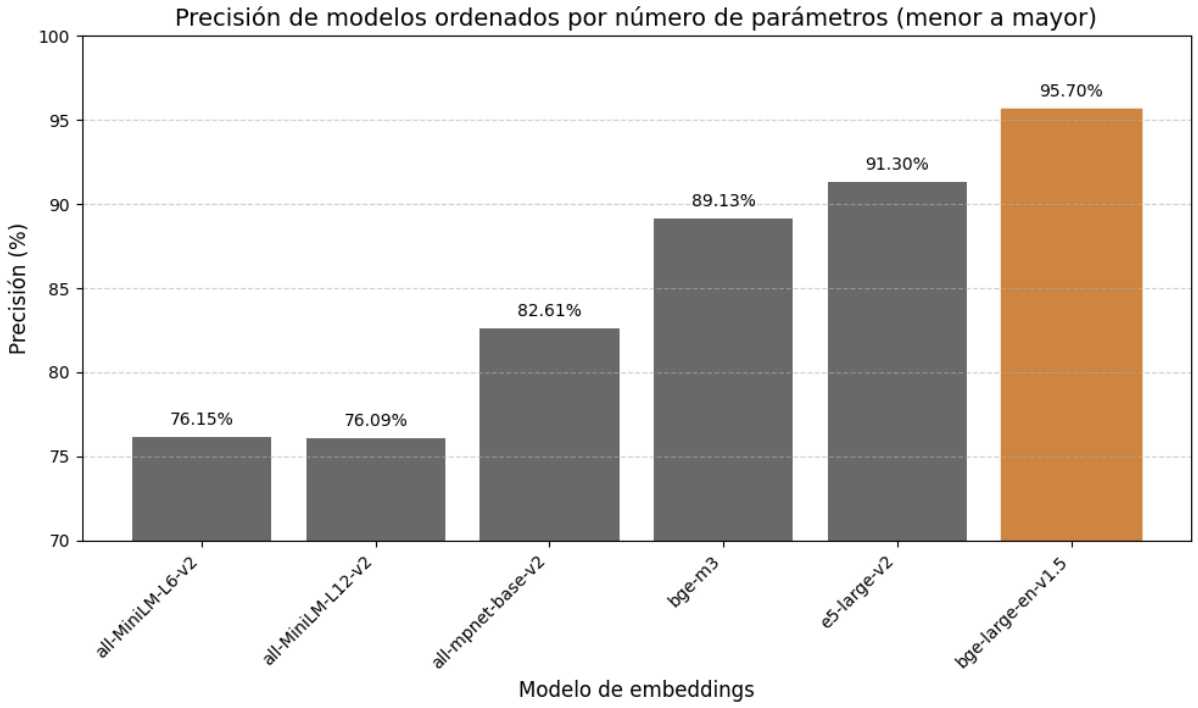

Figura 24. Comparación de los modelos de embeddings en el proceso de validación según la precisión obtenida

Después de realizar la validación con todos los modelos considerados se concluyó empíricamente que en efecto el modelo bge-large-en-v1.5 es el que mayor precisión alcanzó de todos. De esta forma se toman como válidos los resultados obtenidos tras la aplicación de este modelo ya que tanto de forma teórica como empírica se demuestra que es el que mayor rendimiento ofrece. Posiblemente su desempeño es notablemente superior debido a que se trata de un modelo Transformer encoder (tipo BERT) entrenado para tareas de recuperación semántica y optimizado para que las distancias coseno entre los embeddings en el espacio n-vectorial reflejen similitud semántica real, es decir, a mayor similitud entre dos chunks, menor será la distancia en el espacio vectorial de embeddings generado, lo cual permite una recuperación contextualizada del conocimiento relevante.

Esta evaluación se llevó a cabo generando representaciones vectoriales de los chunks procedentes del corpus documental y se generó un índice FAISS para así llevar a cabo la recuperación de información característica de los RAG. Después con el modelo Mistral-7B se midió la precisión con un criterio binario de "Yes" y "No", el retorno se considera afirmativo cuando existe coherencia entre la pregunta, la respuesta esperada y el contexto que se recupera de cada embedding. La Figura 24 muestra los resultados obtenidos, donde se aprecia que el modelo bge-large-en-v1.5 alcanzó una precisión del 95.70 %, superando ampliamente al resto de modelos evaluados.

Para ello se crea un *Vector store* que se utilizará para buscar el contexto más relevante de cada pregunta, como se ve a continuación:

```
# Creación del Vector Store

vector_store = FAISS.from_texts(chunks, embedding_model)


# Voctorización de la pregunta y búsqueda por similitud semántica

retrieved_contexts = vector_store.similarity_search(pregunta, k=3)
```

Tabla 12. Aplicación de FAISS y búsqueda de contexto

El Vector Store es una estructura optimizada para realizar búsquedas por similitud entre conjuntos representados en el espacio vectorial. En este caso se han tomado los 3 chunks más similares semánticamente, ya que el corpus documental verse alrededor del mismo tema y por ende es probable que varios chunks contengan información complementaria o redundante

dando lugar a imprecisiones en la búsqueda. Esta aproximación permite minimizar el efecto de información repetida entre artículos del corpus.

Finalmente, se guarda el resultado de la validación, el contexto recuperado y la explicación del modelo. Esto se realiza para poder guardar cierta trazabilidad de aquellas preguntas que no han sido consideradas como válidas a la hora de la validación. Véase en el ejemplo que aparece a continuación sobre una pregunta realizada:

```
        "question": "How does the Transformer architecture differ from
traditional recurrent or convolutional neural networks used in sequence
transduction models?",

        "expected_answer": "The Transformer architecture differs from
traditional recurrent or convolutional neural networks by relying solely
on attention mechanisms instead of using recurrence or convolutions. This
allows for greater parallelizability and significantly shorter training
times.",

        "retrieved_context": "[/INST]\n     Yes, the expected answer is
correct according to the context. The Transformer architecture presented
in the paper is different from traditional recurrent or convolutional
neural networks used in sequence transduction models because it dispenses
with recurrence and convolutions entirely, replacing them with multi-
headed self-attention mechanisms. This results in models that are more
parallelizable and require significantly less time to train compared to
complex recurrent or convolutional neural networks."
```

Tabla 13. Ejemplo del proceso de validación de los pares pregunta-respuesta

Una vez obtenidos todos los resultados del proceso de validación, se computan X métricas para analizar el dataset en su totalidad:

- Precisión (*accuracy*): se calcula como el cociente entre las respuestas válidas (categorizadas como "Yes" en el proceso de validación) sobre el número total de preguntas evaluadas.
- Cobertura de la información: cada vez que un chunk es accedido por FAISS durante la búsqueda de contexto relevante para responder, se almacena su índice. Una vez finalizada la validación se cuenta el número de chunks accedidos sobre el número total de chunks generados a partir del corpus.

4.4.7. Métricas de la validación

Esta última métrica resulta clave en el contexto del proyecto, ya que permite medir de forma objetiva y porcentual la cantidad de información que ha sido cubierta por el modelo durante la generación. Es importante asegurar que en el proceso se cubre toda la información del corpus ya que es uno de los requisitos del proyecto. También es importante analizar con prudencia el resultado ya que un 100% no debería ser correcto:

- *No toda la información es relevante:* Como se ha mencionado con anterioridad el corpus de documentos versa sobre una temática en particular, y debido al volumen de información algunos de los chunks contienen información redundante. Otros, en cambio, contienen información vacía o poco significativa ya que, a pesar de haber eliminado las referencias durante la fase de preparación de los datos, hay muchos otros casos particulares de similar índole.

- *Limitaciones del método de recuperación*: El sistema de búsqueda vectorial que se ha aplicado, pone por delante en el orden de prioridad aquellos fragmentos más similares a cada pregunta, si varias preguntas son similares siempre se acabará encontrando los chunks más relevantes y quedarán descartados otros que pueden tener menos información pero que para la pregunta evaluada en particular son perfectamente válidos, de esta forma no llegan nunca a ser accedidos.

- *Cobertura como indicador, no como objetivo absoluto:* A pesar de que el porcentaje de chunks accedidos es una métrica útil para identificar vacíos no cubiertos del corpus, esta debe ser interpretada junto con la relevancia y claridad de las preguntas para cerciorar que no se generan con el único objetivo final de cubrir toda la información.

4.4.8. Integración del Código de Generación

Una vez procesados individualmente todos los módulos previamente explicados, se desarrolló un script principal que integrase todos los componentes de la pipeline de forma modular y secuencial. El principal propósito de este enfoque se alinea con el objetivo que viene liderando el proyecto: facilitar la escalabilidad y el control de flujo de todo el sistema semiautomatizado. Este script tiene la función de representar el input del proceso y realiza las siguientes tareas:

- La instalación e importación automática de dependencias.

- La inicialización del modelo generativo Qwen2-7B-Instruct desde Hugging Face con cuantización en 4 bits mediante *BitsAndBytes*, para reducir el uso de memoria.

- La ejecución del preproceso de información mediante extracción y segmentación de texto desde documentos PDF (apoyado en un script modular importado al principal)

- La generación de pares de pregunta-respuesta (Q&A) usando un *prompt* estructurado y realizando inferencia con el modelo

- Gestión de escritura, carga y limpieza del JSON generado para adecuarse al formato de ajuste fino y de validación.

Una de las principales mejoras implementadas en esta fase de integración fue la utilización de inferencia por lotes, mayormente conocido como *batching*. Esta técnica se aplicó a través del parámetro *batch_size,* una vez más producto del pipeline de Hugging Face. Esta técnica básicamente consiste en enviar los prompts al modelo agrupados en lugar de enviarlos individualmente para así poder aprovechar mejor la paralelización en GPU (concepto vital al trabajar con LLMs y con recursos limitados), reducir la latencia por consulta y acelerar el tiempo de inferencia, que ha sido durante todo el proyecto el principal lastre de la generación.

El tamaño de lote seleccionado fue de 4, ya que como se explica con mayor detalle en el apartado 4.4.2 de este documento, no se pueden agrupar más prompts debido al límite de tokens del modelo. Es buena práctica dejar siempre un margen prudencial de más del 50% de los tokens para la generación ya que en este caso de uso la segmentación de la información se ha realizado por párrafos y estos tienen dimensiones cambiantes.

Por último y siguiendo las técnicas del apartado 4.4.5, el resultado es almacenado en un archivo JSON y posteriormente procesado para eliminar la generación que no es útil para el proceso de ajuste fino, como referencias ambiguas o poco útiles. Esta limpieza también se integró como parte del flujo principal, garantizando una salida final coherente, depurada y lista para evaluación.

4.4.9. Tokens de acceso y descargas de código

A lo largo del desarrollo del proyecto se ha tenido como fuente principal de información, modelos y datasets la plataforma online de Hugging Face. Como es habitual en el campo de los proyectos relacionados con procesamiento de lenguaje natural (PLN), se ha tenido que descargar y autenticar varios modelos provenientes de la plataforma. Es de gran importancia tener en cuenta que los modelos no siempre se obtienen directamente del repositorio oficial. Por ejemplo, si una versión *instruct* del modelo Llama de la empresa Meta ha sido optimizada y ajustada para realizar tareas de clasificación, dicho modelo será publicado por el autor particular que haya realizado esa mejora y no por la propia empresa Meta.

Esto tiene varias implicaciones, en primer lugar, se debe incluir en el código el token de autenticación para poder acceder a esos modelos y realizar procesos de inferencia. Sin este log in desde el repositorio de ejecución, no se puede hacer uso de estos modelos.

En segundo lugar, el hecho de usar este tipo de modelos implica cierto riesgo que debe ser asumido, pero no debe ser ignorado. Para realizar la carga de los modelos generalmente se necesita añadir un parámetro de ejecución indicando que a pesar de que el código no ha sido expresamente desarrollado por un usuario de confianza, el usuario administrador permite a la unidad de computación acceder a ese código. Generalmente se emplea la siguiente expresión:

```
trust remote code = True
```

La inclusión de este parámetro en el proyecto implica explícitamente que se autoriza la ejecución de código remoto no revisado. En este caso de uso puede no resultar tan crítico ya que se cómo se ha mencionado con anterioridad en reiteradas ocasiones, la ejecución se ha llevado a cabo en una cuenta compartida de Google Colab, pero si el proyecto se ejecutase estrictamente en un entorno local este parámetro debería ser aplicado con cautela.

Finalmente, a pesar de que se ha revisado el código externo antes de autorizar su ejecución, es importante verificar que las unidades de computación cuentan con capacidad suficiente para soportar los requerimientos de los códigos externos ya que en la mayoría de los casos es el principal problema de cuellos de botella en la pipeline e incluso puede llegar a ocasionar daños permanentes e irreversibles en el hardware.

5. Análisis y Discusión de Resultados

5.1. Métricas

Aunque el objetivo principal del proyecto consiste en diseñar e implementar un flujo de trabajo semiautomatizado para la generación de conjuntos de datos a partir de corpus documentales privados (con el fin de permitir el ajuste fino de un modelo de lenguaje (LLM) sobre dicha información), resulta fundamental complementar este proceso con métricas cuantitativas que permitan evaluar objetivamente su eficacia.

Por ello se han seleccionado dos métricas principales:

- *Precisión semántica del sistema de preguntas y respuestas (Q&A):* se debe asegurar la precisión de sistema diseñado, es decir, si la generación de los pares pregunta-respuesta es coherente con la información de entrada proveniente de los textos. Es importante tomar conciencia de que el modelo responde según lo que se le entrega en el prompt y no según información que ha visto durante el entrenamiento, es precisamente por eso por lo que se eligieron modelos anteriores a los documentos con los que se han generado las preguntas y respuestas.
- *Cobertura del corpus documental:* se analiza si los diferentes fragmentos en los que se ha dividido el texto han sido accedidos durante la generación. De este modo se garantiza un aprovechamiento completo del conjunto de datos y se satisface la necesidad de cubrir toda la información disponible.

Finalmente, estas métricas se complementan con una tercera que deriva de las mismas: el número absoluto de errores, entendido como respuestas validadas negativamente por el sistema RAG evaluación automática. Este indicador ofrece una perspectiva adicional sobre la calidad del dataset generado. A continuación, se presentan algunos de los datos que conforman el proceso diseñado:

Característica	Valor
Cantidad de documentos	57
Número de chunks	10,845
Vocabulario del tokenizador	151,643
Tokens empleados	135,221
Pares Q&A (RAW)	44,429
Pares Q&A (Limpio)	40,628

Tabla 14. Características del Dataset

Como se puede apreciar en esta iteración, que ejemplifica el funcionamiento del proceso generado, se obtiene una media de aproximadamente 4 preguntas por párrafo de información.

Obviamente este valor oscila mucho ya que hay párrafos que solo son relevantes para una pregunta en particular y otros (como el *abstract* de los artículos) que, por su naturaleza más generalista y cargada de información relevante, dan lugar a 10-12 preguntas.

También cabe mencionar que a pesar de que el vocabulario del tokenizador no se emplea en su totalidad, no se aprecia una holgura de tokens innecesarios excesiva como para plantear un cambio de tokenizador. Además, este corpus documental es una prueba y el proceso se realiza con pretensiones de escalar la generación a corpus más grandes, por lo que sería conveniente contar con mayor cantidad de tokens.

Una vez analizadas las características que conforman el dataset, se presentan las métricas mencionadas con anterioridad:

Métrica	Valor
Precisión	96,73%
Chunks accedidos	97,24%

Tabla 15. Métricas de validación del proceso

5.2. Discusión de resultados

La precisión que presenta el dataset de control es de un 96,73%. A pesar de no alcanzar el 100% existen dos razones principales que explican este comportamiento, gracias a que, durante el RAG de validación, el modelo Mistral 7b no solo indicaba la veracidad de la generación, sino que además añade una explicación de porque se toma o no como válida la respuesta. A continuación, se presentan las razones que explican el 3,27% de respuestas consideradas como no válidas:

〉 En primer lugar, gran parte de las respuestas consideradas no válidas son correctas en cuanto a información se refiere. No obstante, los chunks accedidos por el RAG de validación para justificar la respuesta no representan la información de esta.

  Este indicio puede darse bien porque no se ha accedido al chunk específico del cual se ha originado la generación o bien porque el modelo ha contestado con información vista durante su entrenamiento, ya que a pesar de que los artículos utilizados son posteriores al lanzamiento del modelo, hay muchos conceptos que no tienen por qué ser necesariamente novedosos y sirven para explicar algunos de los avances de los artículos.

A continuación, se observa un claro ejemplo, donde el mecanismo de atención es una suposición plausible y de hecho técnicamente correcta, pero no es verificable a partir del contexto encontrado en los chunks más relacionados con la pregunta. A pesar de que el resultado de estas preguntas es negativo para la validación del proceso, destaca el rendimiento del RAG de validación, que penaliza cualquier tipo de afirmación no respaldada por el contexto concreto. Por tanto, estos falsos negativos respaldan aún más si cabe el proyecto, ya que la información que se genera no es fruto de alucinaciones y el proceso de validación es tanto estricto como necesario.

```
        "question": "What are the key components of the dominant sequence
transduction models mentioned in the document?",

        "expected_answer": "The key components of the dominant sequence
transduction models are complex recurrent or convolutional neural networks
with an encoder and a decoder, connected through an attention mechanism."

        "validation_result": "[/INST]\n    No, the expected answer is not
entirely correct according to the context. The context describes models
with an encoder-decoder structure, but it does not explicitly mention the
use of attention mechanisms. The models described in the context have an
encoder that maps input sequences to continuous representations, and a
decoder that generates output sequences one symbol at a time, consuming
previously generated symbols as additional input. The context also mentions
the importance of learning long-range dependencies and the effect of path
length on this ability. The models use learned embeddings and a shared
weight matrix between the embedding layers and the pre-softmax
transformation."
```

Tabla 16. Ejemplo de validación negativa de un par pregunta-respuesta

〉 En segundo lugar, el resto de las respuestas validadas de forma negativa encuentra su justificación en la capacidad limitada de los modelos utilizados para generalizar la evidencia textual disponible.

Si una respuesta es correcta pero el Mistral 7B ha empleado un término explícito en la verificación para dar sentido a la respuesta y el modelo de generación Qwen2 no ha empleado ese mismo término a pesar de que quede sobreentendido en la respuesta, la validación será considerada negativa. Este comportamiento simplemente pone en evidencia la importancia de generar una buena representación de los embeddings, ya que palabras íntimamente relacionadas con su contexto pueden estar lejos de este en corpus documentales que versan sobre temas relacionados.

Estos resultados validan el proceso diseñado y además permiten también validar el proceso implementado con RAG sobre el cual no se habían estipulado métricas. La revisión humana ha sido posible a través de un proceso iterativo en el cual se procesaba uno o dos documentos y se revisaban tanto los chunks extraídos como las validaciones del modelo.

Por último, cabe destacar que no se ha observado un deterioro del rendimiento en aquellos documentos en los que había mayor cantidad de información en formato de imagen, pero si se ha visto un decrecimiento notable de la cantidad de preguntas generadas en el JSON limpio. Este decrecimiento se evidencia únicamente en el JSON limpio ya que, en el *raw,* la generación ha mantenido unos niveles estables y de acuerdo con el resto de los documentos, pero referenciando figuras que no podían ser procesadas ya que se habían eliminado en fases previas del preprocesamiento de la información.

5.3. Discusión y Justificación de iteraciones previas

El desarrollo completo del proyecto se extiende a lo largo un año, y durante ese tiempo se han ido tomando varias alternativas de diseño con el fin de llegar al proceso más eficiente posible preservando las restricciones de ejecución en local y con vistas a futuro de garantizar flexibilidad y escalabilidad. En este apartado se pretende presentar algunas de las variaciones más relevantes que se han realizado en fases exploratorias y que finalmente no han sido implementadas en el proceso:

- *Implementación de YOLO para procesamiento de imágenes*: Durante las etapas de preprocesamiento de la información se tuvo muy presente la posibilidad de considerar imágenes como parte de la información del corpus documental, ya que, a pesar de su limitada información, permiten contextualiza muchas de las preguntas generadas. Siguiendo algunos proyectos de investigación se implementó el detector YOLO (*You Only Look Once).* Existen enfoques basados en YOLO dirigidos a identificar y clasificar bloques semánticos dentro de documentos PDF, como en Straka & Gruber con YOLOv8 [74] , o Sugiharto et al. con YOLOv5 [75] . La idea del proceso era escanear cada página de los documentos para así poder procesar tanto la información en formato de texto mediante OCR (*Optical Recognition of Characters*) e identificar las imágenes que acompañaban a este.

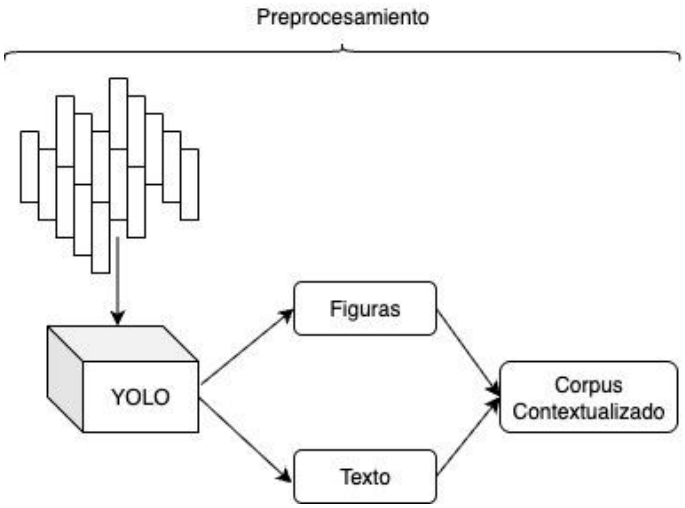

Figura 25. Flujo de preprocesamiento con implementación de YOLOv10

No obstante, a pesar de que el proceso era capaz de localizar las imágenes en el documento, no tenía la habilidad de extraer contenido de estas, es decir, a pesar de detectar la figura no era posible extraer conclusiones útiles ni contenido semántico (como sucede en el reciente artículo de Mosleh et al. [73]) contextual que permitiese incluir la información en el corpus procesado. Por ende, se descartó esta aproximación, aunque cabe destacar que el procesamiento del texto si era preciso y permitía procesar páginas completas sin necesidad de separar por chunks. De todos modos, el similar rendimiento del preproceso no justificaba el incremento de recursos computacionales.

〉 *Formateo de la información en archivo JSON:* Uno de los enfoques más utilizados en fases iniciales de proyectos NLP con grandes corpus de información textual es el de utilizar datos de Wikipedia. Esta fuente, disponible a través de Wikimedia en formato JSON y ofrece una estructura semántica clara además de una gran cantidad de información en formato de texto y referencias bibliográficas.

Sin embargo, las primeras iteraciones dieron visibilidad a dos limitaciones importantes. En primer lugar, la calidad y profundidad de la información y en segundo lugar la estructura excesivamente organizada con la que estaban redactados los artículos, lo que limitaba la validación del proceso. A pesar del intento fallido, se intentó adaptar artículos provenientes de arXiv al formato estructurado de los *dumps* de Wikimedia (en JSON), con el objetivo de aprovechar los scripts de procesamiento ya existentes. Sin embargo, esto no fue finalmente posible ya que los documentos científicos de arXiv

presentan un lenguaje más denso y una estructura desorganizada ya que no todos se rigen por las mismas normas de redacción.

〉 *Mixtral 8x7B para validación y generación*: Como se ha explicado a lo largo del proyecto, la intención inicial era la de implementar un proceso con Mixtral 8X7B para seguir una estrategia basada en la especialización por expertos, en la que se asignarían tareas específicas para la generación y luego otras para la validación. De este modo se

Figura 26. Posible aplicación de especialización de expertos en el proyecto

podría unificar el proyecto dentro de un solo modelos con expertos especializados en tareas particulares y adaptados al tema en cuestión, como por ejemplo se puede apreciar en la figura que aparece a continuación:

El principal problema radicaba no solo en el excesivo coste computacional, sino que además algunos de los expertos del modelo generaban las preguntas y respuestas en diferentes niveles de complejidad. Esta característica dificulta enormemente el proceso de validación. A pesar de ello, Mixtral 8x7B es una de las mejores opciones a implementar en este proyecto si no se tuviese la restricción de recursos y no se realizase la especialización de expertos ya que al tener 8 instancias permite abarcar un rango muy grande de registros y alcanzar unos niveles de complejidad insospechados para LLMs de 16 billones de parámetros.

〉 *Ejecución en entorno local Linux:* Al final de la primera fase iterativa del proyecto en relación con la generación del dataset, se trató de minimizar las limitaciones que presentaba de la cuenta de Google Colab mediante la ejecución en local en un ordenador con sistema operativo Linux. Con este enfoque se podía garantizar que el proceso se ejecutaba de forma ininterrumpida y se tenía un control absoluto del uso de recursos.

Lamentablemente, la implementación de librerías y paquetes no fue tan sencilla como se esperaba y la capacidad de la GPU tendía a saturarse a pesar de limpiar los datos ya utilizados después de subir el JSON al repositorio final, ya que los paquetes de control y gestión no eran accesibles. Finalmente, y gracias a esta iteración, se optó por realizar subidas constantes después de la ejecución de cada documento, como se ha explicado con anterioridad (ver apartado 4.4.4)

6. Conclusiones

6.1. Evaluación Global de proceso propuesto

El completo desarrollo de este proyecto ha evidenciado la posibilidad de generar datasets sintéticas de forma semiautomatizada en formato de preguntas y respuestas (Q&A) a partir de grandes corpus de información que versan sobre una temática común. El objetivo final de este proyecto no solo es el de llevar a cobo el posterior ajuste fino de un LLM, sino el de proponer un flujo de trabajo controlado en un entorno local que democratice el uso de LLMs en unidades de computación comunes, como puede ser un ordenador portátil.

Este proceso ha conseguido integrar en un mismo *pipeline* diferentes etapas del flujo de trabajo, desde la segmentación y preprocesamiento contextual, a la generación y validación de los resultados. Todo ello con las optimizaciones necesarias para lograr una eficiencia computacional cercana a la óptima sin sacrificar el rendimiento de los modelos utilizados.

Uno de los aspectos más importantes del proyecto reside en la capacidad del proceso para ser reproducible aun sabiendo que los LLMs siguen siendo en su gran mayoría cajas negras por donde la información se va procesando de acuerdo con los pesos de los parámetros sin aparente capacidad de trazar una ruta definida. Esta reproducibilidad es precisamente la que ha facilitado la ejecución en lotes de documentos y ha permitido asegurar la escalabilidad del proyecto.

Otra característica validada durante el proceso ha sido que los modelos generativos empleados han sido capaces de generar respuestas que reflejan el contenido explícito del documento minimizando los errores por alucinaciones hasta cero. Esta capacidad para discernir entre el conocimiento previamente aprendido durante el entrenamiento y el contenido obtenido por el prompt de entrada no es tan trivial de conseguir, ya que los modelos de Hugging Face empleados no siempre proporcionan información de los procesos de entrenamiento. Esta comprobación ha sido posible gracias a la cuidadosa selección del corpus, formado por documentos posteriores a la fecha de publicación del modelo, y a la aplicación de un sistema de validación cruzada con un segundo LLM (Mistral 7B), lo cual garantiza que el modelo no estaba influido por memorias latentes externas al corpus.

También se debe destacar que el proceso de validación implementado con el RAG se ha realizado de forma completamente determinista, lo que confirma una vez más la robustez del diseño. Esta faceta determinista se ha conseguido al recurrir a las versiones más antiguas de los modelos de embeddings: los embeddings estáticos. Estos han sido utilizados para la recuperación semántica del modelo RAG, dejando de lado los embeddings contextualizados, los cuales podrían introducir variabilidades incontrolables en función de su representación contextual.

Otra de las condiciones de diseño que no debe ser ignorada, es que el proceso se ha desarrollado en su totalidad (a excepción de esta memoria) en lengua inglesa para aprovechar los rendimientos de los modelos durante sus fases de entrenamiento. Ya que, si bien se ha afirmado que se desconoce la información utilizada para realizar dichos procesos, también se puede asegurar que, por calidad y cantidad, los grandes modelos de lenguaje son entrenados en inglés y posteriormente adaptados a otras lenguas con una pequeña perdida de facultades.

En conjunto, el sistema propuesto sienta las bases para la creación segura, controlada y eficiente de datasets adaptables a diferentes dominios, permitiendo su posterior uso en tareas de fine-tuning local sin comprometer la privacidad ni la integridad de los datos originales y garantizando un estándar de calidad en dominios complejos como pueden ser aquellos de corte académico.

6.2. Limitaciones

6.2.1. Limitaciones técnicas

A lo largo del desarrollo del proyecto se han puesto en evidencia y de forma constante las limitaciones técnicas a las que se enfrenta un proyecto de este tipo de características, fundamentalmente derivadas de la necesidad de reproducir el flujo de trabajo en entornos locales. En primer lugar, y por ende la más evidente, está la limitación técnica de ceñirse a ejecutar grandes modelo de lenguaje únicamente en inferencia sin modificar los pesos internos de los parámetros. De esta forma queda descartado cualquier proceso de ajuste fino, aunque por suerte, la comunidad de Hugging Face colabora activamente a la constante evolución y adaptación de estos.

Asimismo, a pesar de haber contado con acceso a GPU de alto rendimiento (concretamente la A100) mediante Google Colab Pro, el uso compartido de la cuenta entre dos usuarios ha introducido restricciones tanto temporales como de recursos, limitando la capacidad de ejecutar en cualquier momento y obligando a tomar precauciones extras para asegurar la preservación de los datos, dada la extensión de la ejecución. Estas restricciones han condicionado el diseño de la pipeline, obligando a implementar subidas y escrituras intermedias del JSON para asegurar la persistencia de los resultados.

Finalmente, también se ha debido tener en cuenta la visión de escalabilidad y reproducibilidad con la que se ha realizado el proyecto, por ejemplo, sobredimensionando el tokenizador elegido o fijando el parámetro de la temperatura a cero (véase el apartado 3.1.5 de este documento) respectivamente.

6.1.1. Principales desafíos

Uno de los principales desafíos del proyecto ha sido tener que pivotar del modelo Mixtral8x7B al Qwen2. Este desafío no engloba únicamente las dificultades de adaptar el código de implementación de un modelo a otro, ya que el Mixtral está construido sobre una arquitectura *Mixture of Experts (MoE)*, mientras que Qwen2 responde a una arquitectura densa tradicional, sino que también implica que gran parte del proceso de ingeniería del prompt que había sido optimizado para Mixtral8x7B debía ser reestructurado desde cero para garantizar una salida útil y válida.

Otro de los grandes desafío que han acompañado al proyecto desde la etapa inicial, es el de la falta de acceso a documentación privada, ya que es realmente complicado conseguir un corpus suficientemente grande que valida el diseño del proceso y que además se garantice que la información que aparece en él no ha sido vista previamente por el modelo. A pesar de ello, una de las ventajas que derivan de haber elegido un corpus fundamentalmente formado por documentos académicos, es que la estructura que siguen es relativamente similar.

Por último, también se encontraron muchas dificultades en las primeras fases iterativas de preprocesamiento de la información, ya que se estaba intentado realizar una segmentación en chunks de tal forma que cada partición de texto fuera capaz de relacionarse semánticamente con otras de similar representación vectorial en el espacio de embeddings. Sin embargo, este enfoque no es viable en la escala de entornos locales y realizando inferencia con modelos ya previamente entrenados. No obstante, la partición del texto en párrafos ha presentado un rendimiento excelente dado los resultados de la validación.

6.2. Líneas Futuras

A pesar de haber completado con éxito la consecución del proyecto, la naturaleza que engloba a las tareas de NLP y modelos de IA obliga a estructurar varias áreas de desarrollo y mejora adicionales. Este proyecto pone en evidencia la viabilidad de realizar un conjunto de datos sintético de pares pregunta-respuesta a partir de documentación privada (en este caso de uso documentos científicos de arXiv), y con ello sienta las bases para un desarrollo completo en direcciones futuras de optimización y sobre todo de expansión.

- *Implementación de Modelos Multimodales*: Como se ha mencionado en reiteradas ocasiones a lo largo de todo el proyecto, la principal limitación reside en la forma de extraer la información del corpus documental, ya que esta es puramente de carácter

textual. La incorporación de modelos multimodales y VLMs permitir a incluir en el proceso de inferencia la inclusión de elementos visuales como tablas o figuras. Además, esto también facilitaría enormemente la tarea de limpieza del JSON raw, ya que de las 3,801 preguntas eliminadas en la Tabla 14, la gran mayoría es a causa de referencias textuales a elementos gráficos no incluidos en el prompt de entrada.

Esta línea futura ya ha sido explorada con anterioridad en trabajos como el de Xie et al., donde desarrollan *PDF-Wukong* [76] , un modelo multimodal diseñado para combinar visión y lenguaje para realizar conjuntos de Q&A sobre PDFs largos mediante muestreo visual-textual eficiente.

Pero lo realmente valioso para este proyecto seria conseguir encontrar un balance razonable entre el desempeño del modelo y su tamaño, ya que los modelos multimodales como GPT-4o son extremadamente pesados e incompatibles en entornos locales con unidades de computación limitadas como es el caso del proyecto.

Finalmente, dentro del contexto de expansión de capacidades de modelos LLM, sería interesante estudiar la posibilidad de cruzar referencias dentro de los archivos, un enfoque que dada la naturaleza de este proyecto de ejecución en entornos locales resulta imposible, ya que si se accediese a todas las referencias bibliográficas de los documentos se acabaría ocupando todo el repositorio académico. Pero sí que es una opción en aquellos entornos cerrados como pueden ser pequeñas empresas con espacio de trabajo como por ejemplo *Confluence*.

〉 *Optimización de Recursos Computacionales*: el uso de técnicas como la cuantización han demostrado ser efectivas sin comprometer gravemente el rendimiento del modelo para este caso de uso. Sin embargo, con acceso a modelos más novedosos y ya ajustados para tareas específicas, se podrían explorar otro tipo de técnicas de optimización como la destilación del conocimiento de modelos de mayor tamaño, el uso de arquitecturas complejas eficientes (MoE) o la poda de pesos (*pruning*).

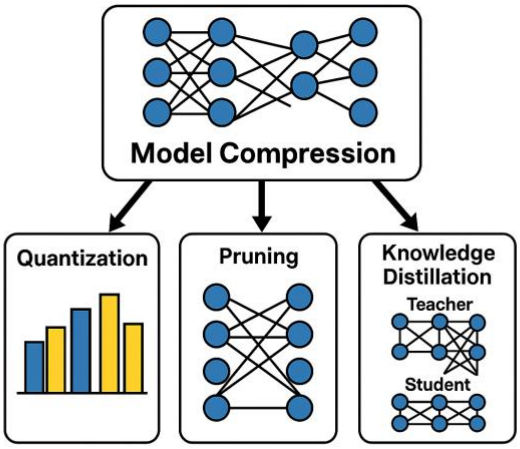

Figura 27. Técnicas comunes de compresión de modelos para reducción de recursos computacionales

En esta misma línea, también se podrían hacer avances en la velocidad de inferencia ya que representa el cuello de botella de la generación del dataset Para ello, nos solo se deben atender modelos novedosos, también sería interesante experimentar con modelos más ligeros y antiguos como BERT e intentar reproducir las prestaciones de modelos más complejos por medio de un entrenamiento adaptado a las tareas de generación de preguntas y respuestas.

〉 *Mejora de la Trazabilidad y Validación*: Actualmente, el proceso de validación se basa en una validación cruzada en un entorno de modelo RAG con una búsqueda vectorial para extraer los segmentos de texto más representativos. Futuras investigaciones podrían ahondar en buscar mejorar la trazabilidad dentro de los pesos del modelo, aunque tal y como se explica en [77] , resolver el problema de la caja negra en los grandes modelos de lenguaje implica comprender cómo y por qué los modelos toman determinadas decisiones a partir de representaciones internas no interpretables directamente.

Una trazabilidad completa o por lo menos más profunda, podría permitir vincular directamente cada par pregunta-respuesta con los tokens directamente involucrados en la generación, quedando de esta forma validada la generación de forma instantánea. Además, permitiría realizar un subproceso de validación y otro de generación de forma simultánea sin necesidad de cargar los chunks extraídos y aumentando la velocidad total de proyecto ya que se podrían paralelizar los procesos de inferencia en lugar de procesarlos de forma secuencial como se está realizando en este caso de uso

〉 *Expansión Multilingüe y del Corpus Documental:* El proyecto se ha desarrollado con documentos de arXiv y a pesar de haber demostrado una competencia absoluta con documentación técnica y especializada de esta índole, se podría indagar más aun en las diferencias de rendimiento al cambiar de registro. De hecho, el proyecto está pensado para implementarse en documentación privadas como puede ser el de una PYME que busca integrar herramientas de IA en su metodología de trabajo.

Otra característica de este diseño es que se basa completamente en la lengua inglesa, desde el diseño del prompt hasta la información de los documentos. Esto se debe principalmente a que la gran mayoría de textos con los que se entrenan los grandes modelos de lenguaje es en inglés. A pesar de que los modelos son competentes en muchos otros idiomas, esta conversión del idioma raíz a otra lengua, va acompañada de una perdida notable de prestaciones. Como se demuestra en [78] , al comparar el desempeño de los LLMs en lenguas mayoritarias frente a lenguas minoritarias con menos información vista durante el entrenamiento:

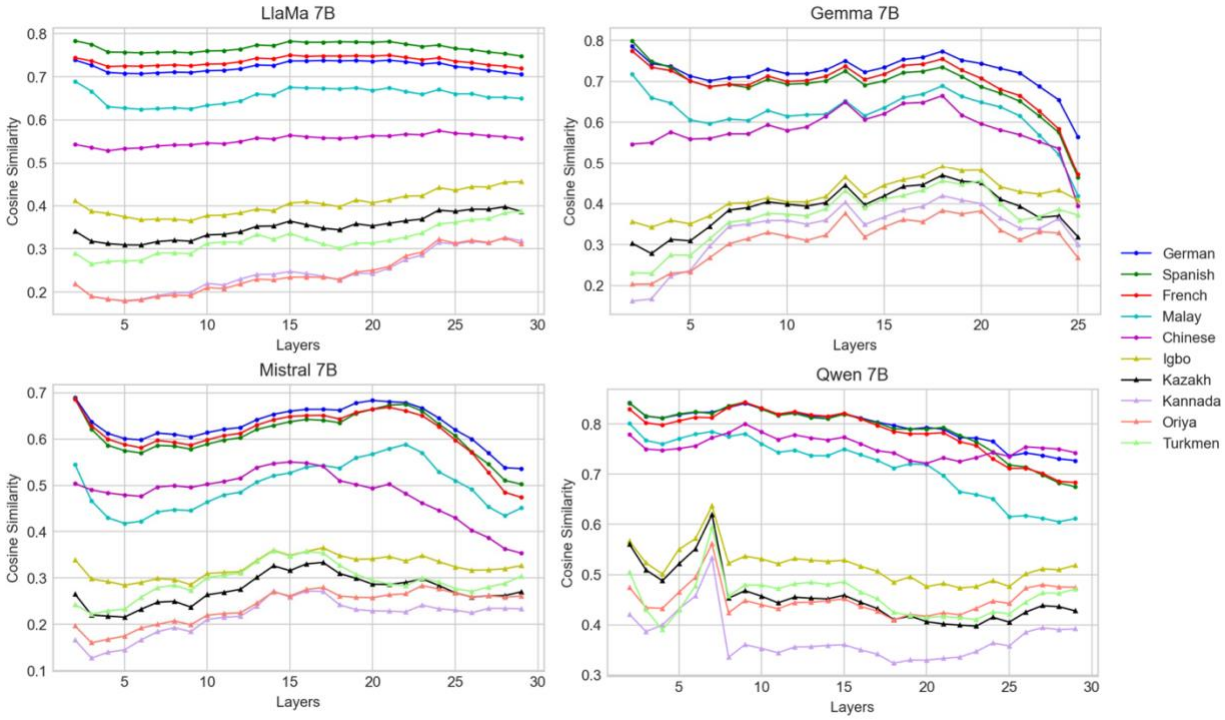

Figura 28. Rendimiento de diferentes LLMs para diez tipos de idiomas. Fuente [78]

En la Figura 28, se puede observar como el rendimiento de los idiomas de alta disponibilidad de recursos: el alemán, español, francés, indonesio y chino, está de forma continuada muy por encima del resto de idiomas, en este caso los de baja disponibilidad de recursos, el igbo, kazajo, canarés, oriya y turcomano.

Finalmente, se debe destacar que la investigación en la dirección de estas líneas futuras que se acaban de presentar está sujeta a la evolución de los LLMs y su capacidad para destilar su

conocimiento en modelos más manejables y procesables por herramientas de limitadas prestaciones, ya que el uso de estos modelos sin restricciones de computación ya da lugar a la consecución de los objetivos de estas líneas futuras.

7. Planificación

7.1. Planificación temporal

A continuación, en este apartado, se desglosa la planificación temporal del proyecto. Se debe destacar que, a pesar de haber realizado una planificación inicial, a lo largo del proyecto de diseño han ido surgiendo diferentes tareas secundarias que debían ser completadas para poder continuar.

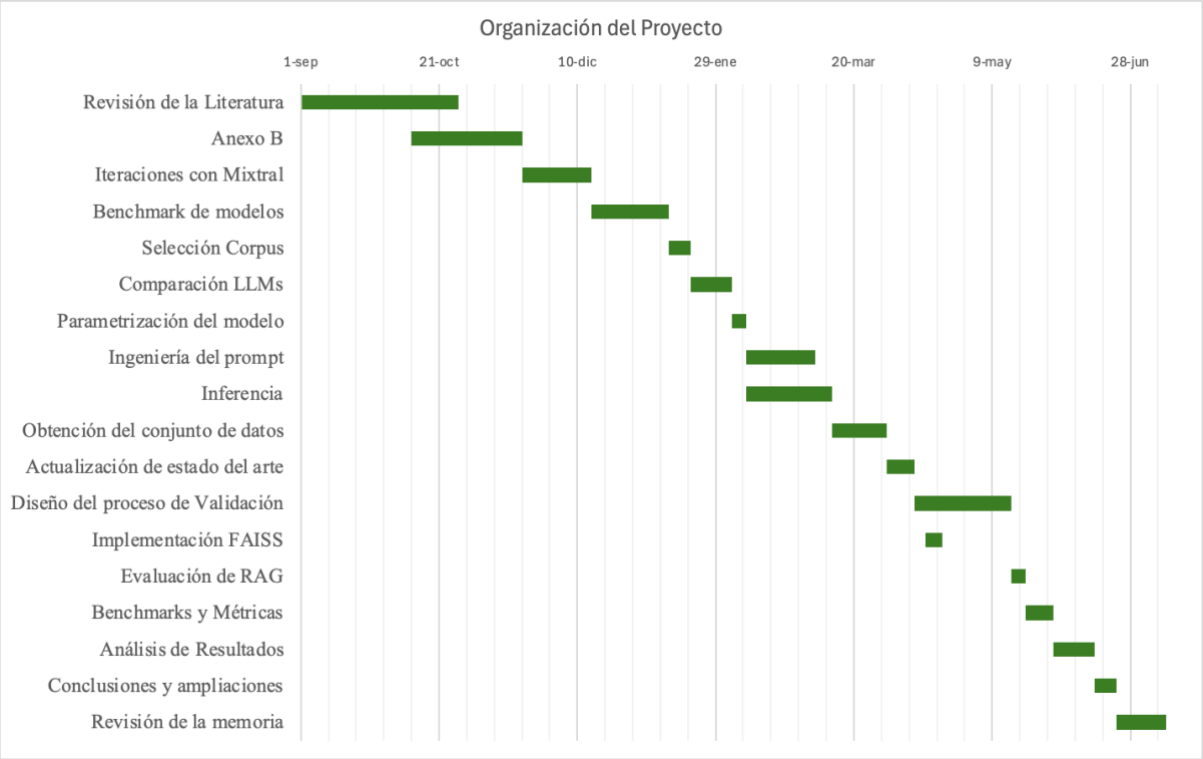

Figura 29. Organización temporal del proyecto - Diagrama de Gantt

A pesar de las múltiples etapas que conforman el proyecto, debe destacarse que como ya se ha mencionado con anterioridad, la rápida evolución de los modelos de lenguaje y de las herramientas utilizadas en proyectos de NLP, fue necesario realizar una actualización del estado del arte a mitad del proyecto. Esta decisión viene motivada por la necesidad de producir un trabajo que reflejase lo máximo posible la situación actual del campo de investigación en cuestión.

También se debe destacar que el proceso de inferencia y de diseño del proceso de validación son las tareas más largas del proyecto principalmente por la influencia que tienen los largos tiempos de computación en ese tipo de tareas.

7.2. Estudio Económico

Como en todo proyecto, se debe documentar la viabilidad y costo de este. Seguidamente se incluye una estimación de los recursos económicos destinados a la consecución del proyecto. Este apartado es de relativa importancia para poder contextualizar el proceso en su totalidad, ya que no solo determina la viabilidad de este, sino que también sienta las bases para continuar con las líneas futuras (véase el apartado 6.2 de este documento) y determina la escalabilidad de este.

En esta estimación únicamente se incluyen las partidas de gastos que afectan directamente al proyecto de generación y no a la posterior fase de ajuste fino del modelo. Además, no se han considerado los activos utilizados durante el proyecto que ya fueran adquiridos previos al inicio del mismo.

Id.	Unidades	Concepto	Precio (€/unidad)	Cantidad	Importe (€)
1		*Recursos humanos*			
1.1	h	Autor	10	420	4200,00
2		*Recursos TIC*			
2.1	Mes	Google Colab Pro	25,56	6	153,36
3		*Recursos Materiales*			
3.1	kWh	Consumo eléctrico	0,1155*	89,95	10,38
3.2	-	Ordenador	1000	1	1000
4		*Amortizaciones*			
4.1	h	Ordenador	0,057**	420	23,97
Total					5387,71

Tabla 17. Desglose de las partidas de gastos del proyecto

*Para realizar el cálculo del consumo eléctrico del proyecto se ha tenido en cuenta la energía consumida por el ordenador durante las horas de trabajo y se ha tomado de referencia el precio del kWh a día 1 de julio de 2025 según Red Eléctrica Española.

**Para realizar el cálculo de la amortización del proyecto se ha tenido en cuenta el precio del ordenador portátil utilizado durante las horas de trabajo y contando con que el activo se deprecia en un espacio de 6 años a 8 horas de trabajo diarias.

8. Referencias

[1] M. R. J, K. VM, H. Warrier, and Y. Gupta, "Fine-tuning LLM for Enterprise: Practical Guidelines and Recommendations," *arXiv preprint* arXiv:2404.10779, 2024.

[2] C. Jeong, "Domain-specialized LLM: Financial fine-tuning and utilization method using Mistral 7B," *Journal of Intelligence and Information Systems*, vol. 30, no. 1, pp. 93–120, Mar. 2024.

[3] Z. Han, C. Gao, J. Liu, J. Zhang, and S. Q. Zhang, "Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey," *arXiv preprint arXiv:2403.14608*, 2024.

[4] L. Zhang, K. Jijo, S. Setty, E. Chung, F. Javid, N. Vidra, and T. Clifford, "Enhancing Large Language Model Performance to Answer Questions and Extract Information More Accurately," *arXiv preprint arXiv:2402.01722*, 2024.

[5] F. Liu, Y. Liu, L. Shi, H. Huang, R. Wang, Z. Yang, L. Zhang, Z. Li, and Y. Ma, "Exploring and Evaluating Hallucinations in LLM-Powered Code Generation," *arXiv preprint arXiv:2404.00971*, 2024.

[6] C. X. Yu, C. S. Y. James, and P. H. L. P. David, "CAN LLMs Have a Fever? Investigating the Effects of Temperature on LLM Security," 2024.

[7] W. Liu, W. Zeng, K. He, Y. Jiang, and J. He, "What Makes Good Data for Alignment? A Comprehensive Study of Automatic Data Selection in Instruction Tuning," *arXiv preprint* arXiv:2312.15685, 2024.

[8] J. Ye, Y. Yang, Q. Zhang, T. Gui, X. Huang, P. Wang, Z. Shi, and J. Fan, "Empirical Insights on Fine-Tuning Large Language Models for Question-Answering," *arXiv preprint arXiv:2409.15825*, 2024.

[9] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. Siren's song in the ai ocean: A survey on hallucination in large language models. arXiv preprint arXiv:2309.01219, 2023.

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

[11] A. Vaswani *et al.*, "Attention Is All You Need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[12] X. Zhao, S. Cheng, G. Lu, J. Fang, H. Zhou, B. Jia, Z. Liu, and Y. You, "AutoChunk: Automated Activation Chunk for Memory-Efficient Long Sequence Inference," *arXiv*, 2024

[13] M. Ren, B. Cao, H. Lin, C. Liu, X. Han, K. Zeng, W. Guanglu, X. Cai, and L. Sun, "Learning or Self-aligning? Rethinking Instruction Fine-tuning," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Bangkok, Thailand, Aug. 2024, pp. 6090–6105.

[14] Z. Gekhman, G. Yona, R. Aharoni, M. Eyal, A. Feder, R. Reichart, and J. Herzig, "Does Fine-Tuning LLMs on New Knowledge Encourage Hallucinations?" *arXiv*, 2024.

[15] Kang, Y., Cai, Z., Tan, C. W., Huang, Q., & Liu, H. (2020). Natural language processing (NLP) in management research: A literature review. *Journal of Management Analytics*, *7*(2), 139–172. https://doi.org/10.1080/23270012.2020.1756939

[16] E. Hua, B. Qi, K. Zhang, Y. Yu, N. Ding, X. Lv, K. Tian, and B. Zhou, "Intuitive Fine-Tuning: Towards Simplifying Alignment into a Single Process," *arXiv preprint* arXiv:2405.11870, 2024.

[17] AI Engineer World's Fair, "The Hierarchy of Needs for Training Dataset Development," 2024.

[18] N. Ding, Y. Chen, B. Xu, Y. Qin, Z. Zheng, S. Hu, Z. Liu, M. Sun, and B. Zhou, "Enhancing Chat Language Models by Scaling High-quality Instructional Conversations," *arXiv preprint* arXiv:2305.14233, 2023.

[19] G. Cui, L. Yuan, N. Ding, G. Yao, B. He, W. Zhu, Y. Ni, G. Xie, R. Xie, Y. Lin, Z. Liu, and M. Sun, "UltraFeedback: Boosting Language Models with Scaled AI Feedback," *arXiv preprint* arXiv:2310.01377, 2024.

[20] G. Wang, S. Cheng, X. Zhan, X. Li, S. Song, and Y. Liu, "OpenChat: Advancing Open-source Language Models with Mixed-Quality Data,"

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv preprint arXiv:1810.04805*, 2018

[22] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving Language Understanding by Generative Pre-Training," OpenAI, 2018. [Online]. Available: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

[23] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," *arXiv preprint arXiv:1910.10683*, 2019

[24] A. Yang *et al.*, "Qwen2 Technical Report," *arXiv preprint arXiv:2407.10671*, 2024.

[25] H. H. Jiang, L. Brown y J. Cheng, "AI Art and its Impact on Artists," *en Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society (AIES '23)*, 2023, pp. 1–10. DOI: 10.1145/3600211.3604681.

[26] D. Bahdanau, K. Cho, and Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate,” *in Proceedings of the 3rd International Conference on Learning Representations (ICLR 2014)*, 2014.

[27] Z. Niu, G. Zhong, y H. Yu, “A review on the attention mechanism of deep learning,” *Neurocomputing*, vol. 452, pp. 48–62, 2021. doi: 10.1016/j.neucom.2021.03.091.

[28] P. Kumar, “Large language models (LLMs): survey, technical frameworks, and future challenges,” *Artificial Intelligence Review*, vol. 57, no. 260, Aug. 2024. [Online]. Available: https://link.springer.com/article/10.1007/s10462-024-10888-y

[29] C. Ling, X. Zhao, J. Lu, C. Deng, C. Zheng, J. Wang, T. Chowdhury, Y. Li, H. Cui, X. Zhang, T. Zhao, A. Panalkar, D. Mehta, S. Pasquali, W. Cheng, H. Wang, Y. Liu, Z. Chen, H. Chen, C. White, Q. Gu, J. Pei, C. Yang, and L. Zhao, “Domain Specialization as the Key to Make Large Language Models Disruptive: A Comprehensive Survey,” *arXiv preprint arXiv:2305.18703v7*, Mar. 2024. [Online]. Available: https://arXiv.org/abs/2305.18703v7

[30] OpenAI, “GPT-4 Technical Report,” *arXiv preprint arXiv:2303.08774*, Mar. 2023. [Online]. Available: https://arXiv.org/abs/2303.08774

[31] O. Onitilo *et al.*, “Application of large language models in clinical speech-to-text systems,” cited in: P. Kumar *et al.*, “Large language models (LLMs): survey, technical frameworks, and future challenges,” *Artificial Intelligence Review*, vol. 57, no. 260, Aug. 2024. [Online]. Available: https://link.springer.com/article/10.1007/s10462-024-10888-y

[32] A. Cvetanović and P. Tadić, “Synthetic Dataset Creation and Fine-Tuning of Transformer Models for Question Answering in Serbian,” *arXiv preprint arXiv:2404.08617*, 2024. [Online]. Available: https://arXiv.org/abs/2404.08617

[33] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, “SQuAD: 100,000+ questions for machine comprehension of text,” in *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2016, pp. 2383–2392.

[34] F. Pourpanah, M. Abdar, Y. Luo, X. Zhou, R. Wang, C. P. Lim, X.-Z. Wang, and Q. M. J. Wu, "A review of generalized zero-shot learning methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 4, pp. 4051–4070, 2023, doi: 10.1109/TPAMI.2022.3191696.

[35] J. Chen *et al.*, "Zero-Shot and Few-Shot Learning With Knowledge Graphs: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 111, no. 6, pp. 653–685, Jun. 2023, doi: 10.1109/JPROC.2023.3279374.

[36] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing," *arXiv preprint arXiv:1808.06226*, 2018. [Online]. Available: https://arXiv.org/abs/1808.06226.

[37] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2020. [Online]. Available: https://arXiv.org/abs/1910.01108.

[38] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016. [Online]. Available: https://arXiv.org/abs/1609.08144.

[39] M. Berglund and B. van der Merwe, "Formalizing BPE Tokenization," *Electronic Proceedings in Theoretical Computer Science*, vol. 388, pp. 16–27, Sep. 2023, doi: 10.4204/eptcs.388.4.

[40] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, Berlin, Germany, Aug. 2016, pp. 1715–1725. [Online]. Available: https://arXiv.org/abs/1508.07909

[41] P. Gage, "A new algorithm for data compression," *The C Users Journal*, vol. 12, no. 2, pp. 23–38, Feb. 1994.

[42] R. Boselli, S. D'Amico, and N. Nobani, "eXplainable AI for Word Embeddings: A Survey," *Cognitive Computation*, vol. 17, 2024, doi: 10.1007/s12559-024-10373-2.

[43] R. Borah, A. Pal, S. Bhattacharjee, S. Das and R. Naskar, "Are Word Embedding Methods Stable and Should We Care About It?," *2021 IEEE 7th International Conference on Big Data Security on Cloud (BigDataSecurity)*, New York, NY, USA, 2021, pp. 63–68, doi: 10.1109/BigDataSecurity52399.2021.0001.

[44] A. Thawani, S. Ghanekar, X. Zhu, and J. Pujara, "Learn Your Tokens: Word-Pooled Tokenization for Language Modeling," *arXiv preprint arXiv:2310.11628*, 2023. [Online]. Available: https://arXiv.org/abs/2310.11628

[45] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *arXiv preprint arXiv:1301.3781*, 2013. [Online]. Available: https://arXiv.org/abs/1301.3781

[46] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation," in *Proc. 2014 Conf. Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, Oct. 2014, pp. 1532–1543. [Online]. Available: https://aclanthology.org/D14-1162/

[47] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," *Trans. Assoc. Comput. Linguist.*, vol. 5, pp. 135–146, 2017. [Online]. Available: https://aclanthology.org/Q17-1010/

[48] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018. [Online]. Available: https://arXiv.org/abs/1802.05365

[49] Mistral AI, "Mixtral of Experts: A High-Quality Sparse Mixture-of-Experts Model," *Mistral AI*, Dec. 2023. [Online]. Available: https://mistral.ai/en/news/mixtral-of-experts

[50] K. M. Lo, Z. Huang, Z. Qiu, Z. Wang, and J. Fu, "A Closer Look into Mixture-of-Experts in Large Language Models," *arXiv preprint arXiv:2406.18219*, 2024. [Online]. Available: https://arXiv.org/abs/2406.18219

[51] R. Bommasani *et al.*, "On the Opportunities and Risks of Foundation Models," *arXiv preprint arXiv:2108.07258*, Aug. 2021. [Online]. Available: https://arXiv.org/abs/2108.07258

[52] Y. Liu, H. He, T. Han, X. Zhang, M. Liu, J. Tian, Y. Zhang, J. Wang, X. Gao, T. Zhong, Y. Pan, S. Xu, Z. Wu, Z. Liu, X. Zhang, S. Zhang, X. Hu, T. Zhang, N. Qiang, T. Liu, and B. Ge, "Understanding LLMs: A Comprehensive Overview from Training to Inference," *arXiv preprint arXiv:2401.02038*, 2024. [Online]. Available: https://arXiv.org/abs/2401.02038

[53] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," *arXiv preprint arXiv:1906.02243*, 2019.

[54] S. Vatsal and H. Dubey, "A survey of prompt engineering methods in large language models for different NLP tasks," *arXiv preprint arXiv:2407.12994*, 2024. [Online]. Available: https://arXiv.org/abs/2407.12994

[55] P. P. Liang *et al.*, "MultiBench: Multiscale Benchmarks for Multimodal Representation Learning," *Advances in Neural Information Processing Systems*, vol. 2021, no. DB1, pp. 1–20, 2021.

[56] W. Yang, Y. Xie, L. Tan, K. Xiong, M. Li, and J. Lin, "Data Augmentation for BERT Fine-Tuning in Open-Domain Question Answering," *arXiv preprint arXiv:1904.06652*, 2019. [Online]. Available: https://arXiv.org/abs/1904.06652

[57] S. Y. Feng, V. Gangal, D. Kang, T. Mitamura, and E. Hovy, "GenAug: Data Augmentation for Finetuning Text Generators," *arXiv preprint arXiv:2010.01794*, 2020. [Online]. Available: https://arXiv.org/abs/2010.01794

[58] A. Yang *et al.*, "Qwen3 Technical Report," *arXiv* preprint arXiv:2505.09388, May 2025. [Online]. Available: https://arXiv.org/abs/2505.09388

[59] Gemma Team *et al.*, "Gemma 3 Technical Report," *arXiv preprint* arXiv:2503.19786, Mar. 2025. [Online]. Available: https://arXiv.org/abs/2503.19786

[60] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling Laws for Neural Language Models," *arXiv* preprint arXiv:2001.08361, Jan. 2020. [Online]. Available: https://arXiv.org/abs/2001.08361

[61] J. Hoffmann *et al.*, "Training Compute-Optimal Large Language Models," *arXiv* preprint arXiv:2203.15556, Mar. 2022. [Online]. Available: https://arXiv.org/abs/2203.15556

[62] S. Bird, E. Klein, and E. Loper, Natural Language Processing with Python. Sebastopol, CA, USA: O'Reilly Media, 2009.

[63] Y. Liu *et al.*, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," *arXiv preprint arXiv:1907.11692*, 2019. [Online]. Available: https://arXiv.org/abs/1907.11692

[64] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, y J. Dean, *Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer*, arXiv preprint arXiv:1701.06538, 2017. [En línea]. Disponible en: https://arXiv.org/abs/1701.06538

[65] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer y Z. Chen, *GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding*, arXiv preprint arXiv:2006.16668, 2020. [En línea]. Disponible en: https://arXiv.org/abs/2006.16668

[66] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. de las Casas, E. Bou Hanna, F. Bressand, G. Lengyel, G. Bour, G. Lample, L. R. Lavaud, L. Saulnier, M.-A. Lachaux, P. Stock, S. Subramanian, S. Yang, S. Antoniak, T. Le Scao, T. Gervet, T. Lavril, T. Wang, T. Lacroix y W. El Sayed, *Mixtral of Experts*, arXiv preprint arXiv:2401.04088, 2024. [En línea]. Disponible en: https://arXiv.org/abs/2401.04088

[67] K. Enevoldsen *et al.*, "MMTEB: Massive Multilingual Text Embedding Benchmark," *arXiv preprint arXiv:2502.13595*, 2025. [Online]. Available: https://arXiv.org/abs/2502.13595

[68] P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020. [Online].
Available: https://papers.nips.cc/paper_files/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

[69] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, "MTEB: Massive Text Embedding Benchmark," arXiv preprint arXiv:2210.07316, 2023. [Online]. Available: https://arXiv.org/abs/2210.07316

[70] J. Amodei, "Inference: The next step in GPU-accelerated deep learning," *NVIDIA Developer Blog*, Aug. 2016. [Online]. Available: https://developer.nvidia.com/blog/inference-next-step-gpu-accelerated-deep-learning/

[71] "RAG Vs Fine-Tuning for Enhancing LLM Performance," GeeksforGeeks, May 6, 2024. [Online]. Available: https://www.geeksforgeeks.org/nlp/rag-vs-fine-tuning-for-enhancing-llm-performance/

[72] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder, and F. Wei, "Improving Text Embeddings with Large Language Models," arXiv preprint arXiv:2401.00368, 2024. [Online]. Available: https://arXiv.org/abs/2401.00368

[73] A. Mosleh, M. L. Mekhalfi, A. Khodja, and A. Boudjelal, "Historical Document Layout Analysis Using YOLO Object Detection Algorithms," *Applied Sciences*, vol. 15, no. 6, p. 3164, Mar. 2023. [Online]. Available: https://doi.org/10.3390/app15063164

[74] J. Straka and I. Gruber, “Object Detection Pipeline Using YOLOv8 for Document Information Extraction,” in *CEUR Workshop Proceedings*, vol. 3497, 2023. [Online]. Available: https://ceur-ws.org/Vol-3497/paper-051.pdf

[75] H. Sugiharto, Y. Silviana, and Y. S. Nurpazrin, “Unveiling Document Structures with YOLOv5 Layout Detection,” *arXiv preprint arXiv:2309.17033*, Sep. 2023. [Online]. Available: https://arXiv.org/abs/2309.17033

[76] X. Xie, H. Yan, L. Yin, Y. Liu, J. Ding, M. Liao, Y. Liu, W. Chen, and X. Bai, “PDF-WuKong: A Large Multimodal Model for Efficient Long PDF Reading with End-to-End Sparse Sampling,” *arXiv preprint arXiv:2410.05970*, Jan. 2025. [Online]. Available: https://arXiv.org/abs/2410.05970

[77] H. Luo and L. Specia, “From Understanding to Utilization: A Survey on Explainability for Large Language Models,” *arXiv preprint arXiv:2401.12874*, Jan. 2024. [Online]. Available: https://arXiv.org/abs/2401.12874

[78] Z. Li, Y. Shi, Z. Liu, F. Yang, A. Payani, N. Liu, and M. Du, “Quantifying Multilingual Performance of Large Language Models Across Languages,” *arXiv preprint arXiv:2404.11553*, Dec. 2024. [Online]. Available: https://arXiv.org/abs/2404.11553

[79] L. Wu, Z. Zheng, Z. Qiu, H. Wang, H. Gu, T. Shen, C. Qin, C. Zhu, H. Zhu, Q. Liu, H. Xiong, and E. Chen, "A survey on large language models for recommendation," *World Wide Web*, vol. 27, Aug. 2024, doi: 10.1007/s11280-024-01291-2

Tabla de Abreviaturas

BERT	Bidirectional Encoder Representations from Transformers
BPE	Byte-Pair Encoding
CNN	Convolutional Neural Network
FNN	Feedforward Neural Network
FLOPs	Floating-Point Operations Per second
GPT	Generative Pre-trained Transformer
IFT	Instruction Fine-Tuning
JSON	JavaScript Object Notation
LLM	Large Language Models
MTEB	Massive Text Embedding Benchmark
NDA	Non Disclousure Agreement
NLP/PLN	Natural Language Processor/Procesamiento de Lenguaje Natural
OCR	Optical Character Recognition
OOV	Out-of-Vocabulary
Q&A	Questions and Answers
RAG	Retrieval-Augmented Generation
RLHF	Reinforcement Learning from Human Feedback
RNN	Recurrrent Neural Network
SFT	Supervised Fine-Tuning
VLM	Vision Language Model
YOLO	You Only Look Once

ANEXO I: Alineación con los ODS de la agenda 2030

El proyecto contribuye tanto directa como indirectamente a varios Objetivos de Desarrollo Sostenible (ODS) planteados por la ONU en la Agenda 2030, especialmente en áreas de innovación, trabajo decente, y seguridad de la información:

〉 ODS 8: Trabajo Decente y Crecimiento Económico

Este proyecto contribuye al ODS 8, en especial a la meta de mejorar la productividad mediante la incorporación de tecnologías avanzadas. La creación de un flujo de trabajo semi automatizado para generar conjuntos de datos que faciliten la implementación de asistentes virtuales optimiza y agiliza el proceso de implementación de estos, permitiendo que los empleados dediquen menos tiempo a la búsqueda de información y más a tareas que generan un valor añadido a la empresa.

〉 ODS 9: Industria, Innovación e Infraestructura

Este proyecto está alineado con el ODS 9, especialmente en cuanto al fomento de la innovación tecnológica dentro de la infraestructura corporativa. La generación de un conjunto de datos específico para el ajuste de modelos de lenguaje permite a las empresas desarrollar herramientas de IA personalizadas, lo que permite a las industrias de cualquier tamaño contar con herramientas de IA ajustadas a sus necesidades independientemente de los recursos y necesidades.

〉 ODS 16: Paz, Justicia e Instituciones Sólidas

En el contexto de la protección de la información, este proyecto contribuye al ODS 16 en cuanto a la promoción de instituciones responsables. Al desarrollar un flujo de trabajo que permite la generación de conjuntos de datos en un entorno seguro y controlado, evitando la publicación de información privada en nubes externas, por lo que el proyecto fomenta una cultura de gestión responsable de la información.