UNIVERSIDAD ALFONSO X EL SABIO

Business Tech

Máster Universitario en Inteligencia Artificial

TRABAJO DE FIN DE MÁSTER

Diseño e Implementación de un Sistema RAG para la Evaluación Comparativa de LLMs en Asistencia Académica

Pablo Antonio García Pastor

Tutor: Ángel Manuel Rayo Acevedo

23/06/2025

UAX Universidad Alfonso X el Sabio

Índice general

CAPÍTULO 1

Resumen

El presente Trabajo de Fin de Máster aborda la implementación y evaluación exhaustiva de un asistente académico fundamentado en la sinergia entre modelos de lenguaje de gran tamaño (**LLM**, *Large Language Models*) y la arquitectura de Generación Aumentada por Recuperación (**RAG**, *Retrieval-Augmented Generation*). El estudio se centra en construir un sistema RAG funcional y, posteriormente, utilizarlo como plataforma para comparar rigurosamente el rendimiento de cuatro LLM de vanguardia: **GPT-4o**, **Gemini 1.5 Pro**, **Mixtral 8x22B Instruct** y **Llama 3.1 70B Instruct**. El objetivo final es determinar la configuración LLM+RAG más idónea para tareas académicas genéricas.

La fase experimental implicó el desarrollo de un pipeline RAG capaz de procesar consultas, recuperar fragmentos de información pertinentes de una base de conocimiento documental representativa de diversas áreas académicas (Biología, Física, Historia, y Lengua y Literatura), y proveer este contexto a los LLM. Sobre esta base, se evaluó la capacidad de cada modelo para ejecutar tareas clave como la *síntesis de textos*, la *generación de preguntas* y la *elaboración de respuestas fundamentadas*.

El análisis comparativo se realizó a través de múltiples dimensiones. Cualitativamente, se midió la *precisión factual*, la *calidad de síntesis*, la *coherencia narrativa* y la *corrección gramatical y de estilo*. Cuantitativamente, se evaluó la incidencia de *alucinaciones*, la efectividad en el *uso del contexto RAG proporcionado*, los *costes económicos* asociados al uso de las API de cada modelo y la *eficiencia temporal* del sistema completo. Los resultados de esta investigación ofrecen una visión detallada de las fortalezas, debilidades y el balance general de cada LLM al operar dentro de la arquitectura RAG implementada, proveyendo criterios fundados para la selección del modelo más adecuado en el contexto de un asistente académico.

CAPÍTULO 2

Introducción

En los últimos años, los modelos de lenguaje de gran tamaño (**LLM**) se han convertido en una pieza central de los avances en inteligencia artificial, logrando un rendimiento sin precedentes en tareas de *procesamiento de lenguaje natural* (e.g., [2]). Modelos como **GPT-3** y **GPT-4** de OpenAI han demostrado la capacidad de generar *texto coherente y contextual* a nivel casi humano, facilitando aplicaciones que van desde *asistentes virtuales* hasta apoyo en *redacción* y *análisis de documentos*. Esta capacidad los convierte en candidatos naturales para entornos académicos, donde podrían asistir en la *comprensión y generación de conocimiento* (*resúmenes de textos*, *respuesta a preguntas técnicas*, *tutoría personalizada*, etc.). Sin embargo, la aplicación directa de **LLM** en contextos académicos presenta desafíos importantes. Por un lado, estos modelos suelen ser entrenados con datos generales que quedan desactualizados con el tiempo, lo que limita su conocimiento sobre hechos recientes o específicos. Por otro lado, tienden a *"alucinar" contenido* [5]: generan información aparentemente plausible pero incorrecta cuando no conocen la respuesta, presentando afirmaciones falsas con gran confianza. Este problema no es trivial, pues en contextos educativos una respuesta errónea o inventada puede inducir a confusión o aprendizaje incorrecto. Un ejemplo notable fue el modelo **Galactica** (2022) de Meta, diseñado para el dominio científico, que fue retirado días después de su lanzamiento tras comprobarse que generaba artículos con *referencias bibliográficas ficticias* y *contenido inexacto* [13]. Este incidente evidenció el riesgo de confiar ciegamente en un **LLM** no verificado en ámbitos donde la precisión es crítica, reforzando la necesidad de métodos para controlar y verificar la información que estos modelos producen.

En respuesta a estos retos, han surgido *técnicas híbridas* que combinan el poder generativo de los **LLM** con fuentes externas de información. Entre ellas destaca la Generación de Texto Aumentada con Recuperación (**RAG**) [7]. Esta técnica consiste en incorporar en tiempo real datos de un *repositorio de conocimiento externo* (por ejemplo, una base de documentos académicos o legales) en el proceso de generación de texto. De este modo, antes de producir una respuesta final, el modelo

recupera *fragmentos relevantes* de información y los utiliza como *contexto adicional*. Esto permite anclar las respuestas del **LLM** en contenido verídico y actualizado, mitigando las alucinaciones y superando la limitación de conocimiento estático entrenado. En esencia, **RAG** convierte la interacción con el modelo en un *"examen de libro abierto"* en lugar de uno de memoria: el **LLM** puede acceder a *fuentes autorizadas* durante la generación, mejorando la precisión y relevancia de sus respuestas sin requerir *re-entrenamiento completo*. Diversos proveedores y proyectos han incorporado esta idea; por ejemplo, los sistemas de *chatbots* modernos en el ámbito empresarial utilizan **RAG** para brindar respuestas apoyadas en la *documentación interna* de la organización, garantizando que el asistente virtual responda con datos veraces y específicos del dominio.

Considerando este panorama, el presente Trabajo de Fin de Máster se enfoca en desarrollar un asistente académico inteligente que aproveche **LLM** avanzados junto con **RAG** para ofrecer soporte efectivo en tareas de estudio. En particular, nos proponemos comparar varios modelos de última generación, desde **GPT-4o** [10], referente comercial de alto rendimiento, hasta *modelos abiertos* recientes como **Llama 3.1 70B Instruct** de Meta [4] o **Mixtral 8x22B** [8], con el fin de determinar cuál ofrece el mejor equilibrio entre calidad de respuestas y viabilidad de implementación en un entorno académico. La motivación principal es identificar si alternativas de código abierto pueden aproximarse al desempeño de **GPT-4o** al estar reforzadas con recuperación de información, lo que tendría implicaciones prácticas importantes (por costos y privacidad de los datos, entre otras ventajas). Asimismo, se incluye en el estudio el modelo **Gemini 1.5 Pro** de Google [3], representante de la siguiente generación de *LLM multimodales*, con el objetivo de evaluar las tendencias más recientes de la industria.

El trabajo se centra en la evaluación y comparación directa de varios modelos de lenguaje de gran tamaño (**LLM**) operando dentro de una arquitectura de Generación Aumentada por Recuperación (**RAG**) que ha sido implementada como parte de esta investigación. Se evalúan todos los modelos seleccionados (**GPT-4o**, **Gemini 1.5 Pro**, **Mixtral 8x22B Instruct** y **Llama 3.1 70B Instruct**) en tareas académicas genéricas, tales como la síntesis de textos (resúmenes), la generación de preguntas a partir de materiales educativos y la elaboración de respuestas fundamentadas. Para ello, se utiliza un sistema **RAG** que integra un conjunto de conocimientos específico representativo de diversas asignaturas. De esta forma, se puede observar y comparar el rendimiento, la fiabilidad y las limitaciones de cada LLM cuando se le proporciona contexto relevante, permitiendo determinar su idoneidad para actuar como un asistente académico general en un escenario de estudio que requiere precisión y el uso de información de un *dominio de conocimiento específico* proporcionado por el RAG.

En resumen, esta investigación busca sentar las bases para un asistente académico apoyado por inteligencia artificial, evaluando comparativamente las capacidades de diferentes **LLM** y demostrando cómo la incorporación de recuperación de información puede solventar sus deficiencias.

CAPÍTULO 3

Objetivos

El objetivo general de este Trabajo es desarrollar y evaluar un asistente académico basado en **RAG** que aproveche las fortalezas de los **LLM** para brindar apoyo en tareas de estudio. Para lograr este propósito, se definen los siguientes objetivos específicos:

1. **Implementar un sistema de Generación Aumentada por Recuperación (RAG)** como base para un asistente académico, desarrollando un pipeline que permita la ingesta de conocimiento de diversas áreas académicas, su indexación para búsqueda semántica y la recuperación eficiente de contexto relevante para las consultas de los usuarios.

2. **Evaluar y comparar exhaustivamente el rendimiento** de distintos modelos de lenguaje de gran tamaño (**GPT-4o**, **Gemini 1.5 Pro**, **Mixtral 8x22B Instruct** y **Llama 3.1 70B Instruct**) cuando se integran con la arquitectura RAG implementada. Esta evaluación se realizará en tareas académicas genéricas, incluyendo la generación de *resúmenes fieles*, la propuesta de *preguntas relevantes* y la *elaboración de respuestas fundamentadas*.

3. **Analizar cualitativamente** la capacidad de cada configuración LLM+RAG, midiendo la *precisión factual*, la *calidad de síntesis*, la *coherencia narrativa* y la *corrección gramatical y de estilo* de las respuestas generadas. Se investigará la efectividad de cada modelo en el *uso del contexto RAG* y su propensión a generar *alucinaciones* o información incorrecta.

4. **Evaluar cuantitativamente** la *viabilidad práctica* de cada solución LLM+RAG, considerando el *coste económico* asociado al uso de sus API y la *eficiencia temporal*, incluyendo los tiempos de inferencia de los modelos y la latencia introducida por el sistema RAG.

5. **Identificar las fortalezas, debilidades y el balance óptimo** entre rendimiento cualitativo, fiabilidad, coste y eficiencia para cada LLM dentro del

marco RAG propuesto. El objetivo es determinar la idoneidad de cada configuración para un asistente académico general y ofrecer recomendaciones fundadas para la selección del modelo más adecuado según diferentes prioridades.

Cumplir con estos objetivos permitirá no solo construir un *prototipo funcional* y realizar una *evaluación detallada* de un asistente académico inteligente basado en RAG, sino también generar *conocimiento comparativo valioso* sobre las *tecnologías LLM de vanguardia* y su interacción con sistemas de recuperación de información en un contexto de uso práctico y académico. En última instancia, el trabajo proporcionará recomendaciones sobre cómo aprovechar de manera óptima los modelos de lenguaje y la *recuperación de información* para mejorar *procesos de aprendizaje* y apoyo al estudio.

CAPÍTULO 4

Metodología

Enfoque general:

Para abordar los objetivos planteados, se ha seguido una metodología experimental centrada en la implementación y evaluación de una arquitectura de sistema basada en *Retrieval-Augmented Generation* (**RAG**) para comparar el rendimiento de varios modelos de lenguaje. A continuación, se describen los pasos seguidos, así como las herramientas empleadas y los criterios de evaluación definidos.

1. Preparación del corpus y base de conocimiento:

Dado que la arquitectura **RAG** requiere un repositorio de información del cual recuperar contenido, el primer paso consistió en recopilar un conjunto de documentos académicos y material de estudio relevante. Esto incluyó artículos y apuntes de diversas materias (Biología, Física, Historia, y Lengua y Literatura), que fueron utilizados para las tareas de evaluación como la síntesis de textos, la generación de preguntas y la respuesta a cuestiones específicas. Estos documentos fueron procesados (limpieza, segmentación en fragmentos o *chunks*) y almacenados de forma estructurada para facilitar su posterior indexación y recuperación.

2. Indexado semántico y recuperación eficiente:

Con el *corpus* textual previamente procesado, se construyó un sistema de *búsqueda híbrida* que combina técnicas de *búsqueda semántica por vectores* y *búsqueda léxica* clásica.

En lo referente a la *búsqueda semántica*, se generaron *embeddings vectoriales* de 1536 dimensiones para cada fragmento textual. Esta tarea se llevó a cabo empleando el modelo `text-embedding-3-small`, accesible mediante la plataforma `Azure OpenAI Service`. Se configuró un cliente de Azure OpenAI para interactuar con el

endpoint específico del servicio y, a través de su API, se enviaron los fragmentos textuales en lotes para obtener sus correspondientes representaciones vectoriales. Estos vectores resultantes capturan el *significado semántico*, lo que posibilita la recuperación de documentos basada en la *similitud de contenido*. Los vectores se indexaron mediante `FAISS`, una librería desarrollada por Facebook AI para búsquedas eficientes en grandes espacios vectoriales.

Por otro lado, se implementó una indexación basada en `BM25` (mediante la librería `rank-BM25`) para realizar *búsquedas por palabras clave*, lo que asegura la recuperación de textos relevantes aunque no compartan una semántica global (especialmente útil para términos técnicos o expresiones normativas específicas).

Posteriormente, los fragmentos candidatos de ambas búsquedas se fusionan y se reordenan utilizando un modelo de *reranking* (`cross-encoder/ms-marco-MiniLM-L-12-v2`), que evalúa con mayor precisión la relación entre la consulta y el contenido. Este paso es crucial para seleccionar los fragmentos más pertinentes que conformarán el contexto final. En tiempo de consulta, el sistema **RAG** ejecuta ambas búsquedas (semántica y léxica), combina los resultados y los prioriza mediante *reranking* antes de la selección final de fragmentos para el **LLM**.

3. Integración con modelos LLM:

Los cuatro modelos de lenguaje candidatos – **GPT-4o**, **Mixtral 8x22B Instruct**, **Llama 3.1 70B Instruct** y **Gemini 1.5 Pro** – fueron integrados como *módulos generadores* de texto que operan sobre el contexto recuperado.

Para ello se definió un *esquema de prompting* consistente: dada una consulta del usuario (por ejemplo, "Resumir el artículo X", o "¿Qué implica la ley Y en el tema Z?"), el sistema primero recupera y reordena los fragmentos de texto más relevantes del índice, como se describió anteriormente. A partir de esta lista reordenada, se selecciona un número de fragmentos, k, para conformar el *contexto de apoyo*. Es importante destacar que, en lugar de un valor fijo para k (e.g., $k = 3$), el sistema implementa una estrategia de k **dinámico**. Este enfoque permite ajustar la cantidad de contexto proporcionado al **LLM** basándose en la puntuación de relevancia asignada por el modelo de *reranking* a cada fragmento. Típicamente, se seleccionan aquellos fragmentos que superan un umbral de confianza predefinido (`RERANKER_SCORE_THRESHOLD`), respetando además unos límites configurables (un mínimo, `MIN_CHUNKS_DYNAMIC`, para asegurar suficiente información, y un máximo, `MAX_CHUNKS_DYNAMIC`, para evitar la sobrecarga informativa o el ruido). Esta flexibilidad permite adaptar de forma más granular la cantidad de información al **LLM** según la naturaleza de la consulta y la calidad del contenido recuperado.

Estos k fragmentos seleccionados se concatenan y se incluyen junto a la pregunta del usuario en el *prompt* proporcionado al modelo. De esta forma, el **LLM** recibe no sólo la pregunta sino también *información factual* concreta y altamente relevante sobre la cual basar su respuesta.

En la práctica, para **GPT-4o** y **Gemini 1.5 Pro** – al ser modelos de tipo *caja negra* alojados en servicios externos – la interacción se realizó mediante sus respec-

tivas *APIs* (OpenAI y Google Cloud), enviando el prompt construido y recibiendo la respuesta generada. En el caso de **Llama 3.1 70B Instruct** y **Mixtral 8x22B Instruct**, al ser *modelos de código abierto*, se accedió a ellos a través de la API de Fireworks AI, simulando un acceso estandarizado vía API para la evaluación.

4. Definición de tareas y casos de prueba:

Para llevar a cabo una comparación sistemática, se definieron tareas representativas de las funciones que desempeñaría un asistente académico:

- **Síntesis de Documentos:** Se solicitó a cada configuración LLM+RAG la generación de resúmenes concisos de textos académicos, preservando las ideas clave y con una longitud controlada.
- **Generación de Preguntas:** A partir de un contenido educativo proporcionado (contextualizado por RAG), se pidió a los modelos que formularan preguntas (tipo test o de desarrollo) relevantes para la evaluación del aprendizaje.
- **Respuesta a Preguntas de Contenido Académico:** Se diseñaron preguntas específicas basadas en el contenido de los materiales del corpus, cuya resolución requería la consulta y el razonamiento sobre los fragmentos recuperados por el sistema RAG.

Todas las tareas se presentaron a los cuatro modelos bajo condiciones idénticas (misma consulta inicial, mismo sistema RAG subyacente) para asegurar una comparación justa. El proceso de ejecución se automatizó, almacenando todas las respuestas generadas para su posterior análisis.

5. Métricas y Criterios de Evaluación

La evaluación del rendimiento de cada configuración LLM+RAG combinó un análisis cualitativo con la recopilación de datos cuantitativos:

Evaluación Cualitativa.

Realizada mediante juicio humano experto y de forma ciega respecto al modelo generador:

- *Precisión Factual y Alucinaciones:* Se realizó una verificación manual de la exactitud de la información generada por cada modelo, identificando cualquier contenido no sustentado por el contexto RAG o instancias de alucinaciones.
- *Calidad de Síntesis:* Se valoró la capacidad de los modelos para capturar las ideas centrales del texto original, así como la comprensibilidad, coherencia y adecuación de las respuestas producidas.
- *Coherencia Narrativa:* Se evaluó la estructura lógica, la fluidez y la cohesión de los textos generados, considerando la organización interna y el uso apropiado de conectores discursivos. Esta evaluación se llevó a cabo de forma ciega respecto al modelo generador.

- *Corrección Gramatical y Estilo Académico:* Se efectuó una revisión humana de la corrección lingüística. También se verificó la adecuación al estilo académico formal.
- *Uso del Contexto RAG:* Se analizó específicamente cómo cada modelo LLM integró y utilizó la información contextual proporcionada por el sistema RAG durante la generación de sus respuestas.

Evaluación Cuantitativa.

- *Coste Económico:* Se estimó el coste asociado a la ejecución de cada tarea, basándose en las tarifas de las API de los respectivos modelos y el conteo de tokens de entrada y salida.
- *Eficiencia Temporal:* Se midió el tiempo total de respuesta para cada tarea, desglosándolo en el tiempo de recuperación de información del sistema RAG y el tiempo de inferencia específico de cada LLM.

6. Análisis comparativo:

Con las métricas recopiladas, se realizó un análisis comparativo para cada tarea y métrica, identificando tendencias. Por ejemplo, se elaboraron tablas resumen de desempeño y gráficos comparativos (precisión vs. tiempo de respuesta, etc.). En base a estos resultados, se evaluó la idoneidad de cada configuración LLM+RAG para actuar como un asistente académico general, identificando sus fortalezas y debilidades relativas. Adicionalmente, se extrajeron conclusiones sobre la efectividad del sistema RAG implementado y cómo los diferentes modelos interactuaron con el contexto proporcionado.

A través de esta metodología, el proyecto combina desarrollo práctico (implementación de un sistema con múltiples componentes) con evaluación rigurosa tipo *benchmark* de distintos modelos de última generación. Esto permite extraer conclusiones sólidas sobre qué modelo y configuración resulta más eficaz como base de un asistente académico con **RAG**, así como sobre los beneficios concretos que aporta la recuperación de información en la interacción con **LLM** en contextos educativos.

CAPÍTULO 5

Estado del Arte

La presente sección revisa la literatura científica y los desarrollos tecnológicos relevantes que fundamentan este Trabajo de Fin de Máster. Se centra en dos pilares tecnológicos clave: los **Modelos de Lenguaje de Gran Tamaño (LLM)**, que proporcionan las capacidades generativas fundamentales, y la técnica de **Generación Aumentada por Recuperación (RAG)**, empleada para mejorar la precisión y relevancia contextual de dichos modelos en aplicaciones específicas como la asistencia académica. El análisis de estos componentes sienta las bases para comprender el enfoque adoptado en este trabajo para desarrollar y evaluar un **asistente académico**, comparando modelos como **GPT-4o**, **Mixtral 8×22B**, **Llama 3.1 70B Instruct** y **Gemini 1.5 Pro**, y especializándolo para el ámbito de la preparación de oposiciones.

5.1 › Modelos de Lenguaje de Gran Tamaño (LLM)

Los **Modelos de Lenguaje de Gran Tamaño (LLM)** representan la vanguardia de la inteligencia artificial en el **procesamiento del lenguaje natural (PLN)**. Construidos predominantemente sobre la arquitectura *Transformer* [12], estos modelos se entrenan con cantidades masivas de datos textuales, lo que les permite aprender patrones lingüísticos complejos, adquirir un vasto conocimiento general y desarrollar capacidades notables en tareas como la generación de texto coherente, la traducción automática, la respuesta a preguntas y la síntesis de información [2].

La evolución de los LLM ha sido rápida, con modelos cada vez más grandes y potentes. Ejemplos destacados incluyen la serie **GPT** de OpenAI (culminando en **GPT-4**, uno de los modelos evaluados en este TFM), **LaMDA** y **PaLM** de Google (precursores de **Gemini**, también incluido en el estudio por sus capacidades *multimodales* emergentes) y modelos de código abierto que han ganado tracción por su rendimiento y accesibilidad, como **LLaMA 2** de Meta [11] y **Mistral** [6], ambos considerados en este trabajo por su potencial como alternativas viables a

los modelos propietarios. Estos modelos demuestran una habilidad creciente para comprender instrucciones complejas (*zero-shot* y *few-shot learning*) y adaptarse a diversos dominios y tareas [14].

No obstante, a pesar de sus impresionantes capacidades, los LLM presentan limitaciones significativas, especialmente críticas en contextos educativos o profesionales que demandan alta fiabilidad:

1. **Conocimiento estático y desactualizado:** Los LLM se entrenan sobre corpus de datos con una fecha de corte determinada. No poseen conocimiento intrínseco de eventos o información surgida posteriormente, lo que limita su utilidad para temas recientes o dominios que evolucionan rápidamente [1].

2. **Alucinaciones y falta de verificabilidad:** Los LLM pueden generar información que suena plausible pero es fácticamente incorrecta o inventada, fenómeno conocido como *alucinación* [5]. Esto ocurre a menudo cuando el modelo carece de información específica en sus parámetros internos. La dificultad para rastrear el origen de la información generada complica la verificación de su exactitud, como se evidenció en casos como el del modelo **Galactica** [13].

3. **Conocimiento genérico frente a conocimiento específico:** El conocimiento de los LLM es predominantemente generalista. Tienen dificultades para acceder o razonar sobre información altamente especializada, propietaria o de nicho (como el contenido detallado de un temario de oposición, normativa legal específica o apuntes de una asignatura concreta), a menos que esta haya estado prominentemente representada en sus datos de entrenamiento

Estas limitaciones motivan la necesidad de enfoques que permitan a los LLM acceder y utilizar fuentes de conocimiento externas y actualizadas durante el proceso de generación, lo cual nos lleva directamente a la técnica **RAG**.

5.2 › Generación de Texto Aumentada con Recuperación (RAG) en Asistentes Educativos

La **Generación Aumentada por Recuperación (RAG)** (*Retrieval-Augmented Generation*) [7] emerge como una arquitectura particularmente prometedora para superar las limitaciones inherentes de los LLM estándar en el exigente contexto de las aplicaciones educativas. La necesidad crítica de **precisión factual**, **adherencia a un currículo específico** y **acceso a conocimiento actualizado** hace que la capacidad de RAG para fundamentar las respuestas del LLM en fuentes externas verificables sea especialmente valiosa

En lugar de depender únicamente del conocimiento generalista y potencialmente desactualizado del modelo base, un asistente educativo basado en RAG puede operar sobre una base de conocimiento curada y específica del dominio, como libros de texto, apuntes de clase, artículos científicos relevantes o, como en el caso de este TFM, temarios y apuntes educativos.

Sinergia entre RAG y las necesidades educativas

La arquitectura RAG aborda directamente varios desafíos clave de la IA en educación:

- **Fiabilidad y reducción de alucinaciones:** Al condicionar la generación de respuestas a fragmentos recuperados de un corpus autorizado (por ejemplo, el material oficial del curso), RAG mitiga significativamente el riesgo de que el asistente proporcione información incorrecta o inventada [9].
- **Conocimiento específico del currículo:** Permite que el asistente çonozcaz utilice el contenido exacto de una asignatura, módulo o temario específico, incluso si este material es reciente o muy especializado y no formaba parte del entrenamiento del modelo [15].
- **Personalización fundamentada:** RAG puede recuperar pasajes específicos relevantes para la duda concreta de un estudiante dentro del material del curso, permitiendo al LLM generar explicaciones adaptadas y contextualizadas.
- **Transparencia potencial:** La capacidad de identificar y mostrar los fragmentos de texto recuperados abre la puerta a sistemas que citen sus fuentes, facilitando la verificación y el aprendizaje autónomo por parte del estudiante.

Aplicaciones específicas de RAG en asistencia educativa

- **Sistemas de preguntas y respuestas (Q&A):** RAG permite responder a preguntas sobre contenidos específicos de materiales de estudio, generando respuestas informadas con base en fragmentos pertinentes.
- **Tutores inteligentes contextualizados:** RAG puede potenciar tutores que guíen al estudiante a través de problemas, ofrezcan pistas o generen ejercicios alineados con los objetivos del curso.
- **Asistentes de estudio y síntesis:** Pueden ayudar a resumir capítulos, extraer ideas clave o realizar comparaciones entre conceptos, siempre basándose en el corpus proporcionado.
- **Soporte en la preparación de exámenes y evaluaciones:** En contextos educativos generales, RAG puede asistir en la preparación para diversos tipos de evaluaciones. Permite generar respuestas a dudas sobre el temario, crear preguntas de repaso y ofrecer explicaciones rigurosas basadas en el material de estudio específico.

Desafíos y consideraciones en el contexto educativo

- **Calidad y curación del corpus:** La efectividad de RAG depende de que el corpus esté bien estructurado, sea exhaustivo y esté libre de ambigüedades o errores.
- **Adecuación pedagógica:** El sistema debe generar respuestas claras, adaptadas al nivel del estudiante y pedagógicamente útiles, no solo correctas desde un punto de vista factual.

- **Riesgo de sobre-confianza y uso pasivo:** Es importante que el diseño del sistema fomente el aprendizaje activo y el pensamiento crítico, evitando que el asistente se convierta en una "muleta" intelectual.
- **Evaluación educativa rigurosa:** Es necesario ir más allá de métricas estándar de NLP para evaluar su impacto real en el aprendizaje, motivación y comprensión del alumnado.
- **Integración multimodal:** Muchos materiales educativos incluyen elementos visuales; RAG debe evolucionar hacia una integración efectiva de texto, imagen y video.

Conclusión parcial para el estado del arte

En resumen, la arquitectura **RAG** representa un avance clave para desarrollar asistentes educativos con base en LLM que sean fiables, contextualizados y alineados con contenidos curriculares. Las aplicaciones potenciales van desde sistemas de preguntas hasta tutores adaptativos o herramientas de síntesis, pero su implementación exige superar importantes desafíos técnicos y pedagógicos. Este TFM contribuye al campo investigando la viabilidad y comparando el rendimiento de diferentes LLM potenciados con RAG en un caso de uso educativo.

CAPÍTULO 6

Desarrollo e Implementación del Sistema RAG

En este capítulo se detalla la implementación técnica del asistente académico basado en el paradigma de **Generación Aumentada por Recuperación (RAG)**, describiendo la arquitectura del sistema, los componentes clave y las decisiones de diseño adoptadas. El objetivo es construir un sistema capaz de mitigar las limitaciones intrínsecas de los Modelos de Lenguaje Grandes (LLM) —como la generación de información no verificada (alucinaciones) y la falta de conocimiento específico o actualizado— mediante la integración de una base de conocimiento externa y relevante. La arquitectura general, ilustrada en la **Figura 6.1**, sigue un flujo donde la consulta del usuario desencadena un proceso de recuperación de información contextual que fundamenta la respuesta final generada por el LLM.

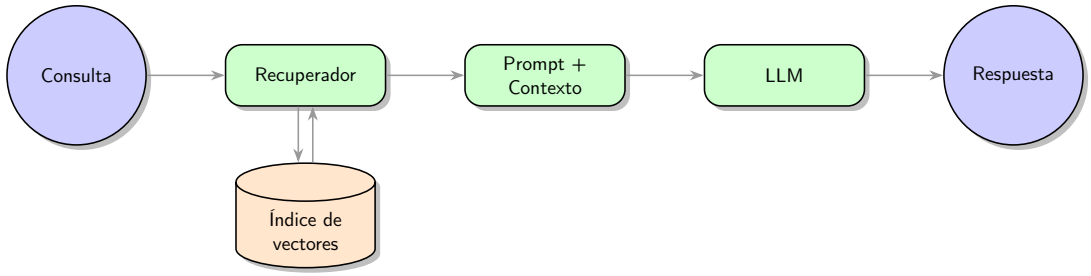

Figura 6.1: Esquema funcional del sistema RAG implementado. El proceso sigue los pasos: (1) Recepción de la consulta del usuario; (2) Recuperación de fragmentos (*chunks*) relevantes del corpus ; (3) Construcción de un *prompt* enriquecido con el contexto recuperado; (4) Generación de la respuesta por parte del LLM, condicionada por el contexto proporcionado.

6.1 › Preparación y Procesamiento del Corpus de Conocimiento

El pilar fundamental de cualquier sistema de *Retrieval Augmented Generation* (RAG) es la **base de conocimiento externa** que complementa las capacidades del modelo de lenguaje. La calidad y pertinencia de esta base son directamente proporcionales a la eficacia y fiabilidad del sistema. Por ello, la primera etapa crítica del desarrollo consistió en la preparación exhaustiva del **corpus de documentos académicos** que alimentaría al asistente educativo. Para este trabajo, se seleccionaron textos de referencia en formato PDF correspondientes a las materias de Biología, Física, Historia y Lengua Castellana, cubriendo un espectro representativo del material didáctico habitual en contextos educativos.

6.1.1 › Ingesta y Limpieza de Documentos PDF

La extracción de texto directamente de documentos PDF es un desafío conocido debido a la variabilidad de su estructura y la presencia de artefactos de formato que no son informativos para una base de conocimiento textual. La **ingesta** de estos documentos requirió la implementación de un módulo robusto de **extracción y limpieza de texto**. Se utilizó la librería `PyMuPDF` (`fitz`) por su eficiencia en la extracción textual a nivel de página, sirviendo como base para el posterior procesamiento.

La función `extract_and_clean_pdf_smart()` , desarrollada específicamente para este proyecto aplicó un conjunto de **heurísticas avanzadas** y reglas de limpieza para depurar el contenido:

- **Filtrado de Páginas Irrelevantes:** Se implementaron detectores heurísticos para identificar y omitir automáticamente páginas estructurales como portadas, tablas de contenido (índices), y secciones de bibliografía. Esto evita la introducción de ruido y contenido no sustantivo en la base de conocimiento. La detección de índices, por ejemplo, se basó en patrones de numeración jerárquica y palabras clave, mientras que las portadas se identificaron por su escaso contenido textual y posición inicial.
- **Detección de Contenido Estructurado:** Se identificó la presencia de elementos no textuales o con formato especial, cruciales en contextos científicos o académicos:
  - **Fórmulas LaTeX:** Detección de patrones de LaTeX (`$...$`, `$$...$$`), `\(...\)`, `\[...\]`, y entornos como `\begin{equation}`), permitiendo registrar su existencia y contenido para futuras estrategias de recuperación o contextualización, aunque su contenido literal no se use directamente para la incrustación.
  - **Expresiones Matemáticas Heurísticas:** Reconocimiento de patrones que sugieren la presencia de expresiones matemáticas sin formato LaTeX explícito (e.g., subíndices, superíndices, símbolos griegos comunes, notación funcional como `f(x)`). Este metadato es valioso para inferir el tipo de contenido del fragmento.

- **Regiones de Imágenes:** Detección de *bounding boxes* de imágenes y dibujos vectoriales, asociando metadatos de su ubicación y tamaño. Esto permite, en futuras iteraciones, considerar la información visual indirectamente o excluir fragmentos predominantemente visuales si no son pertinentes para la recuperación textual.

- **Normalización y Limpieza Textual:** El texto extraído de cada página, una vez filtrado y antes de la segmentación en *chunks*, fue sometido a un proceso de limpieza adicional mediante la función personalizada `clean_pdf_text_robust()`. Este proceso fue crucial para transformar el texto en un formato altamente consistente y legible, optimizado para el procesamiento posterior, especialmente de cara a la generación de *embeddings* y su uso en un sistema de Recuperación Aumentada por Generación (RAG).

  Los pasos clave implementados en la función `clean_pdf_text_robust()` incluyeron un enfoque multifásico para asegurar la máxima calidad del texto de entrada:

  - **Reconstrucción de Palabras Fragmentadas por Guiones:** Se aplicó un mecanismo para identificar y unir palabras que aparecían divididas por un guion al final de una línea y continuaban en la siguiente (ej., "aplicación" se transformó en "aplicación"). Este proceso se repitió para cubrir casos adyacentes.
  - **Eliminación de Elementos Redundantes o Irrelevantes (General):**
    - **Numeración de Página:** Se implementaron reglas para detectar y eliminar diversas formas de numeración de página, como aquellas que comienzan con la palabra "Página" seguida de números (opcionalmente con un total de páginas, como "Página X de Y"), así como formatos numéricos simples del tipo "X / Y".
    - **URLs y Direcciones de Correo Electrónico:** Se utilizaron patrones para identificar y suprimir direcciones web completas (tanto las que comienzan con `http://` o `https://` como las que inician con `www.`), y direcciones de correo electrónico, que constituyen ruido para el análisis semántico del contenido principal.
  - **Aplicación de Reglas de Limpieza Específicas del Documento:** Se diseñaron e implementaron reglas a medida, basadas en la observación empírica de los documentos fuente, para eliminar patrones de ruido recurrentes y únicos. Estas reglas se encargaron de:
    - Suprimir URLs pertenecientes a dominios específicos identificados como no relevantes (ej., enlaces a `opo.cl`) y menciones directas a ciertos dominios web (ej., `opositatest.com`).
    - Eliminar indicadores de versión de software o documentos (ej., cadenas como `vX.Y.Z`).
    - Borrar números de teléfono formateados de una manera particular (ej., aquellos precedidos por el prefijo `+34`).
    - Identificar y eliminar líneas completas que funcionaban como cabeceras o pies de página específicos del temario (ej., aquellas que contenían la palabra "TEMARIO", "ORGANIZACIÓN DEL ESTADO",

o frases promocionales como “Accede a los recursos...”, “Comprueba si tu temario...”).
    - Suprimir bloques de texto extensos que describían la utilidad de iconos o secciones estructurales del documento (ej., párrafos introductorios sobre “RECURSOS GRÁFICOS” o “PLAZOS” que finalizaban con expresiones como “...simple vistazo.”).
    - Eliminar líneas que contenían únicamente un número (potencialmente numeración de esquemas o listas internas) o títulos aislados asociados a elementos gráficos (ej., “PLAZOS”, “Destacados”, “Pregunta de examen”).

  La aplicación de estas reglas específicas fue vital para reducir el ruido contextual y mejorar la relevancia de los *chunks* generados.
- **Normalización Final de Espacios y Saltos de Línea:** Una vez depurado el contenido principal, se estandarizó su formato:
    - **Espacios Horizontales:** Múltiples espacios consecutivos, tabulaciones y otros caracteres de espaciado horizontal se consolidaron en un único espacio.
    - **Espacios alrededor de Saltos de Línea:** Se eliminaron espacios superfluos inmediatamente antes y después de los saltos de línea.
    - **Reducción de Saltos de Línea Múltiples:** Secuencias de tres o más saltos de línea consecutivos se redujeron a solo dos, para estandarizar la separación entre párrafos.
    - **Líneas Vacías Residuales:** Se suprimieron líneas completamente vacías que pudieran haber quedado como residuo de otras transformaciones.
    - **Normalización de Viñetas:** Diversos caracteres utilizados como inicio de lista o viñeta (ej., `-`, •, `*`, `o`, », ·) se unificaron a un formato estándar (`-`) para una representación consistente.
- **Eliminación de Caracteres de Control Nulos o No Imprimibles:** Se procedió a eliminar caracteres de control (códigos ASCII no visibles) que a menudo se introducen durante la extracción de texto de PDFs y pueden causar problemas en etapas posteriores del procesamiento.
- **Limpieza de Bordes del Texto y Estandarización de Terminación:** Finalmente, el texto resultante fue despojado de cualquier espacio en blanco al inicio o al final. Además, se aseguró que cada segmento de texto procesado concluyera con exactamente dos saltos de línea, facilitando una segmentación consistente y uniforme para el modelo de *embedding*.

Este enfoque meticuloso y multifacético en la limpieza garantizó que el texto de entrada para las fases de *embedding* y RAG fuera lo más puro, consistente y semánticamente denso posible, minimizando la introducción de información irrelevante o malformada que podría degradar el rendimiento del sistema de recuperación de información.

El resultado de este preprocesamiento fue un `cleaned_text` por cada documento, depurado y focalizado exclusivamente en el contenido académico sustantivo, junto con metadatos estructurados sobre la presencia de elementos clave como fórmulas e imágenes.

6.1.2 › Segmentación del Texto en Chunks

Tras la obtención del texto limpio, el siguiente paso fue su segmentación en fragmentos más pequeños y manejables, denominados ***chunks***. Esta división es esencial en sistemas RAG por varias razones:

- **Optimización del Contexto del LLM:** Los LLM tienen una ventana de contexto limitada. Los *chunks* aseguran que la información recuperada sea concisa y quepa dentro de esta ventana, minimizando el ruido irrelevante.
- **Precisión en la Recuperación:** Permite que el sistema de recuperación (*retriever*) identifique y devuelva únicamente las porciones de información más pertinentes a una consulta específica, en lugar de documentos enteros que podrían contener grandes volúmenes de texto no relacionado.

Se empleó la función `chunk_text_by_sentences()`, que utiliza `NLTK` para identificar los límites de las oraciones. Esta estrategia basada en oraciones es preferible a la segmentación por caracteres o párrafos simples, ya que ayuda a preservar la coherencia semántica de cada fragmento. Los *chunks* fueron configurados con un tamaño objetivo de aproximadamente **250 palabras** (`max_words`) y manteniendo un **solapamiento** de **2 oraciones** (`overlap_sentences`) entre fragmentos consecutivos. El solapamiento es crucial para asegurar que la información contextual no se pierda en los límites de los *chunks*, facilitando una transición suave y una recuperación más robusta de información que podría abarcar dos fragmentos.

6.1.3 › Generación y Asociación de Metadatos por Chunk

Finalmente, cada *chunk* generado fue enriquecido con metadatos relevantes (`chunks_with_metadata`). Estos metadatos no solo incluyen el texto del *chunk* y su fuente (`source`), sino también las propiedades detectadas durante la fase de limpieza:

- `has_latex_formula`: Booleano que indica si el *chunk* contiene al menos una fórmula LaTeX explícita.
- `has_heuristic_math`: Booleano que señala la presencia de patrones matemáticos heurísticos.
- `math_sections`: Un diccionario disperso que, si no está vacío, detalla los tipos de secciones matemáticas formales (e.g., teorema, definición, lema) que se inician o se mencionan dentro del *chunk*.

Aunque en esta implementación el uso principal de estos metadatos es la trazabilidad y la categorización, ofrecen un potencial significativo para futuras extensiones del sistema RAG, como la implementación de estrategias de recuperación híbridas (por ejemplo, priorizando *chunks* con contenido matemático para consultas de física, o ajustando el *prompt* del LLM cuando se detectan teoremas).

Este enfoque de **conocimiento externo y granular** (los *chunks*) es clave para las ventajas de RAG: permite al LLM consultar información específica bajo demanda, en lugar de depender únicamente de su conocimiento pre-entrenado. Al asegurar que los fragmentos son limpios, relevantes y contextualizados mediante metadatos, se reduce el ruido y se enfoca la generación de respuestas en contenido verificado

y pertinente, mitigando así las alucinaciones y asegurando la **alineación** con el temario de estudio. La granularidad y la riqueza de los metadatos obtenidos en esta etapa son fundamentales para la eficiencia y la precisión de las fases subsiguientes de incrustación y recuperación.

6.1.4 › Orquestación del Procesamiento del Corpus.

Orquestación del Procesamiento del Corpus.

La fase inicial de procesamiento del corpus, que abarca la extracción de texto, su limpieza, la segmentación en fragmentos y el enriquecimiento con metadatos, es de una criticidad fundamental para la calidad y la eficiencia de todo el sistema RAG. Para asegurar la coherencia y la eficiencia de esta preparación, se implementó un script integrador que orquesta de forma automatizada todas estas etapas clave.

Este script invoca primero la función `extract_and_clean_pdf_smart`, encargada de la extracción robusta y la limpieza inicial del texto de los documentos PDF, eliminando contenido irrelevante y artefactos de formato. Posteriormente, el texto depurado es segmentado en fragmentos manejables (*chunks*) mediante la función `chunk_text_by_sentences`, una estrategia que preserva la coherencia semántica y optimiza el tamaño del contexto para el LLM.

Finalmente, para cada uno de estos *chunks* segmentados, el script aplica diversas funciones especializadas para la detección de características, como la presencia de fórmulas LaTeX explícitas (`detect_formulas_in_text`), patrones matemáticos heurísticos (`detect_heuristic_math_expressions`) o la identificación de secciones formales como teoremas y definiciones (`detectar_secciones_matematicas`). Estas detecciones enriquecen cada fragmento con metadatos descriptivos sobre su contenido, tal como se detalló previamente en esta sección.

Este proceso de orquestación garantiza que el corpus de conocimiento esté meticulosamente preparado: los documentos se transforman en una lista de *chunks* (`chunks_with_metadata`) depurados, semánticamente coherentes y ricos en metadatos, listos para la subsiguiente fase de vectorización e indexación en el sistema FAISS.

6.2 › Vectorización Semántica e Indexado Eficiente del Corpus

Para que el sistema pueda buscar eficientemente dentro de la vasta colección de *chunks* textuales, es fundamental transformarlos en un formato que permita comparar su **significado semántico** con la consulta del usuario. Este proceso se conoce como **vectorización** o generación de *embeddings*. Para esta tarea, se empleó el modelo `text-embedding-3-small`, accesible a través del servicio **Azure OpenAI Service**. Este modelo es reconocido por ofrecer un excelente equilibrio entre la calidad de las representaciones semánticas generadas y la eficiencia computacional. Cada *chunk* de texto se convirtió en un vector numérico de alta densidad (1536 dimensiones) que encapsula su contenido semántico.

Estos vectores de *embeddings* se almacenaron y organizaron posteriormente en un índice optimizado para búsquedas rápidas por similitud. Se optó por la biblioteca **FAISS** (Facebook AI Similarity Search) debido a su alta escalabilidad y rendimiento contrastado en la gestión de grandes volúmenes de vectores. Concretamente, se utilizó un índice de tipo `IndexFlatL2`, el cual realiza búsquedas basadas en la distancia Euclidiana (L2) para identificar los vectores del corpus más cercanos (y, por tanto, semánticamente más similares) al vector de la consulta.

El procedimiento de indexación, se implementó mediante un procesamiento por lotes (*batches*). Esta estrategia permite gestionar de forma eficiente tanto el uso de memoria como las llamadas al servicio de Azure para la generación de *embeddings*. Como se observa en el código, cada lote de *chunks* textuales (variable `batch_texts`) es procesado por la función `embed_batch` (que internamente utiliza el cliente de Azure OpenAI) para obtener sus correspondientes *embeddings* (variable `batch_embs`). De forma paralela, los textos originales (almacenados en `all_texts_indexed`) y sus metadatos asociados (en `all_metas_indexed`) se guardan en archivos serializados con Pickle (`.pkl`), específicamente `educacion_texts.pkl` y `educacion_metas.pkl`. Esta práctica facilita la recuperación del contenido textual íntegro una vez que la búsqueda en FAISS identifica los índices de los *chunks* más relevantes.

Al finalizar este proceso de vectorización e indexación, se dispone de un índice vectorial, `educacion.index`, que permite realizar búsquedas semánticas de alta eficiencia sobre el corpus académico previamente procesado y segmentado.

6.3 › Módulo de Recuperación Híbrida de Contexto (FAISS + BM25)

Una vez indexado el corpus, el siguiente componente crucial es el **módulo de recuperación**, cuya función es, dada una consulta del usuario, identificar y extraer los *chunks* más relevantes de la base de conocimiento. Para lograr una recuperación robusta que capture tanto la **similitud semántica** como la **relevancia léxica** (coincidencia de palabras clave), se implementó una estrategia de **búsqueda híbrida**. Esta estrategia combina los resultados de dos enfoques complementarios:

1. **Búsqueda Vectorial (FAISS):** Utiliza el *embedding* de la consulta para en-

contrar los *chunks* cuyos vectores sean más cercanos en el espacio semántico (usando el índice FAISS creado previamente). Esto es eficaz para encontrar información conceptualmente relacionada, incluso si no usa las mismas palabras exactas que la consulta.

2. **Búsqueda Léxica (BM25):** Emplea el algoritmo Okapi BM25 (a través de la librería `rank-bm25`), un método clásico y efectivo de recuperación de información basado en la frecuencia de términos (TF-IDF), para encontrar *chunks* que contengan las palabras clave más importantes de la consulta. Esto es particularmente útil para consultas con términos técnicos, nombres propios o expresiones específicas.

La función `my_hybrid_rerank_retriever` orquesta este proceso. Primero, obtiene el *embedding* de la consulta y realiza búsquedas paralelas en FAISS y BM25, recuperando un número predefinido de candidatos iniciales (`K_FAISS_INITIAL` y `K_BM25_INITIAL`). Los resultados de ambas búsquedas (listas de IDs de *chunks* y sus respectivas puntuaciones de similitud/relevancia) se combinan.

6.3.1 › Fusión Híbrida y Ponderación Dinámica de Resultados

Dado que las puntuaciones de FAISS (basadas en distancia L2 invertida) y BM25 (basadas en relevancia TF-IDF) no son directamente comparables, se aplica una **normalización Min-Max** a cada conjunto de puntuaciones para escalarlas a un rango aproximado de [0, 1]. Luego, estas puntuaciones normalizadas se combinan linealmente utilizando **pesos dinámicos** (`peso_emb` y `peso_bm25`), determinados por la función `calcular_pesos_dinamicos` . Esta función implementa una serie de heurísticas para analizar la consulta del usuario y adaptar la ponderación entre la búsqueda semántica y la léxica, como se detalla a continuación. El resultado es una única lista de *chunks* candidatos, ordenados por una puntuación híbrida combinada, de la cual se seleccionan los `K_RERANK` mejores para el siguiente paso de *reranking*.

La función `calcular_pesos_dinamicos` es fundamental para la efectividad de la búsqueda híbrida y será esencial en la implementación de `my_hybrid_rerank_retriever` , ya que intenta inferir la *intención* del usuario a partir de la formulación de su pregunta. Partiendo de unos pesos base con un ligero sesgo hacia la búsqueda semántica (e.g., `peso_bm25 = 0.4` , `peso_emb = 0.6`), la función aplica una cascada de reglas heurísticas para ajustar estas ponderaciones.

- **Indicadores de Alta Especificidad:** Se prioriza fuertemente BM25 (e.g., `peso_bm25` entre 0.75 y 0.85) si la consulta contiene indicadores de alta especificidad. Esto incluye la presencia de *citas exactas* (texto entrecomillado), la solicitud de definición de un *término clave muy concreto* (detectado mediante patrones como "qué es X" donde X es un término corto), la mención explícita de *leyes, artículos, o teoremas* con identificadores numéricos o nominales, o la detección de patrones que sugieren *fórmulas o ecuaciones*, especialmente en asignaturas científicas. Estas reglas buscan asegurar la recuperación de fragmentos que contengan coincidencias literales o estructurales precisas.
- **Indicadores de Especificidad Media:** Para consultas que, sin ser tan restrictivas, contienen elementos específicos, BM25 recibe un peso moderadamente alto (e.g., `peso_bm25` entre 0.60 y 0.70). Esto ocurre ante la detección de

nombres propios (identificados por mayúsculas iniciales), *fechas, años o siglos*, o *acrónimos* y términos técnicos formados por mayúsculas. En estos casos, aunque la semántica sigue siendo importante, la presencia exacta del término específico es un fuerte indicador de relevancia.

- **Búsqueda de Definiciones Generales:** Si la consulta parece una petición de definición general (e.g., "definición de X", donde X es un concepto más amplio y no un término ultracorto ya capturado antes), y no se han activado reglas de alta especificidad, los pesos pueden ajustarse para dar un ligero predominio a BM25 (e.g., `peso_bm25 = 0.55`) buscando la literalidad del término a definir, pero manteniendo un componente semántico significativo.
- **Indicadores de Conceptualidad:** Por el contrario, si la consulta utiliza palabras clave que denotan una búsqueda de comprensión, análisis o explicación (e.g., "explica cómo...", "analiza las causas de...", "compara y contrasta...", "cuál es la importancia de..."), se incrementa el peso de la búsqueda semántica (`peso_emb` puede llegar a 0.75 o más). Esto se debe a que la recuperación debe priorizar la similitud de significado global sobre la coincidencia exacta de palabras. Si la pregunta conceptual se refiere a un término previamente identificado como específico, se busca un equilibrio para combinar la especificidad del término con la naturaleza explicativa de la consulta.
- **Ajustes Contextuales:** Adicionalmente, la función puede realizar ajustes menores basados en el *tema o asignatura* (`subject`) si este se proporciona, y si no ha predominado una regla fuerte anterior. Por ejemplo, preguntas de análisis literario en "Lengua Castellana" podrían favorecer más la semántica, mientras que la búsqueda de reglas gramaticales específicas podría inclinar la balanza hacia BM25. Finalmente, la *longitud de la consulta* también puede influir: consultas muy largas suelen ser más conceptuales (favoreciendo embeddings), mientras que las muy cortas podrían beneficiarse de un equilibrio o un ligero sesgo hacia BM25 si buscan un término puntual.

Esta lógica multicapa permite al sistema RAG adaptar dinámicamente su estrategia de recuperación, buscando un equilibrio óptimo entre la precisión léxica y la riqueza semántica para cada consulta particular.

6.4 › Reordenamiento de Relevancia con Cross-Encoder

Si bien la búsqueda híbrida, que combina las fortalezas de métodos como BM25 y la búsqueda semántica con FAISS, proporciona un conjunto inicial de candidatos potencialmente relevantes, no todos tendrán el mismo grado de pertinencia exacta para la consulta específica del usuario. Para refinar aún más esta selección y asegurar que el contexto final proporcionado al LLM sea de la máxima calidad y precisión, se implementó una etapa de **reordenamiento (reranking)** utilizando un modelo **Cross-Encoder**.

Un **Cross-Encoder** opera de manera fundamentalmente distinta a los métodos de recuperación inicial como BM25 o los bi-encoders utilizados con FAISS. Mientras que BM25 evalúa la relevancia basándose en la frecuencia y distribución de términos (palabras clave) entre la consulta y los documentos, y los bi-encoders (como los que generan los embeddings para FAISS) procesan la consulta y cada documento de forma independiente para generar representaciones vectoriales que luego se comparan por similitud (ej. distancia L2), el Cross-Encoder realiza una interacción más profunda. Este modelo toma el par `(consulta, chunk)` como una única entrada y lo procesa conjuntamente a través de su arquitectura (generalmente basada en Transformers). Esta co-atención permite al modelo capturar interacciones semánticas mucho más finas y dependencias contextuales entre la consulta y el texto del *chunk*, resultando en una puntuación de relevancia mucho más precisa. Sin embargo, esta evaluación detallada es computacionalmente más intensiva, lo que la hace ideal para reordenar un conjunto reducido de candidatos prometedores en lugar de aplicarla a todo el corpus documental. Se seleccionó el modelo `cross-encoder/ms-marco-MiniLM-L-12-v2` de la librería `sentence-transformers`, pre-entrenado específicamente para tareas de reranking de pasajes.

El proceso, implementado en la función `my_hybrid_rerank_retriever`, toma la lista de identificadores de los *chunks* candidatos (`top_hybrid_candidates_ids`) generada por la búsqueda híbrida. Para cada identificador, se recupera el texto del *chunk* correspondiente y se forma un par con la consulta original del usuario. Todos estos pares se introducen en el modelo Cross-Encoder, el cual asigna una puntuación de relevancia a cada uno. Posteriormente, los *chunks* se reordenan en función de esta puntuación de alta precisión, en orden descendente.

Finalmente, los `K_RERANK` fragmentos (*chunks*) con mayor puntuación preliminar, obtenidos tras la fusión de los resultados de la búsqueda híbrida, son los que se someten a este proceso de *reranking* con el modelo Cross-Encoder. La selección final del número de *chunks* (k) a partir de esta lista reordenada, así como la formación del contexto para el LLM, se detalla en la Sección 6.5.

6.5 › Selección Dinámica del Número de Fragmentos (k) para el Contexto

Una vez que los *chunks* candidatos han sido reordenados por el modelo de *reranking* (`CrossEncoder`), el siguiente paso crucial es determinar cuántos de estos fragmentos (*chunks*) –denominado k– se incluirán en el contexto final que se proporcionará al Modelo de Lenguaje Grande (LLM). En lugar de utilizar un valor fijo para k, este sistema implementa una estrategia de k **dinámico**, controlada por la variable booleana `USE_DYNAMIC_K`. Este enfoque permite adaptar la cantidad de información contextual a la relevancia percibida de los fragmentos recuperados y a la naturaleza de la consulta.

El proceso de selección con k dinámico, implementado en el código, opera de la siguiente manera:

1. **Filtrado por Umbral de Confianza:** Inicialmente, se recorre la lista de *chunks* reordenados (variable `reranked_docs_info`, ordenados descendentemente por su `reranker_score`). Se seleccionan aquellos *chunks* cuya puntuación de *reranking* supera un umbral de confianza predefinido, denominado `RERANKER_SCORE_THRESHOLD`. Dado que la lista está ordenada, este proceso se detiene tan pronto como se encuentra un *chunk* que no cumple con el umbral, optimizando la selección. Los fragmentos que superan este umbral conforman un conjunto preliminar (`selected_for_dynamic_k`).

2. **Aplicación de Límites Mínimos y Máximos:** Sobre este conjunto preliminar, se aplican restricciones para asegurar que el número de *chunks* seleccionados se mantenga dentro de un rango razonable.

   - Si el número de *chunks* que superaron el umbral es inferior al mínimo configurado (`MIN_CHUNKS_DYNAMIC`), y si hay suficientes *chunks* disponibles en la lista reordenada general, el sistema selecciona los `MIN_CHUNKS_DYNAMIC` mejores *chunks* de la lista general, ignorando temporalmente el umbral para este caso, con el fin de garantizar un mínimo de contexto.
   - Si el número de *chunks* que superaron el umbral excede el máximo configurado (`MAX_CHUNKS_DYNAMIC`), el sistema trunca la selección, tomando únicamente los `MAX_CHUNKS_DYNAMIC` *chunks* con mayor puntuación de *reranking* de entre los que superaron el umbral.
   - Si el número de *chunks* que superaron el umbral se encuentra dentro del rango [`MIN_CHUNKS_DYNAMIC`, `MAX_CHUNKS_DYNAMIC`] (o si `MIN_CHUNKS_DYNAMIC` es cero y se supera el umbral), se utiliza directamente este conjunto de *chunks*.

   El resultado de este proceso es la lista final de *chunks* (`final_top_docs`) que conformarán el contexto.

3. **Alternativa con k Fijo:** Si la opción `USE_DYNAMIC_K` está desactivada, el sistema recurre a seleccionar un número fijo de *chunks*, `K_FINAL`, tomándolos directamente del inicio de la lista reordenada por el *reranker*.

La implementación de un k dinámico una vez más en `my_hybrid_rerank_retriever` permite al sistema ser más adaptable: proporciona más contexto cuando hay muchos

fragmentos altamente relevantes disponibles y, a la inversa, evita sobrecargar al LLM con información de baja relevancia o potencialmente ruidosa cuando la calidad de los fragmentos recuperados es menor.

Los parámetros `RERANKER_SCORE_THRESHOLD`, `MIN_CHUNKS_DYNAMIC`, y `MAX_CHUNKS_DYNAMIC` son configurables, lo que permite ajustar este comportamiento según las necesidades específicas de la aplicación o del dominio. Finalmente, los textos de estos `final_top_docs` seleccionados, junto con sus metadatos (como la fuente), se concatenan para formar el string de contexto que se envía al LLM.

Esta etapa de reranking es fundamental para maximizar la precisión del contexto, asegurando que el LLM reciba la información más pertinente y fiable posible para generar su respuesta.

6.6 › Construcción del *Prompt* y Generación de la Respuesta Final

Una vez recuperado y reordenado el contexto más relevante, la etapa final del sistema RAG consiste en la **construcción de un *prompt* efectivo** y la **generación de la respuesta** por parte del Large Language Model (LLM). El diseño del prompt es de importancia capital, ya que debe guiar inequívocamente al modelo para que utilice de forma estricta la información contextual proporcionada. Asimismo, debe orientar al LLM para que cumpla con la tarea específica solicitada (ya sea responder una pregunta directa, generar un resumen conciso, o crear preguntas de evaluación), manteniendo siempre el tono, formato y estilo deseados para el ámbito educativo.

6.6.1 › Plantillas de Prompt para Diversas Tareas RAG

Para conseguir una estructura de prompts versátil y adaptada a las diferentes necesidades del asistente educativo, se definieron varias plantillas utilizando la clase `ChatPromptTemplate` de la librería LangChain. Esta aproximación permite inyectar dinámicamente tanto el contexto recuperado como las variables específicas de la tarea en un prompt predefinido, facilitando la escalabilidad y el mantenimiento. Se describen sus propósitos:

- `rag_summary_prompt_template`: Diseñada para solicitar al LLM la generación de un resumen conciso sobre un tema específico (`topic`), basándose exclusivamente en el contexto recuperado (`context`). La instrucción enfatiza la necesidad de capturar las ideas clave presentadas en el contexto con un estilo académico y formal, y en un formato breve (aproximadamente 2-4 frases).
- `rag_question_gen_prompt_template`: Esta plantilla instruye al LLM para que, a partir del contexto y un tema dados, genere dos tipos de preguntas para evaluación: una pregunta de tipo test (con 4 opciones y la correcta marcada con '*') y una pregunta de desarrollo. Su objetivo es fomentar la creación de material didáctico y de evaluación que esté estrictamente fundamentado en la información contextual.

- `rag_qa_prompt_template`: Es la plantilla estándar para tareas de Pregunta-Respuesta (Q&A). En este caso, se proporciona el contexto recuperado y una pregunta específica del usuario (`question`), y se espera que el LLM genere una respuesta clara y precisa, derivada únicamente de la información disponible en el contexto para evitar alucinaciones.

Un principio subyacente y vital en todas estas plantillas es la directriz explícita de basarse *estrictamente* en el contexto proporcionado. Esta instrucción es la primera línea de defensa contra las alucinaciones del LLM, asegurando que la respuesta final sea fiel a la fuente de información recuperada, un aspecto crítico para la fiabilidad del conocimiento en un entorno educativo.

6.6.2 ❯ La Función Universal `run_rag_based_task`

Para encapsular y orquestar el flujo completo de RAG de manera reutilizable para cualquier tarea, se desarrolló la función `run_rag_based_task`. Este componente central actúa como un coordinador que integra las fases de recuperación y generación, gestionando los datos y las interacciones con el LLM. La función orquesta los dos pasos principales:

1. **Recuperación de Contexto**: Invoca la función de recuperación proporcionada (`retriever_func`) con la consulta del usuario (`user_query`). Durante esta fase, se mide el tiempo de ejecución para evaluar el rendimiento del retriever. La función maneja la salida del recuperador, asegurando que el contexto (`retrieved_context_str`) sea un string formateado adecuadamente para ser inyectado en el prompt, incluso si el retriever devuelve una lista de objetos `Document` de LangChain. Se incluyen mecanismos robustos para la gestión de errores durante la recuperación.

2. **Generación de Respuesta**: Esta fase utiliza LangChain Expression Language (LCEL) para construir una cadena de procesamiento fluida y eficiente:

   - Primero, `RunnablePassthrough.assign(...)` se encarga de preparar dinámicamente las variables de entrada para la plantilla de prompt. Esto incluye inyectar el `retrieved_context_str` y cualquier otra entrada específica de la tarea (como `topic` o `question`) que se haya proporcionado en `task_specific_input`. Cada valor se envuelve en una función lambda para asegurar su evaluación adecuada dentro de la cadena LCEL.
   - Seguidamente, la cadena pasa estos datos ya preparados a la `task_prompt_template` seleccionada. Esta plantilla se encarga de formatear el prompt final que será enviado al modelo de lenguaje.
   - El prompt formateado se envía al modelo de lenguaje (`llm`). El LLM procesa el prompt y genera una respuesta basada en las instrucciones y el contexto.
   - Finalmente, `StrOutputParser()` convierte la salida del LLM, que puede venir en un formato interno de LangChain, a un string simple y legible.

   Durante esta fase también se gestiona la medición del tiempo para evaluar la latencia del LLM, y se capturan posibles errores que puedan surgir en el proceso de generación.

Componentes Clave y Librerías Auxiliares

La robustez y flexibilidad de la función `run_rag_based_task` se construyen sobre una serie de componentes y librerías clave, que aunque no se detallan en esta sección, son fundamentales para su operación. Su implementación y uso se puede consultar en el Anexo.

`time` **(librería estándar de Python)** Utilizada para medir la duración de las fases de recuperación de contexto y generación de la respuesta por el LLM. Esto es esencial para la evaluación del rendimiento del sistema RAG, permitiendo identificar posibles cuellos de botella y optimizar la experiencia del usuario.

`traceback` **(librería estándar de Python)** Fundamental para la depuración. Se emplea en los bloques `try-except` para imprimir la traza completa de la pila de llamadas (*stack trace*) cuando ocurre un error. Esto proporciona información detallada sobre la causa y ubicación exacta del problema, facilitando su resolución.

`ChatPromptTemplate` **(de** `langchain_core.prompts`**)** Una clase central de LangChain para la creación de plantillas de prompts flexibles y estructuradas. Permite definir prompts con marcadores de posición (*placeholders*) que son rellenados dinámicamente con variables como el contexto recuperado, la pregunta del usuario o el tema, adaptando el prompt a cada interacción.

`RunnablePassthrough` **(de** `langchain_core.runnables`**)** Un componente de LangChain Expression Language (LCEL) que permite pasar entradas directamente a la siguiente etapa de una cadena, o en el caso de `assign`, inyectar nuevas claves y valores en el diccionario de entrada. Su uso con `assign(**assign_args)` es crucial para preparar todas las variables requeridas por la plantilla de prompt.

`StrOutputParser` **(de** `langchain_core.output_parsers`**)** Un analizador de salida de LangChain que se encarga de extraer la respuesta generada por el LLM y convertirla a un formato de string simple. Esto es útil para limpiar la salida del modelo de cualquier formato interno y presentarla de manera directa al usuario o a la siguiente etapa del *pipeline*.

`Document` **(de** `langchain_core.documents`**)** Una clase estándar en LangChain utilizada para representar un fragmento de texto o documento. A menudo, las funciones de recuperación (retrievers) devuelven listas de objetos `Document`, cada uno conteniendo el `page_content` (el texto) y, opcionalmente, `metadata` asociada. La función `run_rag_based_task` maneja explícitamente este tipo de objeto al procesar el contexto recuperado.

`load_dotenv` **(de la librería** `python-dotenv`**)** Aunque no se invoca directamente dentro de `run_rag_based_task`, su presencia al inicio del script es vital para cargar las variables de entorno (como las claves de API de Azure OpenAI y el endpoint) desde un archivo `.env`. Esto asegura que las credenciales sensibles no estén codificadas directamente en el código, mejorando la seguridad y la portabilidad del proyecto.

La versatilidad de `run_rag_based_task` se demuestra en la fase de experimentación (invocada repetidamente para diferentes tareas simplemente cambiando la plantilla de prompt y las entradas específicas de la tarea. Este enfoque modular no solo simplifica el desarrollo y mantenimiento del sistema, sino que también facilita la experimentación con diferentes LLMs, estrategias de recuperación y diseños de prompt, contribuyendo a la robustez y adaptabilidad general del asistente académico. El resultado es un sistema capaz de generar respuestas fundamentadas, relevantes y alineadas con los objetivos pedagógicos.

En conclusión, la implementación detallada en este capítulo demuestra la construcción de un sistema RAG funcional y robusto para la asistencia académica. Cada etapa del *pipeline* —desde la meticulosa preparación del corpus hasta la generación final condicionada por el contexto— ha sido diseñada para abordar las limitaciones inherentes de los LLM y potenciar su utilidad en el ámbito educativo:

- La **ingesta y segmentación** proporcionan conocimiento específico y granular.
- El **indexado híbrido** (vectorial + léxico) permite una recuperación versátil y eficiente.
- El **reranking** asegura la máxima relevancia del contexto entregado al modelo.
- La **Elección dinámica** de chunks finales permite adaptar la cantidad de información contextual a la relevancia percibida de los fragmentos recuperados y a la naturaleza de la consulta.
- La **ingeniería de prompts** guía al LLM para generar respuestas fundamentadas y fiables.

El sistema resultante establece un puente efectivo entre las capacidades generativas de los LLM y la necesidad de precisión y fiabilidad del conocimiento académico, sentando las bases para la evaluación comparativa de modelos que se abordará en capítulos posteriores.

CAPÍTULO 7

Experimentación y Comparativa LLM con RAG

Tras el detallado desarrollo e implementación del sistema de Generación Aumentada por Recuperación (RAG) presentado en el Capítulo 6, este capítulo se adentra en la fase experimental del presente Trabajo de Fin de Máster. El objetivo primordial es evaluar y comparar de manera sistemática el rendimiento de un conjunto selecto de Modelos de Lenguaje Grandes (LLM) de vanguardia cuando se integran con la arquitectura RAG desarrollada, en el contexto específico de la asistencia académica.

La implementación de RAG, como se ha demostrado, busca mitigar limitaciones inherentes a los LLMs, tales como la generación de información no verificada (alucinaciones) y la carencia de conocimiento específico o actualizado sobre dominios concretos. No obstante, la efectividad de esta mitigación y la calidad general del asistente académico dependen crucialmente tanto de la robustez del sistema de recuperación como de las capacidades intrínsecas del LLM subyacente. Por consiguiente, resulta imperativo llevar a cabo una evaluación empírica que permita discernir las fortalezas y debilidades de cada modelo en tareas académicas representativas.

En las siguientes secciones, se describirá en detalle el **marco metodológico** diseñado para esta fase experimental. La definición del conjunto de datos de evaluación y las tareas académicas utilizadas como *benchmarks*, y el procedimiento seguido para la orquestación de las inferencias y la recopilación sistemática de resultados. Finalmente, se sentarán las bases para el análisis comparativo –tanto cualitativo como cuantitativo– que se abordará en capítulos subsecuentes, con el fin de extraer conclusiones fundamentadas sobre la idoneidad de cada combinación LLM+RAG para el caso de uso propuesto.

7.1 › Configuración de los Modelos de Lenguaje (LLM)

La selección y configuración adecuada de los Modelos de Lenguaje Grandes (LLM) constituyen un pilar fundamental para la fase experimental. Este apartado detalla los modelos específicos elegidos para la comparación, la justificación de su selección y el proceso de inicialización implementado, enfatizando un enfoque de configuración mínima para asegurar una evaluación comparativa centrada en las capacidades inherentes de cada modelo.

7.1.1 › Selección de Proveedores y Modelos LLM

Para llevar a cabo una evaluación exhaustiva y representativa del panorama actual de los LLM, se seleccionó un conjunto diverso de modelos, combinando tanto opciones comerciales de vanguardia como modelos abiertos prominentes. Todos los modelos fueron accedidos a través de sus respectivas APIs utilizando la librería LangChain, lo que facilitó una integración homogénea dentro del sistema RAG desarrollado. Los modelos evaluados son:

- **GPT-4o (OpenAI):** Representa uno de los modelos comerciales más avanzados y con capacidades multimodales, accedido mediante la API estándar de OpenAI. Su inclusión permite establecer un punto de referencia de alto rendimiento. Se utilizó el cliente `ChatOpenAI` de LangChain.
- **Gemini 1.5 Pro (Google):** Otro modelo propietario de última generación desarrollado por Google, conocido por su amplia ventana de contexto y robustas capacidades de razonamiento. Se accedió a través del cliente `ChatGoogleGenerativeAI`.
- **Mixtral 8x22B Instruct (Mistral AI vía Fireworks.ai):** Un potente modelo abierto de tipo *Mixture-of-Experts* (MoE), conocido por su eficiencia y alto rendimiento. La versión *instruct* está optimizada para seguir instrucciones. Se accedió mediante el proveedor Fireworks.ai utilizando el cliente `ChatFireworks`.
- **Llama 3.1 70B Instruct (Meta AI vía Fireworks.ai):** La versión más grande de la familia Llama 3 disponible en el momento del estudio a través de proveedores de API, destacada por sus mejoras significativas en razonamiento y generación de código. También se accedió vía Fireworks.ai con `ChatFireworks`.

La elección de estos modelos busca cubrir un espectro variado en términos de arquitectura, tamaño, licencia (propietarios vs. abiertos) y proveedor de acceso, permitiendo así obtener una visión más completa de su desempeño en el contexto RAG.

7.1.2 › Proceso de Inicialización y Parametrización Mínima

La inicialización de cada modelo se realizó a través de su respectivo cliente de LangChain, gestionando las claves de API necesarias de forma segura mediante variables de entorno, cargadas con la librería `python-dotenv`. Un diccionario, `llm_providers`, se utilizó para almacenar las instancias de los clientes LLM, facilitando la iteración programática sobre ellos durante la ejecución de las tareas de evaluación.

Un aspecto crucial de la configuración para esta fase comparativa fue la decisión de realizar una **parametrización mínima** de los LLM. Con el objetivo de evaluar

las capacidades fundamentales de cada modelo en una configuración lo más "pura" y comparable posible, se optó por utilizar, en la medida de lo factible, los hiperparámetros por defecto establecidos por sus desarrolladores, con ajustes muy puntuales y consistentes para todos los modelos. Específicamente:

- **Temperatura (`temperature`):** Se estableció un valor bajo y consistente de `0.1` para todos los modelos. Esta elección busca promover respuestas más factuales, deterministas y menos creativas, lo cual es deseable para un asistente académico donde la precisión y la fidelidad al contenido son primordiales.
- **Máximo de Tokens de Salida (`max_tokens`):** Para los modelos accedidos vía Fireworks.ai (Mixtral 8x22B y Llama 3.1 70B Instruct), se configuró un valor de `1024` tokens. Este límite se consideró suficiente para la generación de resúmenes y respuestas a las preguntas formuladas, evitando respuestas excesivamente largas o truncadas prematuramente. Para GPT-4o y Gemini 1.5 Pro, se confió en los límites de tokens de salida por defecto, generalmente más amplios y gestionados dinámicamente por el proveedor.
- **Otros Hiperparámetros:** El resto de los hiperparámetros (e.g., *top_p*, *frequency_penalty*, *presence_penalty*) se mantuvieron en sus valores predeterminados. Esta decisión metodológica busca evitar la introducción de sesgos derivados de una optimización exhaustiva de hiperparámetros para cada modelo y tarea, lo cual excedería el alcance de este estudio y podría enmascarar las diferencias inherentes en el rendimiento base de los modelos.

Este enfoque de parametrización mínima asegura que las diferencias observadas en el rendimiento puedan atribuirse, con mayor confianza, a las arquitecturas y al entrenamiento fundamental de los LLM, en lugar de a una sintonización fina específica para este conjunto de pruebas.

La gestión centralizada en el diccionario `llm_providers` permite que el bucle de experimentación (descrito en secciones posteriores) pueda invocar de manera uniforme cada uno de los LLM configurados.

7.2 › Diseño del Conjunto de Datos de Evaluación y Tareas RAG

Una vez configurados los Modelos de Lenguaje Grandes (LLM), el siguiente paso en el diseño experimental es la definición rigurosa del conjunto de datos y las tareas específicas sobre las cuales se evaluará su rendimiento. La calidad y representatividad de estos elementos son cruciales para obtener conclusiones válidas sobre la efectividad de cada modelo dentro del sistema de Generación Aumentada por Recuperación (RAG) desarrollado.

7.2.1 › Criterios para la Creación del Conjunto de Datos de Evaluación

La construcción del conjunto de datos de evaluación se guio por la necesidad de probar el sistema RAG en escenarios realistas y con un alto grado de fidelidad respecto al contenido académico específico del corpus. Para ello, se establecieron los siguientes criterios:

- **Fundamentación en el Corpus Específico:** Todos los ítems de prueba (temas para resumir, conceptos para generar preguntas y preguntas directas) fueron meticulosamente extraídos y formulados a partir del contenido textual de los documentos PDF que conforman la base de conocimiento del sistema RAG. Esto asegura que las tareas requieran una consulta efectiva al corpus a través del mecanismo de recuperación.
- **Cobertura Multidisciplinar:** Para evaluar la generalización y robustez del sistema, se crearon conjuntos de datos específicos para cada una de las cuatro asignaturas seleccionadas en este estudio: Biología, Física, Historia de España, y Lengua Castellana y Literatura.
- **Alta Calidad y Precisión de los Ítems:** Se procuró que los temas y preguntas fueran claros, unívocos y representativos de conceptos clave dentro de cada materia.
- **Organización Estructurada:** Los ítems de prueba para cada tarea y asignatura se organizaron en estructuras de datos tabulares utilizando la librería `pandas` de Python, facilitando su iteración durante la fase de ejecución de inferencias.

7.2.2 › Definición de las Tareas de Evaluación RAG

Para medir diferentes facetas del rendimiento de los LLM integrados con RAG, se definieron tres tareas principales, cada una asociada a una plantilla de *prompt* específica (descritas en la Sección 6.6.1, Capítulo 6). Estas tareas simulan funciones típicas de un asistente académico:

1. **Resumen Basado en RAG (*RAG Summarization*):**
    - **Objetivo:** Evaluar la capacidad del LLM para sintetizar información relevante sobre un tema específico, basándose estrictamente en el contexto recuperado por el sistema RAG. Se busca medir la concisión, la fidelidad al contenido original y la coherencia del resumen.

- **Datos de Entrada:** Para cada asignatura, se definió una lista de temas o conceptos clave (e.g., `topics_for_summary_bg`, `topics_for_summary_f`, etc.). Para cada tema, la consulta al recuperador RAG se formula como "Información detallada sobre [tema]", y el LLM utiliza la plantilla `rag_summary_prompt_template`.
- **Ejemplo (Biología):** Resumir "El modelo de la doble hélice del ADN propuesto por Watson y Crick, destacando sus características estructurales clave."

2. **Generación de Preguntas Basada en RAG (*RAG Question Generation*):**
   - **Objetivo:** Medir la habilidad del LLM para comprender el contexto recuperado y, a partir de él, formular preguntas de evaluación pertinentes (una de tipo test y una de desarrollo) sobre un tema dado. Esto evalúa la capacidad de extraer conceptos importantes y transformarlos en instrumentos de evaluación.
   - **Datos de Entrada:** Similar a la tarea de resumen, se utilizaron listas de temas clave por asignatura (e.g., `topics_for_question_gen_bg`). La consulta al recuperador RAG se formula como "Detalles clave sobre [tema]", y se emplea la plantilla `rag_question_gen_prompt_template`.
   - **Ejemplo (Física):** Generar preguntas sobre "La diferencia entre sistemas de referencia inerciales y no inerciales según la primera ley de Newton."
3. **Preguntas de Contenido Académico con RAG (RAG Q&A):**
   - **Objetivo:** Evaluar la capacidad del LLM para responder de manera precisa y concisa a preguntas directas sobre hechos, definiciones o explicaciones contenidas en el corpus, utilizando el contexto proporcionado por RAG. Se mide la exactitud factual y la claridad de la respuesta.
   - **Datos de Entrada:** Se elaboró un conjunto de preguntas específicas para cada asignatura (e.g., `rag_questions_bg`). En este caso, la propia pregunta del usuario se utiliza como consulta para el recuperador RAG, y se usa la plantilla `rag_qa_prompt_template`.
   - **Ejemplo (Historia de España):** "¿Qué fue el Liber Iudiciorum (Fuero Juzgo) y qué importancia tuvo en la unificación visigoda?"

La combinación de estas tres tareas permite obtener una visión multifacética del rendimiento de los LLM, abarcando desde la síntesis y la extracción de información hasta la comprensión profunda necesaria para generar material de estudio. La preparación de estos conjuntos de datos, resultó en un número significativo de ítems por tarea y asignatura, concretamente 6 items por tarea, lo que equivale a 18 items por asignatura y por ende 72 items en total, asegurando una base sólida para la comparación.

Para ilustrar de manera concreta la naturaleza de las entradas utilizadas en la fase experimental, la Tabla 7.1 presenta una muestra de los *topics* o preguntas de entrada para cada una de las tres tareas RAG definidas, a través de las cuatro asignaturas del corpus. Estos ejemplos reflejan el tipo de estímulo que se proporcionó

al sistema para la recuperación de contexto y la subsiguiente generación de respuesta por parte de los LLM.

Asignatura	Tarea: Resumen RAG	Tarea: Generación de Preguntas RAG	Tarea: Q&A RAG (Pregunta Directa)
Biología	El modelo de la doble hélice del ADN propuesto por Watson y Crick, destacando sus características estructurales clave.	Los cuatro niveles de estructura de las proteínas (primaria, secundaria, terciaria, cuaternaria) y qué tipo de enlaces estabiliza cada uno.	Según el experimento de Miller descrito en el texto, ¿qué tipo de moléculas orgánicas fundamentales para la vida se formaron?
Física	El enunciado y significado físico de las tres Leyes de Newton.	La diferencia entre sistemas de referencia inerciales y no inerciales según la primera ley de Newton.	¿Cuál es la expresión matemática que define la Segunda Ley de Newton en su forma más general, involucrando el momento lineal (p)?
Historia de España	El proceso de Romanización en Hispania: factores que lo favorecieron y principales manifestaciones.	La organización política, económica y social de Al-Ándalus durante el Emirato y el Califato.	¿Qué fue el Liber Iudiciorum (Fuero Juzgo) y qué importancia tuvo en la unificación visigoda?
Lengua y Lit.	Las propiedades del texto: coherencia, cohesión y adecuación, según se explican en el documento.	Los elementos de la comunicación (emisor, receptor, mensaje, canal, código, contexto) y las funciones de la lengua asociadas a cada uno.	¿Cuál es la diferencia principal entre lengua, habla y norma según las definiciones dadas?

Cuadro 7.1: Ejemplos de entradas (*topics* o preguntas) por tarea y asignatura para la evaluación RAG.

Con la configuración de los modelos LLM establecida y el conjunto de datos de evaluación definido, el sistema estuvo preparado para la ejecución sistemática de las inferencias. El siguiente apartado detalla el proceso de orquestación de estas pruebas y la recopilación de los datos que servirán de base para la comparación de modelos.

7.2.3 ❱ Estructura de los Resultados Recopilados

Para el análisis detallado del rendimiento de los modelos, los datos de cada inferencia experimental (ejecutada mediante `run_rag_based_task`) se almacenaron exhaustivamente. Tras acumular los resultados de cada combinación LLM-tarea-ítem en una lista de diccionarios, estos se consolidaron en un DataFrame de `pandas` por asignatura, una estructura idónea para el análisis posterior.

Los campos clave registrados para cada instancia de inferencia, y por tanto, cada fila en el DataFrame resultante, son los siguientes:

- `"task_type"`: Cadena de texto que identifica la naturaleza de la tarea RAG evaluada, permitiendo agrupar y analizar los resultados por tipo de actividad (e.g., `"RAG Summarization"`, `"RAG Question Generation"`, `"RAG Q\&A"`).
- `"input_id"`: El identificador del ítem de prueba original, que corresponde al tema específico (para resumen o generación de preguntas) o a la pregunta formulada (para Q&A). Este campo permite rastrear el estímulo inicial de cada inferencia.

- `"llm_provider"`: Nombre o identificador único del modelo LLM que procesó la solicitud (e.g., `"gpt4o_openai"`, `"gemini_1.5_pro"`, `"mistral_8x22b_fireworks"`, `"llama3_70b_fireworks"`). Esencial para la comparación directa entre modelos.
- `"input_data"`: La consulta textual (`retriever_query`) que se utilizó efectivamente para interactuar con el módulo de recuperación de contexto del sistema RAG.
- `"output_response"`: La respuesta completa generada por el LLM en formato de cadena de texto. Este es el dato principal para la evaluación cualitativa y para muchas métricas cuantitativas de calidad.
- `"retrieved_context"`: El contenido textual completo que el sistema RAG recuperó del corpus y proporcionó al LLM como contexto para fundamentar su respuesta. La disponibilidad de este campo es **crucial** para el análisis de la efectividad del componente RAG. Permite, por ejemplo, evaluar la fidelidad de la respuesta al contexto suministrado (evitando alucinaciones basadas en conocimiento externo no verificado), la relevancia del propio contexto recuperado en relación con la consulta original, y la capacidad del LLM para utilizar dicho contexto de manera efectiva.
- `"duration_seconds"`: Medición del tiempo, en segundos, que el LLM tardó en procesar el prompt y generar la `"output_response"`. Este dato es relevante para evaluar la eficiencia y latencia de cada modelo.
- `"retrieval_duration_seconds"`: Medición del tiempo, en segundos, que requirió la fase de recuperación de contexto (búsqueda en FAISS, BM25, fusión, y reranking) antes de la invocación del LLM.
- `"error"`: Campo destinado a registrar cualquier mensaje de error o excepción que pudiera haber ocurrido durante el proceso de recuperación o generación para una inferencia particular. Esto es vital para la trazabilidad de problemas y para excluir inferencias fallidas del análisis de rendimiento si fuera necesario.

La recolección sistemática de estos atributos proporciona una base de datos rica y estructurada, indispensable para llevar a cabo el análisis comparativo de los modelos LLM en el contexto RAG, que se abordará en secciones posteriores.

7.2.4 › Persistencia de Resultados para Análisis Futuro

Para asegurar la disponibilidad de los datos experimentales para análisis posteriores y futuras revisiones, los resultados consolidados de cada asignatura, almacenados en el DataFrame `results_df` (descrito en la Sección 7.2.3), fueron persistidos. Se optó por guardar los datos en archivos CSV locales, utilizando una convención de nombres por materia como `rag_evaluation_results_[MATERIA].csv`. Esta elección garantiza la accesibilidad y facilidad de procesamiento de los datos brutos.

Habiendo detallado el proceso de configuración, la definición de las tareas de evaluación, el flujo de ejecución de las inferencias y la metodología para la recopilación y persistencia de los datos, resulta ilustrativo observar algunos ejemplos concretos

de las interacciones generadas. La Tabla 7.2 presenta una muestra selecta de resultados obtenidos para cada una de las cuatro asignaturas del corpus, abarcando diferentes tipos de tareas RAG y uno de los modelos LLM evaluados. Estos ejemplos no pretenden ser exhaustivos, sino ofrecer una visión cualitativa del tipo de entradas procesadas, las respuestas generadas por los LLM y el contexto recuperado por el sistema RAG que fundamentó dichas respuestas.

La observación de estos ejemplos preliminares subraya la complejidad de la evaluación. Se evidencia cómo la calidad del contexto recuperado impacta directamente en la respuesta del LLM y cómo diferentes modelos pueden interpretar y utilizar dicho contexto de maneras variadas. Estos ejemplos también anticipan la necesidad de un análisis más profundo y sistemático, tanto cuantitativo como cualitativo, que se abordará en los capítulos siguientes. La riqueza de los datos recopilados, cuya estructura y persistencia se han descrito, permitirá realizar esta comparativa detallada con el fin de extraer conclusiones significativas sobre el rendimiento y la idoneidad de cada configuración LLM+RAG para el asistente académico propuesto.

Cuadro 7.2: Muestra de inferencias RAG y métricas de rendimiento por asignatura.

Asig.	**Tarea**	**Entrada**	**LLM**	**Respuesta (Idea Central)**	**Contexto (Fragmento Esencial)**	**T.LLM (s)**	**T.Retr. (s)**
Lengua y Lit.	RAG Q&A	Dif. var. diatópica y diastrática.	`gpt4o_openai`	Diatópica: geografía. Diastrática: estratos sociales.	ID:65: Espaciales (diatópico): geográficas. Sociales (diastrático): estratos sociales.	1.66	13.23
Biología	RAG Q&A	Insolubilidad comp. org. no polares.	`gpt4o_openai`	Interrumpen estructura agua, no forman enlaces H.	ID:31: Comp. org. no polares insolubles: interrumpen estructura, no forman enlaces H.	1.12	12.79
Física	RAG Summ.	Leyes de Newton: enunciado, significado.	`llama3_70b_fireworks`	Leyes Newton: fundamentales. 1ª: inercia. 2ª: F=ma (relación). 3ª: acción-reacción.	ID:82: Resolver 2ª ley Newton complejo. Conceptos: trabajo, energía. ID:62: Sist. no inerciales: pseudofuerzas.	0.93	14.33
Hist. España	RAG Q. Gen.	Tendencias Cortes de Cádiz.	`gpt4o_openai`	Test: Lib. radicales? R: Monarquía const. Des.: Dif. ideológicas.	ID:186: Tendencias: Lib. mod. (Jovell.), Lib. rad., absolutistas. Todos contra invasores.	3.25	12.82

Nota: Las entradas, respuestas y contextos han sido muy abreviados. T.LLM (s) y T.Retr. (s) son tiempos en segundos.

7.3 › Análisis Cualitativo de los Resultados

Tras haber detallado la configuración de los modelos LLM (Sección 7.1), el diseño del conjunto de datos y las tareas de evaluación RAG (Sección 7.2), y la estructura de los datos recopilados (Sección 7.2.1), este apartado se enfoca en el análisis comparativo de los resultados obtenidos. El objetivo es evaluar sistemáticamente el rendimiento de los modelos **GPT-4o**, **Mixtral 8×22B Instruct**, **Llama 3.1 70B Instruct** y **Gemini 1.5 Pro** cuando se integran con la arquitectura RAG, tal como se anticipó al final de la introducción de este capítulo (Capítulo 7).

Este análisis se desglosará en dos vertientes complementarias, tal como se planificó en la metodología (Capítulo 4): un análisis cualitativo, centrado en la calidad intrínseca de las respuestas, y un análisis cuantitativo, enfocado en métricas de rendimiento y eficiencia.

El análisis cualitativo se fundamenta en la revisión manual y sistemática de las respuestas (`output_response`) generadas por cada LLM para el conjunto de tareas y asignaturas definidas. Se prestó especial atención a la capacidad de los modelos para:

- Generar resúmenes fieles y concisos (tarea de `RAG Summarization`).
- Proponer preguntas relevantes y bien formuladas (tarea de `RAG Question Generation`).
- Elaborar respuestas fundamentadas y precisas a preguntas directas (tarea de `RAG Q&A`).

Para realizar esta evaluación, se añadieron columnas específicas al DataFrame de resultados consolidados (Sección 7.2.4). Estas columnas, pobladas mediante juicio experto, incluyen:

- **Puntuacion_Exactitud_Factual:** Evaluación de la veracidad de la información, utilizando una escala (e.g., 1-5 o descriptiva), crucial para evitar la propagación de errores, especialmente en el contexto académico y de oposiciones.
- **Puntuacion_Calidad_Sintesis :** Valoración de la capacidad de condensar información manteniendo las ideas clave, la coherencia y la claridad, en una escala definida.
- **Puntuacion_Coherencia_Narrativa:** Calificación de la estructura lógica, fluidez y cohesión del texto generado en todas las tareas, en una escala definida.
- **Puntuacion_Correccion_Gramatical_Estilo:** Medición de la corrección lingüística y la adecuación del estilo. Se puede complementar con el conteo de errores de herramientas como LanguageTool, tal como se previó.
- **Notas_Alucinaciones:** Registro de la presencia (Sí/No) y naturaleza de cualquier información inventada o no sustentada por el `retrieved_context`.

- **Notas_Uso_Contexto_RAG:** Observaciones sobre la efectividad con la que el LLM utilizó la información recuperada por el sistema RAG. Se documentó si el contexto fue bien integrado, parcialmente ignorado, o si el modelo tendió a alucinar a pesar del contexto provisto.

Este análisis cualitativo es fundamental para comprender las nuances del comportamiento de cada modelo más allá de las métricas puramente numéricas.

7.3.1 ❯ Desglose por Asignatura

Biologia

Aquí observamos el análisis cualitativo de los modelos GPT-4o, Gemini 1.5 Pro, Mixtral 8x22B Instruct y Llama 3.1 70B Instruct en 18 tareas de Biología (resumen, generación de preguntas y Q&A, todas con RAG). El análisis se centró en exactitud, calidad de síntesis, coherencia, corrección gramatical, incidencia de alucinaciones y uso del contexto RAG.

Puntuaciones Promedio de Calidad

Las respuestas de cada LLM fueron puntuadas manualmente en una escala de 1 a 5 (donde 5 representa la máxima calidad) para las dimensiones cualitativas mencionadas. La Tabla 7.3 resume las puntuaciones promedio obtenidas por cada modelo en la asignatura de Biología, incluyendo la desviación estándar (std) como medida de la consistencia en su rendimiento.

Proveedor LLM	**Exactitud Factual** *(media ± std)*	**Síntesis Calidad** *(media ± std)*	**Coherencia Narrativa** *(media ± std)*	**Gramática y Estilo** *(media ± std)*
GPT-4o	4.78 ± 0.94	4.44 ± 1.04	5.00 ± 0.00	5.00 ± 0.00
Gemini 1.5 Pro	4.78 ± 0.65	4.44 ± 0.86	4.89 ± 0.32	5.00 ± 0.00
Mixtral 8x22B Instr.	4.56 ± 1.04	4.44 ± 1.04	4.67 ± 1.03	5.00 ± 0.00
Llama 3.1 70B Instr.	4.33 ± 0.91	3.89 ± 1.13	4.28 ± 0.96	4.89 ± 0.47

Cuadro 7.3: Puntuaciones Cualitativas Promedio por LLM (Biología)

De la Tabla 7.3 se desprenden las siguientes observaciones:

- **Exactitud Factual:** GPT-4o y Gemini 1.5 Pro lideran con una puntuación promedio idéntica de 4.78. Gemini 1.5 Pro muestra una consistencia ligeramente superior (std: 0.65 frente a 0.94 de GPT-4o). Mixtral 8x22B se sitúa con un notable 4.56, aunque con una variabilidad mayor. Llama 3.1 70B obtiene la puntuación más baja (4.33) en esta métrica fundamental, indicando una menor fiabilidad en la presentación de información correcta.
- **Calidad de Síntesis:** GPT-4o, Gemini 1.5 Pro y Mixtral 8x22B Instruct alcanzan la misma puntuación promedio de 4.44, demostrando una alta capacidad para generar respuestas relevantes y bien estructuradas. Gemini 1.5 Pro

destaca por su mayor consistencia (std: 0.86). Llama 3.1 70B, con una media de 3.89 y la desviación estándar más alta (1.13), muestra un rendimiento inferior y más irregular en esta dimensión.

- **Coherencia Narrativa:** GPT-4o obtiene una puntuación perfecta de 5.00 con una desviación estándar de 0.00, lo que indica una excelencia y consistencia absolutas en la organización lógica y fluidez de sus respuestas. Gemini 1.5 Pro le sigue muy de cerca con 4.89 y una baja variabilidad. Mixtral 8x22B (4.67) también presenta un buen desempeño. Llama 3.1 70B (4.28) vuelve a ser el modelo con la puntuación más modesta.
- **Corrección Gramatical y Estilo:** Tres de los cuatro modelos (GPT-4o, Gemini 1.5 Pro y Mixtral 8x22B Instruct) logran la puntuación máxima de 5.00 sin desviación alguna, lo que refleja un dominio impecable del lenguaje en términos de gramática, ortografía y adecuación al estilo académico. Llama 3.1 70B, con una puntuación de 4.89 (std: 0.47), es también muy competente, aunque con una mínima incidencia de imperfecciones.

Análisis de Alucinaciones

La generación de información no verificada o incorrecta (alucinaciones) es un aspecto crítico en la evaluación de LLMs, especialmente para un asistente académico. La Tabla 7.4 muestra la incidencia de alucinaciones detectadas en las 18 tareas de Biología.

Proveedor LLM	Respuestas SIN Alucinación	Respuestas CON Alucinación
GPT-4o	17	1
Gemini 1.5 Pro	16	2
Mixtral 8x22B Instruct	17	1
Llama 3.1 70B Instruct	17	1

Cuadro 7.4: Incidencia de Alucinaciones por LLM (Biología)

En general, la tasa de alucinaciones fue baja para todos los modelos. Gemini 1.5 Pro presentó dos instancias, mientras que los demás modelos tuvieron una única alucinación cada uno. Es crucial analizar la naturaleza de estas alucinaciones:

- **Gemini 1.5 Pro (2 instancias):**
  1. Al resumir sobre las propiedades del agua, afirmó incorrectamente que el contexto RAG no proporcionaba información sobre cohesividad (mencionada en ID:26 como tensión superficial) ni calor específico (explícitamente en ID:26).
  2. Al resumir sobre el modelo de mosaico fluido, indicó erróneamente que el contexto RAG no mencionaba tipos de proteínas de membrana, cuando ID:85 sí especificaba "proteínas de canal y transportadoras específicas".

  Ambos casos sugieren una deficiencia en el procesamiento exhaustivo del contexto recuperado, omitiendo o no logrando integrar correctamente toda la información relevante disponible.

- **GPT-4o (1 instancia):** Al ser preguntado por la diferencia entre heterofagia y autofagia, respondió "No puedo responder con la información dada", a pesar de que el fragmento ID:94 del contexto RAG definía claramente ambos términos. Esto representa un fallo de omisión o una incapacidad para localizar y utilizar información explícita.
- **Llama 3.1 70B Instruct (1 instancia):** Al resumir las etapas de preparación de muestras para microscopía óptica, incluyó incorrectamente etapas pertenecientes a la microscopía electrónica (fijación no coagulante, deshidratación, inclusión en resina) extraídas del fragmento ID:8 del RAG. Esto indica una confusión o una mala integración de fragmentos de contexto dispares.
- **Mixtral 8x22B Instruct (1 instancia):** En una tarea de generación de preguntas sobre la evolución química prebiótica, la pregunta de desarrollo solicitaba explicar la "formación de coacervatos". Si bien este término estaba en el `input_id` original, el contexto RAG principal (ID:14) utilizaba el término "protobiontesz no definía çoacervatos". Esto sugiere una posible mayor adherencia al `input_id` que al detalle específico del contexto RAG proporcionado en ese momento, o una inferencia que no se ajustó completamente a la información recuperada.

Análisis del Uso del Contexto RAG

La habilidad de los LLM para utilizar eficazmente el contexto proporcionado por el sistema RAG es fundamental para la calidad y fiabilidad de sus respuestas. Las notas cualitativas recogidas durante la evaluación manual (ejemplificadas en la salida del script de análisis) revelan las siguientes tendencias para la asignatura de Biología:

- **GPT-4o:** Demostró, en la mayoría de los casos, un uso “Muy bueno” o “Excelente” del contexto. Fue capaz de integrar información de múltiples fragmentos relevantes (IDs específicos del RAG) y de discernir e ignorar correctamente la información no pertinente. El fallo en el caso de heterofagia/autofagia fue una excepción notable a este buen desempeño general.
- **Gemini 1.5 Pro:** Presentó un uso del contexto mayoritariamente “Excelente”, con una buena capacidad de síntesis e integración de la información recuperada. No obstante, las dos alucinaciones identificadas se originaron precisamente por no haber procesado o integrado completamente toda la información relevante que el RAG le había proporcionado, lo que llevó a evaluaciones de “Aceptable” y “Deficiente” en esos casos específicos.
- **Mixtral 8x22B Instruct:** Mostró un uso consistentemente “Excelente” o “Muy bueno” del contexto RAG. Logró integrar la información de manera precisa y efectiva. La única alucinación detectada pareció más una leve desviación hacia el `input_id` original que un fallo flagrante en el uso del RAG. En alguna ocasión, pudo ser ligeramente menos detallado que otros modelos al extraer la idea general de un fragmento sin profundizar en todos sus matices.
- **Llama 3.1 70B Instruct:** Fue el modelo con el rendimiento más heterogéneo en cuanto al uso del contexto. Si bien hubo instancias “Muy buenas”,

también se observaron casos calificados como "Buenos" donde no se aprovechó toda la profundidad del material recuperado (e.g., omitiendo detalles sobre la estabilidad del ADN o la integración completa de propiedades del agua). La alucinación sobre microscopía y un uso "Pobre" del contexto en la tarea sobre proteínas de membrana (ignorando información clave de ID:85) evidencian sus mayores desafíos en esta área.

Física

Continuando con la evaluación cualitativa, se analizaron las respuestas de los cuatro modelos LLM (GPT-4o, Gemini 1.5 Pro, Mixtral 8x22B Instruct y Llama 3.1 70B Instruct) para la asignatura de Física.

Puntuaciones Promedio de Calidad

La Tabla 7.5 presenta las puntuaciones promedio (escala 1-5) y las desviaciones estándar (std) obtenidas por cada modelo en la asignatura de Física.

Proveedor LLM	**Exactitud Factual** *(media ± std)*	**Síntesis Calidad** *(media ± std)*	**Coherencia Narrativa** *(media ± std)*	**Gramática y Estilo** *(media ± std)*
GPT-4o	4.67 ± 0.77	4.17 ± 1.20	4.78 ± 0.73	5.00 ± 0.00
Gemini 1.5 Pro	4.67 ± 0.97	3.94 ± 1.16	4.89 ± 0.32	5.00 ± 0.00
Mixtral 8x22B Instr.	4.39 ± 0.92	3.94 ± 1.31	4.39 ± 1.14	5.00 ± 0.00
Llama 3.1 70B Instr.	4.06 ± 1.16	3.61 ± 1.20	4.33 ± 1.19	5.00 ± 0.00

Cuadro 7.5: Puntuaciones Cualitativas Promedio por LLM (Física)

Del análisis de la Tabla 7.5, se extraen las siguientes conclusiones:

- **Exactitud Factual:** GPT-4o y Gemini 1.5 Pro comparten la puntuación promedio más alta (4.67). GPT-4o muestra una mayor consistencia (std: 0.77 frente a 0.97 de Gemini). Mixtral 8x22B (4.39) y Llama 3.1 70B (4.06) obtienen puntuaciones inferiores, siendo Llama 3.1 70B el que presenta la mayor variabilidad (std: 1.16).
- **Calidad de Síntesis:** GPT-4o lidera esta métrica con 4.17, aunque con una variabilidad considerable (std: 1.20). Gemini 1.5 Pro y Mixtral 8x22B le siguen con 3.94, ambos con una alta desviación estándar, sugiriendo un rendimiento irregular en la claridad y estructura de sus respuestas en Física. Llama 3.1 70B obtiene la puntuación más baja (3.61). En general, las puntuaciones de síntesis en Física son más bajas y más variables que en Biología para todos los modelos.
- **Coherencia Narrativa:** Gemini 1.5 Pro destaca ligeramente con 4.89 y una notable consistencia (std: 0.32). GPT-4o le sigue de cerca con 4.78. Mixtral 8x22B (4.39) y Llama 3.1 70B (4.33) muestran puntuaciones más bajas y una mayor variabilidad, indicando desafíos en la organización lógica de las respuestas en este dominio.

- **Corrección Gramatical y Estilo:** Todos los modelos obtuvieron una puntuación perfecta de 5.00 con desviación estándar nula, demostrando un manejo impecable de la gramática y el estilo académico en el contexto de la Física.

Análisis de Alucinaciones

La Tabla 7.6 detalla la incidencia de alucinaciones en las tareas de Física.

Proveedor LLM	Respuestas SIN Alucinación	Respuestas CON Alucinación
GPT-4o	18	0
Gemini 1.5 Pro	17	1
Mixtral 8x22B Instruct	16	2
Llama 3.1 70B Instruct	16	2

Cuadro 7.6: Incidencia de Alucinaciones por LLM (Física)

GPT-4o fue el único modelo que no presentó ninguna alucinación en las tareas de Física. Los otros modelos tuvieron incidencias bajas:

- **Llama 3.1 70B Instruct (2 instancias):**
  1. Al resumir las Leyes de Newton, enunció las leyes correctamente basándose en su conocimiento general, pero dichos enunciados explícitos no se encontraban en el contexto RAG proporcionado (ID:82, ID:62, etc.). Esto es una alucinación por apoyarse en conocimiento externo en lugar de ceñirse al RAG.
  2. Al describir el Movimiento Armónico Simple (MAS), afirmó que la solución de la ecuación diferencial permitía analizar el comportamiento dinámico, lo cual es conceptualmente correcto para el MAS, pero el contexto RAG no presentaba dicha solución ni su análisis explícito.
- **Mixtral 8x22B Instruct (2 instancias):**
  1. En generación de preguntas sobre curvas de energía potencial, una pregunta test asumía una fórmula de energía potencial ($U = 0{,}5kx^2$) no explícita en el RAG (ID:105 solo tenía $F = -kx$ y un gráfico). Otra pregunta de desarrollo pedía distinguir equilibrio estable/inestable, pero el RAG (ID:96) solo indicaba que eran máximos/mínimos sin detallar la diferencia mediante la segunda derivada. Ambas son inferencias que van más allá de la información explícita del RAG.
  2. Al definir cosenos directores, proporcionó las fórmulas correctas ($\cos\alpha = x/r$, etc.), pero estas no estaban explícitamente en el contexto RAG (ID:25 solo los nombraba).
- **Gemini 1.5 Pro (1 instancia):** Al generar preguntas sobre sistemas de referencia, afirmó incorrectamente que el RAG no relacionaba la diferencia entre SRI/SRNI con la primera ley de Newton, cuando el fragmento ID:56 del RAG sí lo hacía explícitamente. Similar a Biología, un fallo en el procesamiento completo del RAG.

En Física, varias alucinaciones (especialmente en Llama y Mixtral) consistieron en complementar la respuesta con conocimiento general correcto pero no directamente extraíble o sustentado por el fragmento RAG específico proporcionado, o en hacer inferencias que iban un paso más allá de lo explícito en el texto.

Análisis del Uso del Contexto RAG

El uso del contexto RAG en Física mostró las siguientes tendencias:

- **GPT-4o:** Generalmente "Bueno" o "Excelente". Demostró una buena capacidad para identificar qué información estaba presente o ausente en el RAG (e.g., definición de escalares, ausencia de a(t)). En algunos casos, se ciñó estrictamente a no inferir si la información no era explícita. Integró bien múltiples fragmentos relevantes (e.g., para trabajo y energía).
- **Gemini 1.5 Pro:** Predominantemente "Excelente". Utilizó eficazmente los fragmentos RAG para definir conceptos y reconoció ausencias de información. La única alucinación se debió a no procesar correctamente un detalle del RAG. Mostró capacidad de inferencia adecuada (e.g., definición de escalares).
- **Mixtral 8x22B Instruct:** Generalmente "Muy bueno" o "Excelente". Fue capaz de complementar la información del RAG con definiciones estándar correctas cuando el RAG era escueto (e.g., escalares), cumpliendo mejor el 'input_id' en esos casos. En ocasiones, sus alucinaciones se debieron a asumir detalles no explícitos en el RAG para generar preguntas más completas.
- **Llama 3.1 70B Instruct:** El más irregular. Tuvo momentos "Excelentes" (e.g., uso de U(x)), pero también "Pobres" donde ignoró el RAG y recurrió a conocimiento interno (Leyes de Newton) o no extrajo información central de los fragmentos proporcionados (trabajo por fuerza constante).

Historia

La evaluación cualitativa continuó con el análisis de las respuestas generadas por los cuatro modelos LLM para la asignatura de Historia, comprendiendo un total de 18 tareas distribuidas entre resumen RAG, generación de preguntas RAG y Q&A RAG. Se aplicaron los mismos criterios de evaluación que en las asignaturas previas:

Puntuaciones Promedio de Calidad

La Tabla 7.7 detalla las puntuaciones promedio (escala 1-5) y las desviaciones estándar (std) obtenidas por cada modelo en la asignatura de Historia.

El análisis de la Tabla 7.7 arroja las siguientes observaciones:

- **Exactitud Factual:** GPT-4o muestra un rendimiento sobresaliente con una media de 4.89 y una desviación estándar muy baja (0.32), indicando una alta fiabilidad y consistencia. Le sigue Mixtral 8x22B con 4.67, y Gemini 1.5 Pro con 4.61. Llama 3.1 70B obtiene la puntuación más baja (4.33), aunque con una consistencia similar a la de Mixtral.
- **Calidad de Síntesis:** GPT-4o también lidera en esta métrica con 4.61. Mixtral 8x22B (4.33) y Gemini 1.5 Pro (4.22) presentan un buen desempeño. Llama 3.1 70B (3.83) se sitúa por debajo, mostrando una mayor variabilidad.

Proveedor LLM	Exactitud Factual *(media ± std)*	Síntesis Calidad *(media ± std)*	Coherencia Narrativa *(media ± std)*	Gramática y Estilo *(media ± std)*
GPT-4o	4.89 ± 0.32	4.61 ± 0.85	5.00 ± 0.00	5.00 ± 0.00
Gemini 1.5 Pro	4.61 ± 0.61	4.22 ± 1.06	4.72 ± 0.57	4.89 ± 0.47
Mixtral 8x22B Instr.	4.67 ± 0.77	4.33 ± 0.97	4.83 ± 0.51	5.00 ± 0.00
Llama 3.1 70B Instr.	4.33 ± 0.77	3.83 ± 1.04	4.56 ± 0.70	4.89 ± 0.47

Cuadro 7.7: Puntuaciones Cualitativas Promedio por LLM (Historia)

- **Coherencia Narrativa:** GPT-4o nuevamente alcanza la puntuación perfecta de 5.00 (std: 0.00). Mixtral 8x22B (4.83) y Gemini 1.5 Pro (4.72) demuestran una excelente coherencia y buena consistencia. Llama 3.1 70B (4.56) obtiene la puntuación más baja, aunque todavía en un nivel aceptable.
- **Corrección Gramatical y Estilo:** GPT-4o y Mixtral 8x22B Instruct logran una puntuación perfecta de 5.00. Gemini 1.5 Pro y Llama 3.1 70B, con 4.89, son también altamente competentes, con mínimas e infrecuentes imperfecciones.

Análisis de Alucinaciones

La Tabla 7.8 resume la incidencia de alucinaciones en las tareas de Historia.

Proveedor LLM	Respuestas SIN Alucinación	Respuestas CON Alucinación
GPT-4o	18	0
Gemini 1.5 Pro	17	1
Mixtral 8x22B Instruct	16	2
Llama 3.1 70B Instruct	15	3

Cuadro 7.8: Incidencia de Alucinaciones por LLM (Historia)

Una vez más, GPT-4o no presentó ninguna alucinación. Llama 3.1 70B fue el modelo con mayor incidencia (3 casos), seguido por Mixtral 8x22B (2 casos) y Gemini 1.5 Pro (1 caso). La naturaleza de estas alucinaciones es la siguiente:

- **Llama 3.1 70B Instruct (3 instancias):**
  1. En la revuelta de los Comuneros, describió consecuencias no presentes en el RAG, recurriendo a conocimiento externo.
  2. Sobre los tipos de repoblación, una pregunta de desarrollo solicitaba información sobre "Repoblación por Donadíos Reales", término no explícito en el fragmento RAG (ID:52).
  3. Al tratar la Desamortización de Mendizábal, mezcló sus objetivos con los de la Desamortización de Madoz (presentes en diferentes IDs del RAG) y añadió información no explícita ("bienes señoriales").

  Estos errores evidencian una tendencia a complementar o contradecir el RAG con conocimiento externo o a confundir información de diferentes contextos.

- **Mixtral 8x22B Instruct (2 instancias):**
  1. Al igual que Llama, en el tema de la repoblación, generó preguntas sobre "Repoblación por Donadíos Reales", no sustentado por el fragmento ID:52.
  2. Atribuyó incorrectamente objetivos de la Desamortización de Madoz (RAG ID:274) a la de Mendizábal (RAG ID:238), confundiendo la información específica de cada fragmento.
- **Gemini 1.5 Pro (1 instancia):** En una pregunta sobre tipos de repoblación, usó el término "Repoblación por Donadíos Reales", no explícito en el fragmento RAG ID:52.

Análisis del Uso del Contexto RAG

La habilidad para utilizar el contexto RAG en Historia se manifestó de la siguiente manera:

- **GPT-4o:** Mostró un uso consistentemente "Excelente" o "Muy bueno" del contexto. Se basó de forma precisa y completa en los fragmentos RAG pertinentes, integrando información de múltiples IDs e ignorando la irrelevante. Manejó adecuadamente los RAG incompletos al no inventar información faltante.
- **Gemini 1.5 Pro:** Predominantemente "Excelente". Utilizó la información RAG de forma precisa y fue capaz de discernir cuándo un RAG era insuficiente para cubrir todo el `input_id`. En alguna ocasión, como en el tema de la romanización, no integró todos los fragmentos disponibles de manera exhaustiva, resultando en una respuesta calificada como "Buena" pero incompleta.
- **Mixtral 8x22B Instruct:** Generalmente "Excelente" o "Muy bueno". Se adhirió bien al contexto RAG. Similar a Gemini, en algunas instancias no llegó a integrar la totalidad de la información de todos los fragmentos RAG recuperados, lo que llevó a respuestas buenas pero no exhaustivas para el `input_id` completo.
- **Llama 3.1 70B Instruct:** Exhibió un rendimiento variable. Hubo respuestas "Excelentes" y "Buenas", pero también "Aceptables" y "Pobres". Esto se notó especialmente cuando el RAG era limitado y el modelo tendía a recurrir a conocimiento externo (e.g., Comuneros) o cuando confundía información (Desamortizaciones).

Lengua y Literatura

Finalmente, se evaluó cualitativamente el desempeño de los cuatro modelos LLM en la asignatura de Lengua y Literatura.

Puntuaciones Promedio de Calidad

La Tabla 7.9 presenta las puntuaciones promedio (escala 1-5) y las desviaciones estándar (std) para cada modelo en la asignatura de Lengua y Literatura.

El análisis de las puntuaciones en Lengua y Literatura (Tabla 7.9) indica:

Proveedor LLM	Exactitud Factual *(media ± std)*	Síntesis Calidad *(media ± std)*	Coherencia Narrativa *(media ± std)*	Gramática y Estilo *(media ± std)*
GPT-4o	4.94 ± 0.24	4.83 ± 0.51	4.94 ± 0.24	5.00 ± 0.00
Gemini 1.5 Pro	4.89 ± 0.32	4.67 ± 0.49	4.94 ± 0.24	5.00 ± 0.00
Mixtral 8x22B Instr.	4.44 ± 1.04	4.28 ± 1.07	4.67 ± 0.84	5.00 ± 0.00
Llama 3.1 70B Instr.	4.72 ± 0.57	4.33 ± 0.84	4.78 ± 0.55	5.00 ± 0.00

Cuadro 7.9: Puntuaciones Cualitativas Promedio por LLM (Lengua y Literatura)

- **Exactitud Factual:** GPT-4o obtiene la puntuación más alta (4.94) con una excelente consistencia (std: 0.24). Le sigue de cerca Gemini 1.5 Pro (4.89). Llama 3.1 70B (4.72) también presenta un buen rendimiento, mientras que Mixtral 8x22B (4.44) obtiene la puntuación más baja y la mayor variabilidad.
- **Calidad de Síntesis:** GPT-4o lidera nuevamente con 4.83 y buena consistencia. Gemini 1.5 Pro (4.67) le sigue, destacando por su alta consistencia (std: 0.49). Llama 3.1 70B (4.33) y Mixtral 8x22B (4.28) muestran un rendimiento bueno pero inferior a los dos primeros.
- **Coherencia Narrativa:** GPT-4o y Gemini 1.5 Pro empatan con una excelente puntuación de 4.94 y muy alta consistencia. Llama 3.1 70B (4.78) y Mixtral 8x22B (4.67) también demuestran una muy buena coherencia en sus respuestas.
- **Corrección Gramatical y Estilo:** Todos los modelos alcanzaron la puntuación perfecta de 5.00 sin desviación estándar, indicando un dominio impecable de la lengua en este contexto.

En general, las puntuaciones en Lengua y Literatura son muy altas para la mayoría de los modelos, especialmente para GPT-4o y Gemini 1.5 Pro.

Análisis de Alucinaciones

La Tabla 7.10 muestra la incidencia de alucinaciones en las tareas de Lengua y Literatura.

Proveedor LLM	Respuestas SIN Alucinación	Respuestas CON Alucinación
GPT-4o	18	0
Gemini 1.5 Pro	18	0
Llama 3.1 70B Instruct	18	0
Mixtral 8x22B Instruct	16	2

Cuadro 7.10: Incidencia de Alucinaciones por LLM (Lengua y Literatura, n=18 tareas)

Destaca notablemente que GPT-4o, Gemini 1.5 Pro y Llama 3.1 70B Instruct no presentaron ninguna alucinación en las 18 tareas de Lengua y Literatura. Mixtral 8x22B Instruct fue el único con incidencias:

- **Mixtral 8x22B Instruct (2 instancias):**
  1. En una pregunta de desarrollo sobre fenómenos métricos, el ejemplo de hiato proporcionado ("ha quedado") era incorrecto; el contexto RAG (ID:136) ofrecía un ejemplo diferente ("mi hado"). Esto sugiere una invención o un recuerdo incorrecto de un ejemplo.
  2. Al ser preguntado sobre la tilde diacrítica en "tú", afirmó incorrectamente que "tú" como pronombre personal sujeto NO lleva tilde y que el contexto no especificaba cuándo sí la llevaba. Sin embargo, el RAG (ID:61) explicaba claramente la regla (pronombre "Tú" con tilde, posesivo "Tu" sin tilde). Esto indica un fallo en la comprensión o aplicación de la información del RAG.

La ausencia total de alucinaciones en tres de los cuatro modelos en esta asignatura es un resultado muy positivo.

Análisis del Uso del Contexto RAG

La capacidad de los modelos para utilizar el contexto RAG en Lengua y Literatura fue generalmente muy alta:

- **GPT-4o:** Consistentemente "Excelente". Demostró una capacidad superior para integrar información de múltiples fragmentos RAG de manera precisa y completa (e.g., propiedades textuales, denotación/connotación, deixis, reglas de acentuación, figuras retóricas), ignorando fragmentos irrelevantes cuando correspondía.
- **Gemini 1.5 Pro:** También "Excelente" en la mayoría de los casos. Utilizó eficazmente la información de los diferentes IDs para construir respuestas completas y precisas, reconociendo la presencia de ejemplos y detalles relevantes en el RAG.
- **Llama 3.1 70B Instruct:** Predominantemente "Excelente". Al igual que GPT-4o y Gemini, mostró una fuerte capacidad para integrar información de diversos fragmentos RAG de forma precisa, como en los temas de propiedades textuales, denotación/connotación, deixis y figuras retóricas. Hubo alguna ligera imprecisión o generalización en temas como las reglas de acentuación.
- **Mixtral 8x22B Instruct:** En general "Excelente" o "Muy bueno". Se basó sólidamente en el RAG para la mayoría de las tareas. Las dos alucinaciones detectadas se debieron a un uso incorrecto o a una mala interpretación del contexto RAG en esos casos específicos. En alguna ocasión, no incorporó toda la información disponible de todos los fragmentos o introdujo elementos no centrales (e.g., diptongos/hiatos en reglas de acentuación general).

7.3.2 › Análisis Comparativo Cualitativo Global

Tras la evaluación detallada del rendimiento cualitativo de los modelos **GPT-4o**, **Gemini 1.5 Pro**, **Mixtral 8x22B Instruct** y **Llama 3.1 70B Instruct** en las asignaturas de Biología, Física, Historia, y Lengua y Literatura, esta sección consolida dichos hallazgos. El objetivo es realizar un análisis comparativo global que permita identificar las fortalezas y debilidades inherentes a cada LLM a través de un total de 72 tareas evaluadas por modelo.

Rendimiento Cualitativo Agregado y por Asignatura

Para obtener una visión comprensiva de la calidad general de las respuestas, se calcularon las puntuaciones promedio globales para cada una de las cuatro dimensiones cualitativas evaluadas (Exactitud Factual, Calidad de Síntesis, Coherencia Narrativa, y Corrección Gramatical y Estilo). Adicionalmente, se creó una métrica de **Puntuación Cualitativa General**, promediando estas cuatro dimensiones para cada respuesta, con el fin de visualizar el desempeño agregado.

La Tabla 7.11 resume las puntuaciones promedio globales obtenidas por cada modelo. La Figura 7.1 ilustra cómo esta Puntuación Cualitativa General varía para cada LLM a través de las diferentes asignaturas, permitiendo observar su consistencia y adaptabilidad a distintos dominios temáticos.

Proveedor LLM	**Exactitud Factual** *(media ± std)*	**Síntesis Calidad** *(media ± std)*	**Coherencia Narrativa** *(media ± std)*	**Gramática y Estilo** *(media ± std)*
GPT-4o	4.82 ± 0.64	4.51 ± 0.95	4.93 ± 0.39	5.00 ± 0.00
Gemini 1.5 Pro	4.74 ± 0.67	4.32 ± 0.95	4.86 ± 0.39	4.97 ± 0.24
Mixtral 8x22B Instr.	4.51 ± 0.93	4.25 ± 1.10	4.64 ± 0.91	5.00 ± 0.00
Llama 3.1 70B Instr.	4.36 ± 0.89	3.92 ± 1.07	4.49 ± 0.89	4.94 ± 0.33

Cuadro 7.11: Puntuaciones Cualitativas Promedio Globales por LLM (Todas las asignaturas)

Del análisis global de las puntuaciones (Tabla 7.11):

- **GPT-4o** se posiciona como el modelo con el mejor rendimiento cualitativo general, liderando en Exactitud Factual (4.82), Calidad de Síntesis (4.51) y Coherencia Narrativa (4.93), y alcanzando la perfección en Corrección Gramatical y Estilo (5.00). Su desempeño es consistentemente alto, como lo indican sus desviaciones estándar relativamente bajas en coherencia y nula en gramática.
- **Gemini 1.5 Pro** le sigue de cerca, demostrando ser un modelo muy competente y fiable, con una excelente Coherencia Narrativa (4.86) y Exactitud Factual (4.74). Su Calidad de Síntesis (4.32) y Corrección Gramatical (4.97) también son muy elevadas.
- **Mixtral 8x22B Instruct** ofrece un rendimiento sólido, destacando con una puntuación perfecta en Corrección Gramatical y Estilo. Su Exactitud Factual

(4.51) y Calidad de Síntesis (4.25) son buenas, aunque con una variabilidad ligeramente mayor que los dos modelos anteriores en algunas métricas.

- **Llama 3.1 70B Instruct** presenta los promedios globales más bajos en las métricas de contenido (Exactitud: 4.36, Síntesis: 3.92, Coherencia: 4.49), aunque su Corrección Gramatical y Estilo (4.94) es muy alta. Este modelo también muestra, en general, las mayores desviaciones estándar, sugiriendo una mayor variabilidad en la calidad de sus respuestas.

La Figura 7.1 ofrece una perspectiva visual de cómo se distribuye la Puntuación Cualitativa General de cada LLM entre las cuatro asignaturas evaluadas.

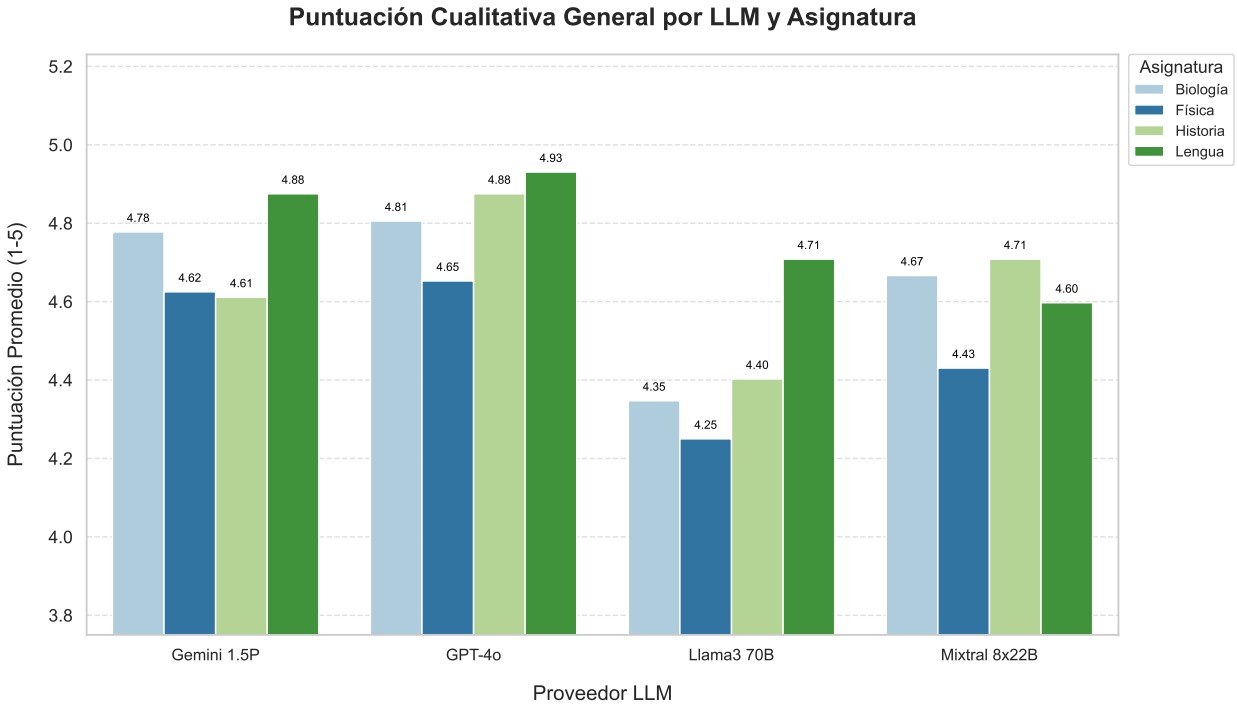

Figura 7.1: Puntuación Cualitativa General Promedio por LLM y Asignatura.

Análisis Global de Alucinaciones y Uso del Contexto RAG

La fiabilidad de un asistente académico depende crucialmente de su capacidad para evitar la generación de información incorrecta (*alucinaciones*) y para utilizar adecuadamente el contexto proporcionado.

Incidencia Global de Alucinaciones

La Tabla 7.12 resume la incidencia total de alucinaciones por modelo a lo largo de las tareas. La Figura 7.2 visualiza la distribución de estas alucinaciones por asignatura para cada LLM.

- **GPT-4o** demostró ser el modelo más fiable, con solo una alucinación (1.39 % del total de sus respuestas).
- **Gemini 1.5 Pro** tuvo una baja incidencia, con 4 alucinaciones (5.56 %).

Proveedor LLM	Respuestas SIN Alucinación	Respuestas CON Alucinación	% Con Alucinación
GPT-4o	71	1	1.39 %
Gemini 1.5 Pro	68	4	5.56 %
Mixtral 8x22B Instr.	65	7	9.72 %
Llama 3.1 70B Instr.	66	6	8.33 %

Cuadro 7.12: Incidencia Global de Alucinaciones por LLM (n=72 evaluac. por LLM)

- **Llama 3.1 70B Instruct** presentó 6 alucinaciones (8.33 %).
- **Mixtral 8x22B Instruct** fue el que tuvo la mayor tasa de alucinaciones, con 7 casos (9.72 %).

La Figura 7.2 ofrece una perspectiva más granular.

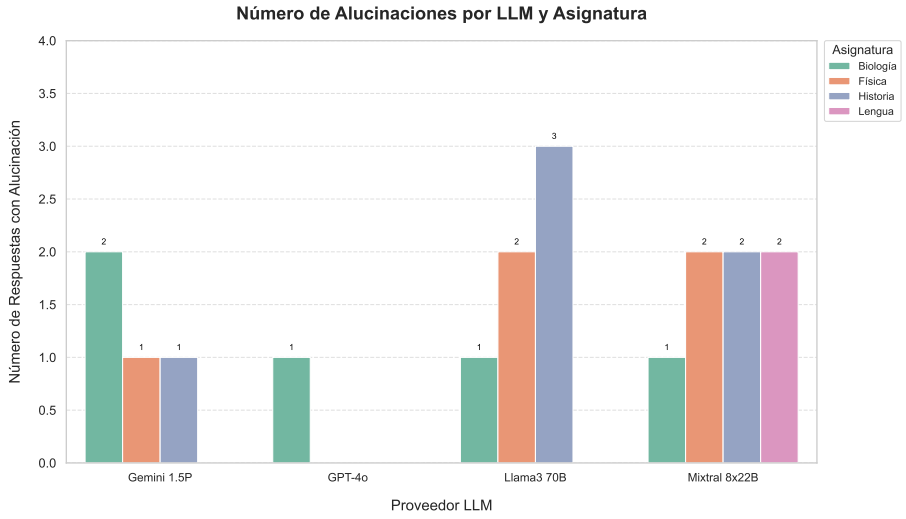

Figura 7.2: Número de Alucinaciones por LLM y Asignatura.

El análisis cualitativo de las descripciones de las alucinaciones (detallado en las secciones por asignatura) indica que estas no solo consisten en invenciones, sino también en omisiones de información clave del RAG, inferencias que exceden lo sustentado por el contexto, o la incorrecta atribución/mezcla de información cuando el RAG es complejo o ambiguo. Los modelos **Llama 3.1 70B** y **Mixtral 8x22B** mostraron una mayor tendencia a complementar con conocimiento externo, mientras que los errores de **GPT-4o** y **Gemini 1.5 Pro** se relacionaron más con el procesamiento del propio contexto RAG.

Uso Global del Contexto RAG

Sintetizando las observaciones de las cuatro asignaturas sobre el uso del contexto RAG:

- **GPT-4o:** Consistentemente el más adepto en el uso del RAG, integrando múltiples fragmentos y discerniendo la relevancia con alta precisión.

- **Gemini 1.5 Pro:** Muy competente en la utilización del RAG, aunque con episodios aislados de no extraer toda la información pertinente del contexto suministrado.
- **Mixtral 8x22B Instruct:** Buen uso general del RAG, si bien a veces tendió a hacer inferencias que iban ligeramente más allá de lo explícito en el texto recuperado.
- **Llama 3.1 70B Instruct:** El más variable, con instancias de excelente uso del RAG, pero también con una mayor propensión a desviarse o a no integrar completamente la información cuando el RAG era percibido como incompleto o era complejo.

7.4 › Análisis Cuantitativo: Costes y Eficiencia Temporal

Además de la evaluación cualitativa del contenido generado por los modelos LLM, resulta fundamental analizar aspectos cuantitativos que impactan directamente en la viabilidad y practicidad de su implementación en un asistente académico. En esta sección, se profundiza en dos dimensiones críticas: el coste económico asociado a la utilización de cada modelo a través de sus respectivas API y la eficiencia temporal, desglosada en los tiempos de recuperación del sistema RAG y los tiempos de procesamiento propios de cada LLM. Estos análisis proporcionan una perspectiva complementaria esencial para una toma de decisiones informada sobre la selección del modelo más adecuado según las restricciones presupuestarias y los requisitos de latencia del sistema.

7.4.1 › Análisis de Coste Económico Estimado

Tal como se estableció en el Objetivo 3 del TFM, la viabilidad práctica es un criterio clave para la selección del modelo. Por ello, se realizó una estimación del coste económico asociado a la utilización de cada uno de los modelos LLM accedidos vía API durante la fase experimental. Los proveedores y modelos evaluados en términos de coste fueron:

- **GPT-4o:** A través de la API de OpenAI.
- **Gemini 1.5 Pro:** A través de Google Cloud AI.
- **Mixtral 8x22B Instruct y Llama 3.1 70B Instruct:** A través de la API de Fireworks AI.

Este análisis se centró en el coste directo de operación por inferencia. Es importante reconocer que para los modelos de código abierto (Mixtral y Llama), el uso de una API de inferencia como la de Fireworks AI introduce un coste por conveniencia y gestión de la infraestructura. Un despliegue local autogestionado de estos modelos modificaría la estructura de costes, trasladándola hacia la inversión inicial y el mantenimiento de hardware, así como el consumo energético, un aspecto que se considera para futuras implementaciones o escenarios de uso a gran escala.

La estimación del coste para cada tarea ejecutada se basó en el conteo de tokens de entrada (prompt) y de salida (respuesta generada), aplicando las tarifas vigentes de cada proveedor por millón de tokens. El proceso de cálculo fue el siguiente:

1. **Conteo de Tokens:**
   - Para **GPT-4o**, se utilizó la librería `tiktoken` de OpenAI con la codificación `cl100k_base`.
   - Para **Gemini 1.5 Pro**, se empleó el método `count_tokens` del SDK oficial de Google (`google.generativeai`).
   - Para **Llama 3.1 70B Instruct** y **Mixtral 8x22B Instruct**, se utilizaron los tokenizadores específicos de la librería `transformers` de Hugging Face (`meta-llama/Llama-3.1-70B-Instruct` y `mistralai/Mixtral-8x22B-Instruct-v0.1` respectivamente), recurriendo a `tiktoken` como aproximación si los tokenizadores específicos fallaban.
2. **Aplicación de Tarifas:** Se consultaron las tarifas públicas de cada proveedor (OpenAI, Google Cloud, Fireworks AI) para los modelos específicos, expresadas en coste por millón de tokens de entrada y por millón de tokens de salida. Las tarifas utilizadas en el momento del estudio se resumen en la Tabla 7.13.

Proveedor LLM	Coste Input / 1M tokens	Coste Output / 1M tokens
GPT-4o (OpenAI)	$2.50	$10.00
Gemini 1.5 Pro (Google)	$1.25	$5.00
Llama 3.1 70B (Fireworks AI)	$0.90	$0.90
Mixtral 8x22B (Fireworks AI)	$1.20	$1.20

Cuadro 7.13: Tarifas de los Modelos LLM por Millón de Tokens (USD)
Nota: Precios vigentes durante la realización del estudio. Estos pueden variar.

Los resultados de la estimación de costes para el conjunto global de tareas experimentales (combinando todas las asignaturas y tareas) se presentan en la Tabla 7.14. Se muestra el coste promedio estimado por tarea para cada LLM y el coste total acumulado para todas las tareas ejecutadas durante la fase experimental.

Proveedor LLM	Coste Promedio por Tarea (USD)	Coste Total Acumulado (USD)
GPT-4o	0.005869	0,422 560
Gemini 1.5 Pro	0.002560	0,184 336
Mixtral 8x22B Instruct	0.002774	0,199 752
Llama 3.1 70B Instruct	0.001724	0,124 119

Cuadro 7.14: Costes Estimados Globales por LLM

Para visualizar mejor estas diferencias de coste, la Figura 7.3 muestra el coste total acumulado durante la experimentación.

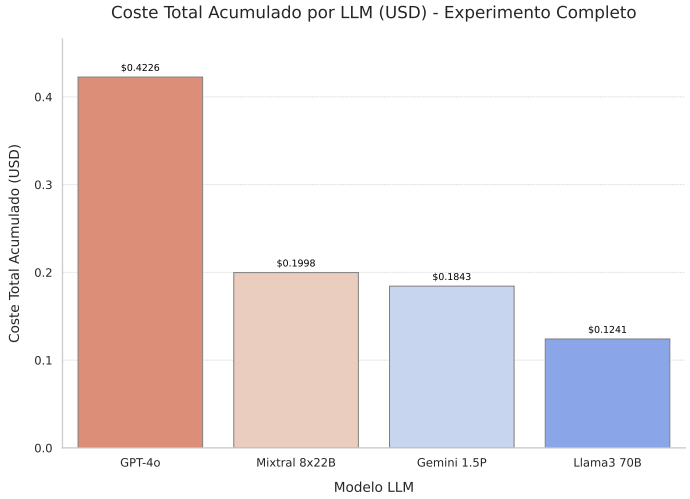

Figura 7.3: Coste Promedio Estimado por Tarea para cada LLM (USD).

De la Tabla 7.14, se observa que **Llama 3.1 70B Instruct** (vía Fireworks AI) resultó ser el modelo más económico por tarea, con un coste promedio de $0.001724. Le siguen **Gemini 1.5 Pro** ($0.002560) y **Mixtral 8x22B Instruct** ($0.002774), que presentan costes promedio por tarea muy similares entre sí. **GPT-4o** es significativamente el modelo más costoso, con un coste promedio por tarea de $0.005869, más del doble que Gemini o Mixtral, y más del triple que Llama 3.1. Esta diferencia se refleja también en el coste total acumulado para el experimento.

Adicionalmente, se analizó el coste promedio estimado desglosado por tipo de tarea RAG, como se muestra en la Tabla 7.15.

Cuadro 7.15: Coste Promedio Estimado por LLM y Tipo de Tarea (USD)

Proveedor LLM	**Tipo de Tarea RAG**	**Coste Promedio / Tarea**
Gemini 1.5 Pro	RAG Q&A	0,002 486
	RAG Question Generation	0,002 779
	RAG Summarization	0,002 416
GPT-4o	RAG Q&A	0,005 862
	RAG Question Generation	0,005 837
	RAG Summarization	0,005 907
Llama 3.1 70B Instruct	RAG Q&A	0,001 856
	RAG Question Generation	0,001 651
	RAG Summarization	0,001 664
Mixtral 8x22B Instruct	RAG Q&A	0,002 909
	RAG Question Generation	0,002 793
	RAG Summarization	0,002 621

La Figura 7.4 ilustra estas ligeras variaciones en el coste promedio por tipo de tarea para cada uno de los modelos LLM evaluados.

El desglose por tipo de tarea (Tabla 7.15) no revela variaciones drásticas en el coste promedio por tarea para un mismo modelo entre los diferentes tipos de actividad RAG. Las diferencias observadas son pequeñas y sugieren que la longitud del prompt de entrada (incluyendo el contexto RAG) y la longitud de la respues-

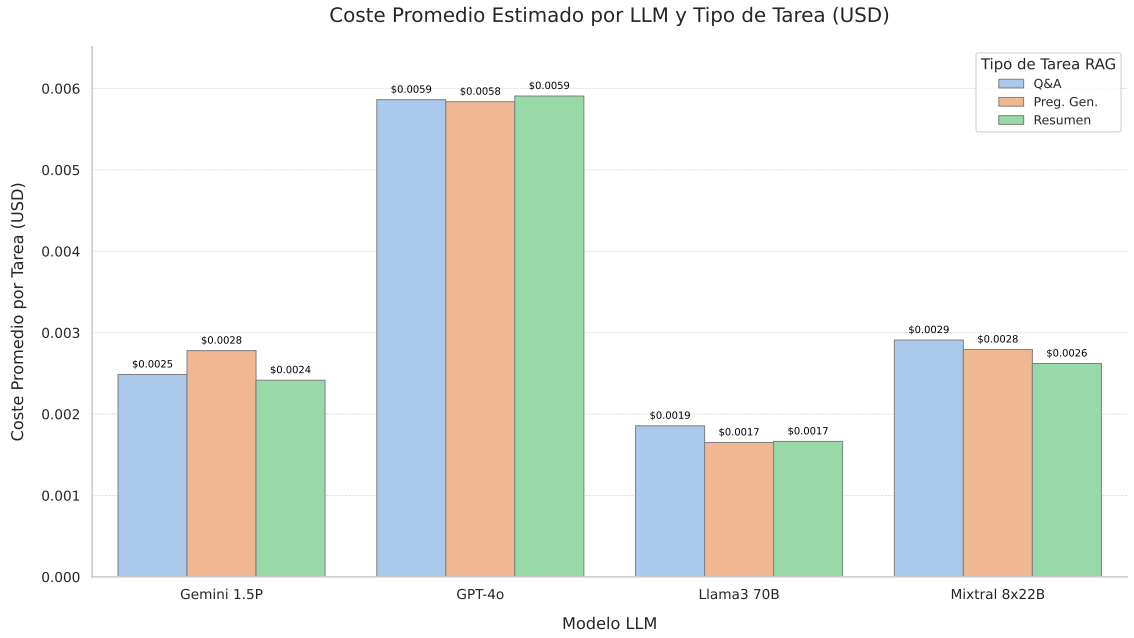

Figura 7.4: Coste Promedio Estimado por LLM y Tipo de Tarea RAG (USD).

ta generada son factores más determinantes del coste que la naturaleza intrínseca de la tarea (resumir, generar preguntas o responder). No obstante, se mantiene la tendencia general de costes relativos entre los modelos.

7.4.2 ❯ Análisis de Tiempos de Inferencia y Eficiencia del Sistema

Además del coste económico, la eficiencia temporal es un factor crucial para la viabilidad de un asistente académico interactivo. Se analizaron los tiempos de procesamiento de los LLM y el tiempo de recuperación del sistema RAG, utilizando los datos recopilados de todas las tareas en las cuatro asignaturas (Biología, Física, Historia y Lengua).

Tiempo de Recuperación del Sistema RAG (`retrieval_time`).

El componente RAG, encargado de la búsqueda semántica, léxica y el reranking, introduce una latencia previa a la invocación del LLM. El análisis global de este tiempo arrojó una media de **13.72 segundos** con una desviación estándar de **0.91 segundos**. El tiempo mínimo de recuperación fue de 12.44 segundos y el máximo de 18.20 segundos. Observando por asignatura, los tiempos promedio de recuperación fueron:

- Biología: 13.47 ± 0.73 s
- Física: 13.95 ± 0.91 s
- Historia: 14.30 ± 1.06 s
- Lengua: 13.15 ± 0.36 s

Estos valores indican que el *overhead* introducido por el sistema RAG es relativamente consistente, aunque con ligeras variaciones entre asignaturas, siendo Lengua la que presentó el menor tiempo de recuperación y la menor variabilidad, e Historia el mayor. Esta latencia es un componente fijo del tiempo total de respuesta, independientemente del LLM utilizado.

Tiempo de Procesamiento del LLM (`llm_processing_time`).

Esta métrica mide el tiempo que cada modelo tardó en generar una respuesta una vez recibido el prompt con el contexto RAG. La Tabla 7.16 resume los promedios globales y sus desviaciones estándar para cada LLM.

Proveedor LLM	**Tiempo Procesamiento LLM** *(media ± std)*
Llama 3.1 70B Instruct	1,69 ±1.768
GPT-4o	2,73 ±1.406
Gemini 1.5 Pro	2,68 ±1.454
Mixtral 8x22B Instruct	3,92 ±2.817

Cuadro 7.16: Promedio y Desviación Estándar Global del Tiempo de Procesamiento del LLM (segundos)

Globalmente, **Llama 3.1 70B Instruct** fue el modelo más rápido, con un tiempo de procesamiento promedio de 1.69 segundos. Le siguen **Gemini 1.5 Pro** (2.68 s) y **GPT-4o** (2.73 s) con tiempos muy similares entre sí. **Mixtral 8x22B Instruct** fue el más lento, con un promedio de 3.92 segundos. Es notable la alta desviación estándar de Mixtral, sugiriendo una gran variabilidad en sus tiempos de respuesta, así como la de Llama 3.1, que a pesar de ser el más rápido en promedio, también presentó una dispersión considerable. GPT-4o y Gemini 1.5 Pro mostraron una consistencia similar en sus tiempos.

Proveedor LLM	**Tipo de Tarea**	**Tiempo Promedio (s)**
Gemini 1.5 Pro	RAG Q&A	1.49
	RAG Question Generation	4.14
	RAG Summarization	2.42
GPT-4o	RAG Q&A	1.99
	RAG Question Generation	3.21
	RAG Summarization	3.00
Llama 3.1 70B Instruct	RAG Q&A	1.41
	RAG Question Generation	1.90
	RAG Summarization	1.76
Mixtral 8x22B Instruct	RAG Q&A	2.37
	RAG Question Generation	6.49
	RAG Summarization	2.90

Cuadro 7.17: Promedio Global del Tiempo de Procesamiento del LLM por Tipo de Tarea (segundos)

Analizando por tipo de tarea a nivel global (ver Tabla 7.17), se observa que las tareas de `RAG Question Generation` fueron consistentemente las más lentas para la mayoría de los modelos (especialmente para Gemini 1.5 Pro y Mixtral 8x22B, superando los 4 y 6 segundos respectivamente), mientras que las tareas de `RAG Q&A` y `RAG Summarization` tendieron a ser más rápidas. Llama 3.1 70B mantuvo

su rapidez relativa en todas las categorías de tareas. Podemos observar mejor la diferencia de tiempos gracias a la Figura 7.5.

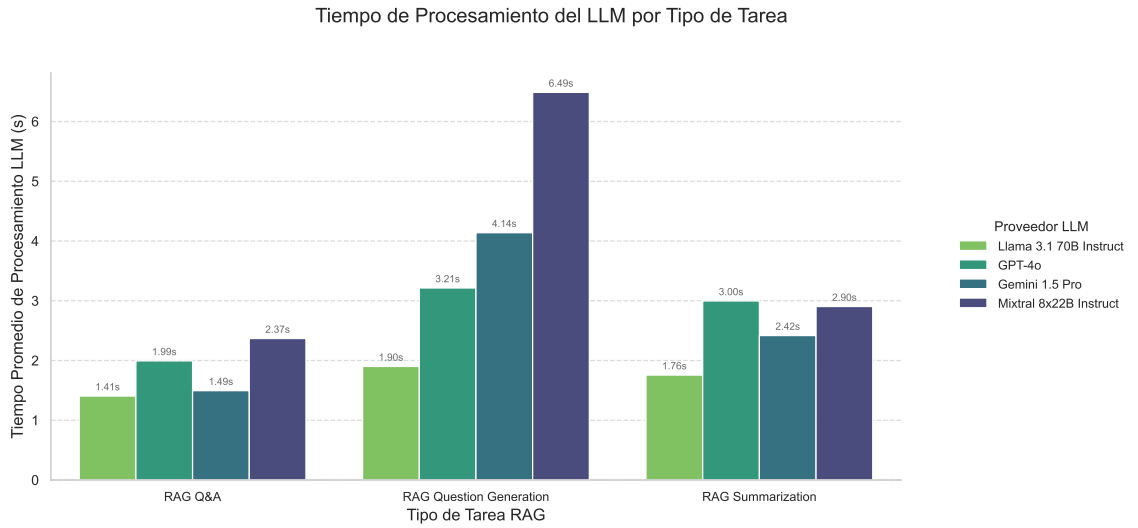

Figura 7.5: Tiempo Promedio Estimado por Tarea (Segundos).

Tiempo Total de Tarea (`total_task_time`).

Este tiempo, que es la suma del `retrieval_time` y el `llm_processing_time`, representa la latencia total experimentada por el usuario. La Tabla 7.18 muestra los promedios globales.

Cuadro 7.18: Promedio Global de la Latencia Total por LLM (segundos)

Proveedor LLM	**Latencia Total Promedio (s)**
Llama 3.1 70B Instruct	**15.41**
Gemini 1.5 Pro	16.40
GPT-4o	16.45
Mixtral 8x22B Instruct	**17.64**

Considerando la latencia total, **Llama 3.1 70B Instruct** se confirma como la opción más rápida con **15.41 s**. Le siguen muy de cerca **Gemini 1.5 Pro** (**16.40 s**) y **GPT-4o** (**16.45 s**), con una diferencia mínima entre ellos. Finalmente, **Mixtral 8x22B Instruct** (**17.64 s**) se posiciona como el más lento en promedio.

Es importante notar que, dado que el tiempo de recuperación RAG (13.72 s en promedio) constituye la mayor parte del tiempo total, las diferencias en el tiempo de procesamiento del LLM, aunque significativas entre modelos, tienen un impacto proporcionalmente menor en la latencia total percibida por el usuario. No obstante, para tareas que puedan requerir múltiples interacciones o generaciones extensas, incluso diferencias de 1-2 segundos en el tiempo del LLM pueden acumularse.

Estos datos de rendimiento temporal son esenciales para la selección del modelo, especialmente al considerar la experiencia del usuario y los posibles cuellos de botella en un sistema interactivo.

La completa evaluación comparativa detallada en este capítulo ha ofrecido una visión completa del rendimiento, la fiabilidad, la eficiencia y los costes asociados a cada configuración LLM+RAG analizada. Estos resultados detallados, que abarcan desde el análisis cualitativo de las respuestas en diversas asignaturas hasta las métricas de coste económico y tiempos de inferencia, proporcionan una base sólida para poder comparar los modelos. Por lo tanto, el próximo capítulo se centrará reunir las observaciones más importantes y presentar las conclusiones generales de la investigación, para responder a los objetivos planteados al inicio de este estudio y ofrecer una valoración final sobre la capacidad de cada modelo para el asistente académico propuesto.

CAPÍTULO 8

Conclusiones

La presente investigación se propuso evaluar y comparar de manera exhaustiva el rendimiento de cuatro modelos de lenguaje de gran escala (LLM) — **GPT-4o**, **Gemini 1.5 Pro**, **Mixtral 8x22B Instruct** y **Llama 3.1 70B Instruct**— integrados en una arquitectura de Generación Aumentada por Recuperación (RAG) para un asistente académico. La estrategia RAG implementada se fundamentó en la recuperación de información relevante de una base de conocimiento externa para aumentar los prompts enviados a los LLM. Este proceso, que incluyó una fase de búsqueda semántica, léxica y *reranking* con una latencia promedio de 13.72 segundos, demostró ser crucial para contextualizar las respuestas. La integración RAG contribuyó significativamente a reducir las alucinaciones y mejorar la fiabilidad al anclar las respuestas en información fáctica, permitió a los modelos utilizar contenido específico de las asignaturas (Biología, Física, Historia, Lengua y Literatura) y mejoró de forma generalizada la calidad de las respuestas, facilitando la formulación de resultados más precisos, coherentes y relevantes para las tareas académicas propuestas. A pesar de la latencia introducida, su papel fue fundamental para habilitar un análisis comparativo sobre cómo los diferentes LLM interactúan con un contexto externo.

El análisis comparativo reveló perfiles de rendimiento distintivos. **GPT-4o** emergió como el líder en la mayoría de las dimensiones cualitativas, obteniendo consistentemente las puntuaciones más altas en Exactitud Factual (4.82 global), Calidad de Síntesis (4.51) y Coherencia Narrativa (4.93), con una perfección constante (5.00) en Corrección Gramatical y Estilo. Su capacidad para integrar la información del RAG fue la más adepta y presentó la tasa de alucinaciones más baja (1.39 %). Si bien fue el modelo más costoso por tarea ($0.005869), su eficiencia temporal, con un tiempo total de tarea promedio de 15.82 segundos, fue competitiva. En resumen, GPT-4o es el modelo de mayor calidad y fiabilidad, ideal si el rendimiento es crítico y el coste es secundario.

Por su parte, **Gemini 1.5 Pro** se posicionó como un modelo muy competente y equilibrado. Mostró un rendimiento sólido con excelente Coherencia Narrativa (4.86) y Exactitud Factual (4.74), utilizando el contexto RAG de manera "Predominantemente Excelente" y presentando una baja incidencia de alucinaciones (5.56 %). Ofreció un coste significativamente menor que GPT-4o ($0.002560 por tarea) con un tiempo total de tarea de 16.62 segundos. Gemini 1.5 Pro representa, por tanto, una excelente alternativa que combina alta calidad y fiabilidad con un coste más moderado, siendo una opción robusta y versátil.

Mixtral 8x22B Instruct ofreció un rendimiento sólido, destacando por una Corrección Gramatical y Estilo perfecto (5.00), aunque con mayor variabilidad en Exactitud Factual (4.51) y Calidad de Síntesis (4.25). Presentó la tasa de alucinaciones más alta (9.72 %), con tendencia a complementar la información del RAG con conocimiento externo. Su coste por tarea ($0.002774) fue ligeramente superior al de Gemini y fue el modelo más lento en procesamiento LLM, resultando en el tiempo total de tarea más largo (17.86 s). Mixtral es un modelo capaz, pero su mayor variabilidad y la tasa más alta de alucinaciones, junto con su menor velocidad, sugieren cautela si la fiabilidad y rapidez son prioritarias.

Finalmente, **Llama 3.1 70B Instruct** se caracterizó por ser el más rápido y económico. Aunque su Corrección Gramatical y Estilo fue muy alta (4.94), presentó los promedios globales más bajos en Exactitud Factual (4.36), Calidad de Síntesis (3.92) y Coherencia Narrativa (4.49), con las mayores desviaciones estándar. Su uso del contexto RAG fue el más variable y tuvo una tasa de alucinaciones considerable (8.33 %). Fue el modelo más económico ($0.001724 por tarea) y el más rápido en procesamiento (tiempo total de tarea de 15.60 s). Llama 3.1 70B Instruct es atractivo en coste y velocidad, pero estos beneficios conllevan una menor calidad y fiabilidad en el contenido, lo que podría limitar su uso en contextos académicos de alta precisión.

En definitiva, la elección del LLM óptimo para un asistente académico basado en RAG dependerá del balance específico que se busque entre calidad de respuesta, fiabilidad, velocidad y coste. GPT-4o ofrece el máximo rendimiento cualitativo a un coste elevado; Gemini 1.5 Pro emerge como una opción muy equilibrada y fiable; Llama 3.1 70B Instruct destaca por su economía y rapidez; y Mixtral 8x22B Instruct ofrece un rendimiento competente con ciertas reservas. Esta investigación subraya la importancia de considerar no solo las capacidades intrínsecas de los LLM, sino también cómo su integración con arquitecturas RAG y las características del proveedor de API impactan en su viabilidad práctica.

De cara a futuras líneas de trabajo, se identifican varias áreas prometedoras para refinar y expandir los hallazgos de esta investigación. Sería beneficioso explorar **técnicas de *prompt engineering*** más avanzadas y adaptadas específicamente a las fortalezas y debilidades de cada modelo LLM, lo que podría optimizar aún más la calidad de las respuestas. En cuanto al componente RAG, investigar el impacto de diferentes **estrategias de *chunking*** (división de documentos) y el uso de modelos de ***embedding*** más sofisticados o especializados podría mejorar la relevancia y precisión de la información recuperada. Adicionalmente, la **incorporación y explotación de metadatos** asociados a los fragmentos de contexto (como fechas,

fuentes, temas o niveles de dificultad) podría permitir un filtrado más inteligente en la fase de recuperación y ofrecer a los LLM información contextual adicional para mejorar la calidad y adecuación de sus respuestas. Estos metadatos también podrían utilizarse para realizar análisis más granulares sobre el rendimiento de los modelos en subconjuntos específicos de datos. Estas futuras investigaciones contribuirían a desarrollar asistentes académicos aún más efectivos, fiables y adaptados a las necesidades del entorno educativo.

La elaboración de este TFM ha supuesto un significativo desarrollo de competencias en múltiples facetas.

A nivel **profesional**, este trabajo me ha permitido profundizar en el manejo práctico de tecnologías de IA de vanguardia, como los LLM (GPT-4o, Gemini, Mixtral, Llama) y las arquitecturas RAG. He fortalecido mis habilidades en la integración de sistemas y la aplicación de técnicas de procesamiento de lenguaje natural. La gestión del proyecto, desde la planificación y ejecución hasta el análisis y la presentación de resultados, ha sido clave para consolidar mis capacidades organizativas y de resolución de problemas técnicos.

A nivel **personal**, he desarrollado una mayor resiliencia ante los desafíos técnicos y la habilidad para resolver problemas complejos de manera metódica. Asimismo, el rigor metodológico requerido y la necesidad de plasmar los resultados de investigación de forma clara han fortalecido significativamente mis habilidades analíticas y de comunicación.

CAPÍTULO 9

Anexo

Este anexo contiene la implementación completa del sistema de Recuperación Aumentada por Generación (RAG) desarrollado para este Trabajo de Fin de Máster.

El código fuente completo desarrollado para este Trabajo Fin de Máster se encuentra disponible públicamente en el siguiente repositorio de GitHub:

https://github.com/pablotgp/RAG-

Índice de figuras

Índice de cuadros

Bibliografía

[1] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell, *On the dangers of stochastic parrots: Can language models be too big?*, Proceedings of the 2021 ACM conference on fairness, accountability, and transparency, 2021.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, ..., and Dario Amodei, *Language models are few-shot learners*, Advances in neural information processing systems 33, 2020.

[3] Gemini Team, Google, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, Soroosh Mariooryad, Yifan Ding, Xinyang Geng, Fred Alcober, Roy Frostig, Mark Omernick, Lexi Walker, Cosmin Paduraru, Christina Sorokin, ..., and Oriol Vinyals, *Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context*, 2024.

[4] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, ..., and Zhiyu Ma, *The Llama 3 herd of models*, 2024.

[5] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung, *Survey of hallucination in natural language generation*, ACM Computing Surveys **55** (2023mar), no. 12.

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

[7] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela, *Retrieval-augmented generation for knowledge-intensive NLP tasks*, 2020.

[8] Mistral AI team, *Cheaper, better, faster, stronger*, 2024. Accessed on 2024-06-03.

[9] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman, *WebGPT: Browser-assisted question-answering with human feedback*, 2022.

[10] OpenAI, *GPT-4o System Card*, 2024.

[11] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, ..., and Thomas Scialom, *Llama 2: Open foundation and fine-tuned chat models*, 2023.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin, *Attention is all you need*, Advances in neural information processing systems 30, 2017.

[13] James Vincent, *After controversy, meta pulls demo of AI model that writes scientific papers*, 2022.

[14] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus, *Emergent abilities of large language models*, 2022.

[15] Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang, *CacheBlend: Fast large language model serving for RAG with cached knowledge fusion*, 2024.