

Leveraging Retrieval-Augmented Generation Architecture for Clinical Guidance: Building a Chatbot to Assist Rheumatology Physicians



Universitat Oberta
de Catalunya

Daniel Camacho Montaña

Machine Learning

Màster en Bioestadística y bioinformática

Nombre del director/a de TF:
Alfredo Madrid García

Nombre del/de la PRA:
Agnès Pérez Millan

23 de marzo de 2025



Esta obra esta sujeta a una licencia de Reconocimiento
<https://creativecommons.org/licenses/by-nc/3.0/es/>

Ficha Del Trabajo Final

| | |
|-------------------------------------|--|
| Título del trabajo: | Leveraging Retrieval-Augmented Generation Architecture for Clinical Guidance: Building a Chatbot to Assist Rheumatology Physicians |
| Nombre del autor/a: | Daniel Camacho Montaña |
| Nombre del director/a de TF: | Alfredo Madrid García |
| Nombre del/de la PRA: | Agnès Pérez Millan |
| Fecha de entrega: | 23 de marzo de 2025 |
| Titulación o programa: | Màster en Bioestadística y bioinformàtica |
| Àrea del trabajo final: | Machine Learning |
| Idioma del trabajo: | Castellano |
| Palabras clave: | Inteligencia artificial, chatbot, guías clínicas, reumatología, sistema de ayuda a la decisión |

Resumen del trabajo

Màxim 250 paraules, amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball.

Abstract

A maximum of 250 words, detailing the purpose, context of application, methodology, results and conclusions of the work.

Índice general

| | |
|--|-----------|
| Siglas | 9 |
| 1. Introducción | 11 |
| 1.1. Contexto y justificación del trabajo | 11 |
| 1.2. Objetivos del trabajo | 12 |
| 1.2.1. Objetivos Primarios | 12 |
| 1.2.2. Objetivos Secundarios | 13 |
| 1.3. Impacto en sostenibilidad, ético-social y de diversidad | 13 |
| 1.3.1. Sostenibilidad y medio ambiente | 13 |
| 1.3.2. Socio-ético | 14 |
| 1.3.3. Diversidad y derechos humanos | 15 |
| 1.4. Enfoque y metodología seguida | 15 |
| 1.5. Planificación del trabajo | 16 |
| 1.5.1. Distribución de tareas | 16 |
| 1.5.2. Diagrama de Gantt | 18 |
| 1.5.3. Análisis de riesgos | 18 |
| 1.6. Breve resumen de los productos obtenidos | 20 |
| 1.7. Breu descripció dels altres capítols de la memòria | 21 |
| 2. Estado del arte | 22 |
| 2.1. Introducción a LLM | 22 |
| 2.1.1. Historia de los LLM | 22 |
| 2.1.2. LLM: fundamentos, estado actual y familias | 24 |
| 2.1.3. Limitaciones de los LLM | 25 |
| 2.2. Sistemas Retrieval-Augmented Generation (RAG) en LLM | 26 |
| 2.2.1. Arquitectura de RAG. Workflow | 26 |
| 2.2.2. Tipos de RAG en la actualidad | 29 |
| 2.2.3. Retos y límites de RAG, direcciones de futuro | 30 |
| 3. Marco teórico | 33 |
| 3.1. Pre-procesamiento | 33 |
| 3.1.1. Introducción | 33 |
| 3.1.2. Framework | 33 |
| 3.1.3. Parseo de Guías Clínicas | 34 |
| 3.1.4. Chunking, la fragmentación de documentos | 35 |

| | | |
|-----------|---|-----------|
| 3.1.5. | Representación vectorial: embeddings y vectorstores | 37 |
| 3.1.6. | RAG: Retrievers y Generators | 41 |
| 4. | Materiales y métodos | 42 |
| 4.1. | Materiales | 42 |
| 4.1.1. | Conjuntos de datos | 42 |
| 4.1.2. | Herramientas y tecnologías | 43 |
| 4.2. | Métodos | 43 |
| 4.2.1. | Preprocesamiento de los documentos | 43 |
| 4.2.2. | Representación y almacenamiento de la información | 43 |
| 4.2.3. | Recuperación de información | 43 |
| 4.2.4. | Generación de respuestas | 43 |
| 4.2.5. | Evaluación de sistema | 44 |
| 5. | Valoració econòmica | 45 |
| 6. | Referencias | 46 |
| .1. | Anexo 1. Preguntas PEC2 | 52 |
| .2. | Anexo 2. Tabla de metadatos | 53 |

Índice de figuras

| | |
|---|----|
| 1.1. Planificación temporal del trabajo final de máster | 18 |
| 2.1. Historia y evolución de los distintos modelos de procesamiento de lenguaje natural. Extraída de (1). | 23 |
| 2.2. Esquema de una arquitectura RAG. Extraído de (2). | 27 |
| 2.3. Workflow de arquitectura RAG. Extraído de (3) | 31 |
| 3.1. Comparación del efecto del chunking. Extraído de (4) | 36 |

Índice de cuadros

Siglas

ACR American College of Rheumatology

API Application Programming Interface

CCEG Competencia en Compromiso Ético y Global

ERM Enfermedades Reumáticas y Musculoesqueléticas

EULAR European Alliance of Associations for Rheumatology

FAISS Facebook AI Similarity Search

GDPR General Data Protection Regulation

GPT Generative Pre-trained Transformer

GPU Graphics Processing Unit

GROBID GeneRation Of Bibliographic Data

HIPAA Health Insurance Portability and Accountability Act

HTML HyperText Markup Language

IA Inteligencia Artificial

JSON JavaScript Object Notation

LlaMA Large Language Model Meta AI

LLM Large Language Model

ML Machine Learning

MTEB Massive Text Embedding Benchmark

NLM Neural Language Models

NLP Natural Language Processing

ODS Objetivos de Desarrollo Sostenible

PDF Portable Document Format

PLM Pre-trained Language Models

RAG Retrieval-Augmented Generation

RLHF Reinforcement Learning from Human Feedback

SFT Supervised Fine-Tuning

SLM Statistical Language Model

STS Semantic Textual Similarity

TEI Text Encoding Initiative

TXT Text File Format

XML Extensible Markup Language

Capítulo 1

Introducción

1.1. Contexto y justificación del trabajo

Las Enfermedades Reumáticas y Musculoesqueléticas (ERM) engloban un amplio espectro de patologías crónicas que afectan principalmente al aparato locomotor, aunque también pueden comprometer otros órganos y sistemas, afectando comúnmente las articulaciones de individuos de todas las edades y géneros, pero también pueden afectar los músculos, órganos internos y otros tejidos. La elevada prevalencia, el dolor y las complicaciones asociadas las sitúan entre las principales causas de deterioro en la calidad de vida a nivel global. La complejidad clínica, sumada a la variabilidad en las manifestaciones y en las respuestas al tratamiento, subraya la necesidad de contar con herramientas avanzadas que faciliten la toma de decisiones, teniendo en cuenta la última evidencia disponible de las mayores asociaciones, European Alliance of Associations for Rheumatology (EULAR), y American College of Rheumatology (ACR).

En este contexto, la Inteligencia Artificial (IA) y los modelos de Lenguaje de Gran Tamaño (Large Language Model (LLM)) han revolucionado la forma en que se procesa y genera información a partir de datos textuales complejos. En la última década, los modelos de LLM han transformado radicalmente la interacción entre humanos y máquinas, marcando un hito en el desarrollo de la inteligencia artificial. Herramientas comerciales como ChatGPT han demostrado una notable capacidad para comprender y generar texto de manera coherente y versátil, facilitando su adopción en prácticamente todos los sectores. Esta revolución responde a la creciente necesidad de sistemas eficientes capaces de procesar grandes volúmenes de información y proporcionar respuestas rápidas, optimizando el flujo de trabajo y mejorando la toma de decisiones en diversas disciplinas.

El reciente avance de los LLM ha impulsado el desarrollo de soluciones inteligentes para la generación y recuperación de información en múltiples áreas. En el ámbito médico, donde el acceso a información precisa, actualizada y basada en evidencia es crucial, estas herramientas prometen transformar la toma de decisiones clínicas, la educación médica y la investigación. Sin embargo, los modelos fundacionales presentan limitaciones importantes, como la tendencia a generar respuestas incompletas o alucinaciones, lo que puede comprometer la precisión en entornos donde la fiabilidad es fundamental (5). Además, es importante considerar que los LLM fundacionales adquieren su conocimiento mediante un preentrenamiento computacionalmente costoso que, si bien es eficaz en tareas generales, no permite incorporar información nueva y actualizada sin llevar a cabo un nuevo entrenamiento, lo que hace que su conocimiento

permanezca estático.

Para abordar estas limitaciones, la técnica RAG combina la generación del lenguaje con la recuperación de documentos relevantes, usados como fuentes externas de conocimiento. Esta técnica permite al modelo acceder a una base de conocimiento externa al entrenamiento, como las guías de prácticas clínicas oficiales y verificadas, asegurando que las respuestas generadas sean más precisas y confiables, reduciendo las alucinaciones y proporcionando una respuesta más fundamentada.

Las guías de práctica clínica, a su vez, son documentos normativos elaborados por instituciones de salud y sociedades científicas que contienen las últimas recomendaciones e indicaciones basadas en una evidencia demostrada. Por lo tanto, integrar estas fuentes de información en una arquitectura RAG permite mejorar la calidad de las respuestas en aplicaciones como asistentes médicos virtuales, permitiendo dar apoyo en decisiones clínicas (6; 7).

El uso de RAG en guías médicas permitiría resolver necesidades críticas en el ámbito médico, ya que se reducirían los errores en la generación de texto por alucinaciones al integrar un sistema de recuperación de documentos (2). También permitiría a los profesionales de la salud e investigadores un acceso rápido y contextualizado a información basada en evidencia y referenciada propiciando una mejora en la toma de decisiones clínicas.

1.2. Objetivos del trabajo

El objetivo principal de este trabajo de fin de máster es la creación de un sistema RAG que incluya guías de práctica clínica de reumatología, mejorando el diagnóstico, el tratamiento y facilitando la consulta médica para maximizar la probabilidad de diagnóstico y tratamientos correctos. El trabajo se centrará en el desarrollo de un modelo que sirva como base para un asistente virtual accesible tanto para expertos del ámbito sanitario (e.g., médicos de atención primaria, enfermeros...) como para pacientes, capaz de responder preguntas con alta precisión basándose en el contenido de las guías médicas del área de reumatología.

Para alcanzar este objetivo general, se plantean los siguientes objetivos específicos:

1.2.1. Objetivos Primarios

1. **Integración de guías clínicas de reumatología en un modelo de lenguaje basado en una arquitectura RAG**
2. **Evaluación del rendimiento del sistema RAG en comparación con un modelo de lenguaje fundacional (LLM):** análisis comparativo de la precisión, eficiencia y calidad de las respuestas generadas por el modelo RAG frente a un modelo de lenguaje fundacional.
3. **Desarrollo de una interfaz visual:** implementación de una interfaz gráfica intuitiva que integre el modelo RAG optimizado, proporcionando a profesionales médicos y pacientes una herramienta automatizada para el diagnóstico y la consulta de guías médicas. Esta interfaz permitirá respuestas rápidas y orientadas al usuario, mejorando el acceso a la información y reduciendo el tiempo de acción.

1.2.2. Objetivos Secundarios

1.1 Objetivos específicos del objetivo 1

- 1.1 **Identificación, recopilación y creación de un conjunto de datos a partir de documentos PDF** sobre patologías reumáticas.
- 1.2 **Construcción de un dataset especializado para un modelo RAG**: creación de un conjunto de datos basado en guías médicas verificadas, que servirá como base de conocimiento para el sistema de recuperación de información del modelo RAG.
- 1.3 **Evaluación de técnicas de procesamiento de texto**: análisis y selección de métodos de parseo, segmentación (chunking) y bases de datos vectoriales para el tratamiento de los documentos de las guías clínicas.

2.2 Objetivos específicos del objetivo 2

- 2.1 **Revisión del estado del arte de RAG en el ámbito médico**: revisión y análisis de las últimas implementaciones de modelos RAG aplicados a guías médicas, evaluando sus capacidades y limitaciones en este contexto.
- 2.2 **Comparación de diferentes arquitecturas RAG**: evaluación y contraste de modelos actuales en términos de métricas de rendimiento, precisión, coherencia y capacidad de adaptación en la generación de texto.

1.3. Impacto en sostenibilidad, ético-social y de diversidad

Esta sección analiza los posibles impactos, tanto positivos como negativos, derivados de esta tesis, considerando las tres dimensiones establecidas por la Competencia en Compromiso Ético y Global (CEEG): sostenibilidad, dimensión socioética y diversidad. El análisis se enmarca dentro de los Objetivos de Desarrollo Sostenible (ODS), garantizando un enfoque basado en la responsabilidad y la ética académica.

1.3.1. Sostenibilidad y medio ambiente

La creciente revolución de la IA ha transformado la industria médica, ofreciendo mejoras significativas en diferentes áreas, como la precisión diagnóstica, la eficiencia y los resultados de los pacientes (8). Sin embargo, estas tecnologías requieren de un gran consumo de energía, y, consecuentemente, una alta producción de CO₂ ambiental, llegando, en ocasiones, superar el consumo ahorrado (8). De hecho, un estudio reciente concluyó que para el entrenamiento de un solo modelo de IA, se generaron cerca de 285.000kg de CO₂, siendo una producción mayor que la de 5 automóviles en toda su vida útil (9).

Además, el funcionamiento de los modelos requieren de agua para enfriar los centros de datos y generar electricidad para proveerlos de energía, aumentando drásticamente el consumo de agua en los últimos años. Se plantea que, para 2027, la demanda de agua relacionada al uso de IA será de 6.600 millones de metros cúbicos.(9). Durante el entrenamiento de uno de los

modelos Generative Pre-trained Transformer (GPT), GPT-3 llegó a consumir 700.000 litros de agua dulce, contribuyendo en gran medida a la huella hídrica relacionada con el uso de IA (10).

Asimismo, la creciente demanda unidades de procesamiento gráfico (Graphics Processing Unit (GPU)) conlleva un impacto ambiental significativo, ya que su producción requiere la extracción de metales como el litio, el cobalto y el níquel (11). Estos materiales son esenciales para la fabricación de otros componentes electrónicos avanzados, pero su obtención implica procesos de minería intensiva que pueden provocar deforestación, contaminación y pérdida de biodiversidad.

Por otro lado, los modelos RAG presentan una alternativa más sostenible, ya que no requieren un reentrenamiento constante para mantenerse actualizados. Los modelos actuales requieren un entrenamiento costoso y computacionalmente intensivo, lo que dificulta su actualización frecuente. En cambio, RAG permite acceder a información actualizada sin la necesidad de desarrollar y entrenar nuevos modelos específicos de LLM, lo que se traduce en una reducción significativa del impacto ambiental asociado a estos procesos. Al disminuir la demanda de recursos computacionales, como las GPU, también se reduce la necesidad de extraer materiales críticos como el litio, el cobalto y el níquel, mitigando así las graves consecuencias ambientales derivadas de la minería intensiva, como la deforestación, la contaminación y la pérdida de biodiversidad.

1.3.2. Socio-ético

Actualmente, el uso de la IA en el ámbito de la salud se centra en la obtención de resultados, mientras que su aplicación en el diagnóstico y las consultas médicas basadas en guías médicas está en desarrollo. De hecho, el uso de modelos LLM y los asistentes virtuales no especializados presentan una baja precisión, ya que no siempre se basan en documentos verificados. (12)

Sin embargo, la implementación de recuperación en la generación de respuestas permitiría el acceso a información actualizada en tiempo real. Además, al proporcionar la visualización de las fuentes de utilizadas mejorarían la transparencia en la toma de decisiones médicas y reducirían el riesgo de desinformación. (13) y (14). Se debe tener en cuenta, que debido a la importancia de la precisión en un área tan delicada como la medicina y el diagnóstico, se han formulado modelos RAG específicos para áreas como biomedicina (15), hepatología (16; 6) o cirugía (17), con documentación médica, como guías médicas hepatológicas, pero no en patologías reumatológicas (6).

Finalmente, los asistentes virtuales basados en IA requieren de una infraestructura potente, no siempre disponible en todos los hospitales, especialmente en regiones con menos recursos. En este sentido, aunque un modelo RAG depende de un modelo de lenguaje de gran tamaño (LLM), este puede ser alojado en la nube, mientras que la base de datos vectorial y los sistemas de recuperación de información pueden mantenerse en servidores locales. Esta arquitectura híbrida facilita el despliegue en áreas rurales donde el acceso a médicos especializados es limitado, permitiendo la provisión de diagnósticos precisos y oportunos sin necesidad de infraestructura computacional avanzada en el sitio. De esta manera, contribuiría a reducir la brecha digital existente, permitiendo que centros de salud con menos recursos también se beneficien de los avances en inteligencia artificial médica (18).

Además, un modelo RAG con memoria podría facilitar el acceso a la IA por parte de usuarios con distintos niveles de conocimiento, adaptándose a sus necesidades específicas. Esto sería

posible mediante el uso de técnicas de prompt engineering, ajustando la formulación de las respuestas en función del contexto y la situación particular del usuario. De esta manera, se democratizaría su uso más allá del ámbito profesional, haciéndolo accesible también a pacientes y cuidadores.

1.3.3. Diversidad y derechos humanos

Se van a usar guías clínicas elaboradas por comités científicos y sociedades científicas, caracterizadas por su rigor y un gran sentido de la diversidad, caracterizadas por su rigor, en las cuales se presupone que el panel de elaboración es diverso. Por lo tanto, el uso de estas guías mediante RAG puede contribuir a reducir el sesgo, asegurando que la información proporcionada esté basada en el consenso de expertos y representando adecuadamente a distintos grupos y contextos clínicos (19).

Este enfoque facilita el acceso equitativo a información médica confiable, mejorando la atención en comunidades marginadas. Además, al adaptar sus recomendaciones en función del usuario, el sistema permite que información médica compleja sea comprensible para cualquier persona (19).

La arquitectura RAG no manejará datos confidenciales de pacientes ni historiales clínicos, limitándose a utilizar información procedente de fuentes validadas y guías clínicas públicas, y su implementación se ajusta plenamente a las normativas vigentes en materia de protección de datos, como el General Data Protection Regulation (GDPR) o la Health Insurance Portability and Accountability Act (HIPAA), garantizando la privacidad y seguridad de la información (20; 21).

1.4. Enfoque y metodología seguida

Los pasos realizados para establecer el modelo RAG fueron los siguientes:

1. **Estudio de la literatura:** Se iniciará con un estudio del estado del arte sobre LLM y los modelos RAG desarrollados en el ámbito médico.
2. **Preparación del dataset:** Se obtendrán las guías médicas en formato Portable Document Format (PDF), se realizará un análisis y descomposición de texto para obtener información (parseo), se fragmentarán (chunking), se realizará una transformación vectorial (embedding) y se creará el repositorio de vectores.
3. **Selección del modelo:** Se estudiarán los diferentes modelos de LLM como generadores de respuesta y los diferentes retrievers disponibles comercialmente. Se evaluará su capacidad de recuperar información a partir del vectorstore, la precisión y rendimiento.
4. **Modelo RAG:** Con los modelos seleccionados, se formulará la arquitectura RAG adecuada para el proceso de los datos. El proceso se enfocará en mejorar las respuestas de LLM fundacionales.
5. **Evaluación del modelo:** Se estudiará el rendimiento del modelo mediante métricas de precisión, comparando las respuestas proporcionadas por el modelo respecto a preguntas de exámenes oficiales y comparándolas con el solucionario oficial.

6. **Interfaz gráfica web:** La arquitectura finalmente diseñada se integrará a una interfaz web (local) que permitirá la interacción usuario-asistente virtual, haciendo uso de herramientas como Streamlit, gradio,...

1.5. Planificación del trabajo

Se han tenido en cuenta las fechas de entrega de las PEC para la distribución de trabajo en 4 bloques. En la figura Figura 1.1 se muestra la distribución del tiempo de las tareas a realizar en cada bloque, descritas a continuación:

1.5.1. Distribución de tareas

Para lograr los objetivos de este trabajo final de máster, se definieron las siguientes tareas

1. Investigación y revisión de literatura (3 semanas)

El objetivo es realizar una profunda revisión de la literatura, enfocada en los modelos LLM, la arquitectura RAG y sus aplicaciones previas en consultas médicas y guías clínicas.

- Identificar los modelos LLM candidatos y sus aplicaciones previas.
- Revisar la arquitectura RAG, sus componentes y sus aplicaciones.
- Resumir los hallazgos de la investigación.

2. Recopilación de documentos y procesamiento (2 semanas)

El objetivo es recopilar guías clínicas en formato PDF y procesarlas para posteriormente crear una base de datos vectorial a partir de la cual se recupere contenido relevante para fundamentar la respuesta del LLM. Se estudiarán los diferentes métodos de parseo disponibles y se realizarán pruebas para determinar el más eficiente para la conversión del formato PDF en texto plano

Debido a la complejidad de la transformación de imágenes en texto plano, solo se vectorizará el contenido textual, dejando de lado las figuras. Sin embargo, debido a la importancia de la información contenida en las tablas, sí que se incluirá dicha información, convirtiendo a formato tabular para evaluar si esto mejora la precisión del modelo.

- Data collection: Recopilar guías médicas verificadas
- Data processing: Identificar, aplicar y evaluar los métodos de parseo, segmentación (*chunking*) y vectorización.

3. Selección del modelo y análisis preliminar (2 semanas)

Se evaluarán diferentes arquitecturas RAG, combinando diversos métodos de recuperación (*retrievers*) y generación (*generators*) para seleccionar la mejor configuración para la arquitectura RAG del trabajo final de máster.

- Realizar pruebas iniciales para evaluar la idoneidad de cada modelo según los datos disponibles.

- Comparar los modelos basados en métricas de rendimiento.
- Documentar el proceso de selección y justificar la elección del modelo final.

4. Modelo RAG (3 semanas)

Se implementará la arquitectura seleccionada para procesar la información del vector store y generar respuestas relevantes basadas en las guías médicas. Se ajustarán los parámetros para tratar de optimizar la relevancia, configurando las estrategias de indexación y búsqueda, así como los parámetros iniciales del modelo.

- Seleccionar el modelo base para la recuperación de información y la generación de respuestas.
- Indexar los documentos utilizando técnicas de *embedding*.
- Entrenar el modelo con datos médicos específicos para especializar el sistema.

5. Evaluación del modelo y optimización (3 semanas)

Se analizarán métricas de rendimiento como la precisión y la capacidad del modelo para adaptar su lenguaje al usuario. Se optimizará el modelo ajustando los componentes del sistema necesarios.

- Evaluar la precisión y la coherencia del lenguaje generado.
- Analizar el rendimiento del modelo y aplicar optimizaciones.

6. Creación de una interfaz web basada en el mejor modelo (2 semanas)

Se diseñará una interfaz gráfica web con herramientas de prototipado como Streamlit o Gradio que integre el mejor modelo de arquitectura RAG, permitiendo la interacción del usuario con el sistema.

- Diseñar una interfaz web clara y fácil de usar.
- Integrar el modelo dentro de la interfaz.
- Evaluar la funcionalidad de la interfaz para asegurar su correcto funcionamiento.

7. Reporte final (3 semanas)

En esta fase se redactará la memoria final del trabajo final de máster, se diseñará y grabará la presentación, y se creará un repositorio en GitHub con el código empleado.

- Redactar la memoria final.
- Crear el repositorio en GitHub.
- Preparar y grabar la presentación.
- Realizar la entrega final de la tesis.

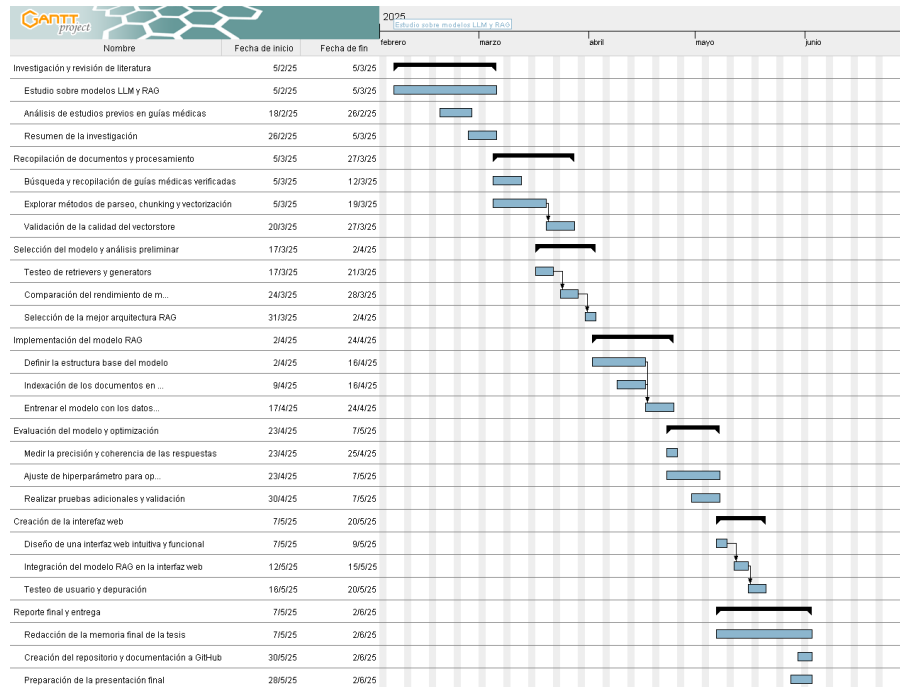


Figura 1.1: Planificación temporal del trabajo final de máster

1.5.2. Diagrama de Gantt

1.5.3. Análisis de riesgos

Antes de la elaboración del proyecto, se han contemplado los posibles riesgos que pueden surgir y causar un impacto negativo en la elaboración del trabajo, determinando el nivel de riesgo y las posibles alternativas para mitigarlos.

1. Problemas en la creación del repositorio

Impacto: Alto

El proceso de parseo es esencial para implementar un modelo RAG, ya que garantiza la extracción y estructuración óptima del texto de los documentos PDF originales. Sin embargo, este paso puede presentar problemas como la presencia de contenido visual (imágenes), formato desorganizado y pérdida de información.

Para mitigar estos riesgos, se establecen herramientas de parseo alternativas y paralelas para estudiar las diferencias y escoger la herramienta más adecuada. También se han seleccionado guías médicas con la estructura lo más similar y simple posible para evitar diferencias entre documentos que alteren el proceso.

2. Diversidad entre guías médicas

Impacto: Medio

Las guías médicas no fueron diseñadas específicamente para ser procesadas por modelos de análisis de datos, sino que su propósito es servir como referencia y apoyo para profe-

sionales de la salud. Aunque muchas están reconocidas por la ACR, cada una presenta una estructura particular, lo que dificulta su tratamiento automatizado.

Por ello, se deben considerar las posibles limitaciones al utilizarlas en un entorno computacional. Como estrategia de mitigación, se priorizarán las guías que permitan su integración en software de recuperación de información.

Además, cabe destacar que algunas guías clínicas pueden estar protegidas por derechos de autor, por lo que su uso en este trabajo se ha realizado exclusivamente con fines académicos, sin ningún propósito comercial.

3. Dificultades en la vectorización

Impacto: Medio

Los modelos RAG recuperan la información basándose en la proximidad vectorial, por lo que la correcta vectorización de la información impacta directamente en la calidad de la recuperación de documentos. La conversión del contenido de los documentos PDF a formato Text File Format (TXT) es fundamental para una representación adecuada de los datos. Una transformación deficiente puede inducir errores en la interpretación del texto.

Para optimizar este proceso, se realizará una segmentación eficiente de los documentos en fragmentos de tamaño óptimo (chunks) para preservar el contexto en la recuperación de información. Asimismo, se evaluarán diferentes enfoques de vectorización en paralelo para determinar la metodología más eficaz.

4. Dificultades en el modelo RAG

Impacto: Medio

El modelo RAG enfrenta desafíos de ajuste fino, interpretabilidad y eficiencia en la recuperación de documentos. Estos riesgos pueden clasificarse en dos factores. Primero, al tratarse de una arquitectura relativamente nueva, existen pocos ejemplos de referencia para guiar el desarrollo. Segundo, su demanda computacional requiere de GPU potentes, limitando las opciones de implementación, pero no resultan significativamente perjudiciales. Se debe tener en cuenta que, si la fase de retrieval no es efectiva, el modelo RAG generará respuestas basadas en documentos irrelevantes.

Para abordar estos problemas, se evaluarán distintas plataformas y alternativas a Google Colab, realizando pruebas en diferentes infraestructuras que se ajusten a las necesidades del proyecto. Además, se optimizará el modelo mediante ajustes de los componentes del sistema para mejorar la eficiencia computacional. Para mitigar la demanda computacional, una posible solución será utilizar APIs de modelos en la nube, lo que evitaría la necesidad de desplegar infraestructuras locales avanzadas, facilitando la implementación en entornos con recursos limitados.

5. Coste económico

En algunos LLM como GPT-4 o Claude3 tienen un coste asociado, que corresponde al tipo de modelo usado y el número de tokens requeridos, tanto en el prompt de entrada (input) como en la respuesta del LLM (output), limitando la complejidad del modelo.

Se comprobaron alternativas para establecer la más económica, que optimice el coste y mantenga la calidad del resultado, eligiendo modelos con un coste por token menor, como GPT-4o mini.

6. Evolución constante del ámbito médico

Impacto: Medio

El entorno médico y clínico está en constante evolución, creando y mejorando nuevas guías clínicas constantemente, por lo que es crucial que el sistema propuesto tenga la misma capacidad de evolucionar y acceder a información actualizada en tiempo real.

Como medida de mitigación, se trabajará exclusivamente con las guías más recientes disponibles hasta la fecha de marzo de 2025, asegurando que el sistema se base en información actualizada. Además, se priorizará la extracción de datos únicamente de fuentes fiables, evitando el uso de información no verificada.

7. Evaluación del sistema

Impacto: Medio

El modelo de arquitectura RAG proporcionará respuestas generadas por el LLM basándose en las guías médicas. Para evaluar la precisión y relevancia de estas respuestas, se requerirá la validación de especialistas médicos en enfermedades reumáticas y musculoesqueléticas.

Con el fin de reducir esta dependencia, se propondrá el uso de un LLM externo capaz de comparar y evaluar automáticamente las diferencias entre las respuestas, facilitando así el proceso de revisión y mejorando la eficiencia del análisis.

8. Tiempo de redacción de la memoria final

Impacto: Bajo

El tiempo de entrega del TFM puede verse afectado por posibles retrasos en el desarrollo y validación del modelo, comprometiendo la calidad del informe final. Este riesgo podría derivar en una documentación incompleta o una falta de análisis en profundidad.

Para mitigar este riesgo, se establecen cronogramas con hitos claros, revisiones periódicas del avance y la realización paralela de tareas clave para optimizar los recursos disponibles. Asimismo, se realizará la redacción de la memoria final a lo largo del trabajo, avanzando en paralelo.

1.6. Breve resumen de los productos obtenidos

Al finalizar el Trabajo Final de Máster, se habrá obtenido uno de los primeros modelos RAG basado en guías médicas verificadas que se podrá emplear mediante la plataforma web en formato de asistente virtual.

Se va a obtener:

- **Un conjunto de datos** con los cuales construir la base de datos vectorial

- Una **base vectorial** con los documentos procesados
- Una **interfaz web** que emplee el modelo RAG, basándose en las guías médicas para recuperar la información.
- Un **repositorio de GitHub** con el código empleado.
- Una **memoria** con la evolución del proyecto.
- Una **presentación virtual** con la tesis final.

1.7. Breu descripció dels altres capítols de la memòria

Breu explicació dels continguts de cada capítol i la seva relació amb el projecte global.

Capítulo 2

Estado del arte

2.1. Introducción a LLM

Los recientes avances en el campo de la inteligencia artificial (IA) ha permitido el desarrollo de modelos de Lenguaje de Gran Tamaño (Large Language Models, LLM) como Large Language Model Meta AI (LLaMA), ChatGPT o Claude, algoritmos potentes capaces de sintetizar grandes cantidades de datos y realizando una amplia variedad de tareas en respuesta a comandos humanos.

Los LLM, como GPT-3, funcionan mediante una combinación de Deep Learning, entrenamiento masivo con grandes volúmenes de texto y generación de texto basada en probabilidades. Específicamente, los LLM están basados en redes neuronales profundas, en la arquitectura Transformers introducida en 2017 [(22) y (23)], en que primero se produce una autoregresión, prediciendo la siguiente palabra en un texto dado un contexto previo, asignando probabilidades a cada palabra posible y eligiendo la más probable, para después realizar un entrenamiento con aprendizaje auto supervisado con grandes cantidades de texto, tratando de predecir palabras ocultas en una oración. (22).

2.1.1. Historia de los LLM

Los LLM sirven como piedra angular en el procesamiento del lenguaje natural (NLP, Natural Language Processing). De hecho, el lenguaje es la herramienta que permite expresar pensamientos y la comunicación entre dos entidades. Sin embargo, las máquinas carecen de la capacidad intrínseca de comprender y comunicarse en un lenguaje humano, por lo que requieren de algoritmos de IA para disponer de esta capacidad. El NLP surge a partir de esta necesidad, con un modelo de lenguaje en su núcleo que predice o genera la probabilidad de unidades lingüísticas en función del contexto (1). La evolución de los modelos lingüísticos permitieron desarrollar modelos con diferentes estados de complejidad, desde los modelos estadísticos iniciales (Statistical Language Model (SLM)) hasta los modelos lingüísticos de gran tamaño (los LLM).

De forma general, los modelos de lenguaje han sufrido diferentes evoluciones que han permitido el desarrollo de los modelos actuales, los LLM. En la Figura 2.1, se puede observar los modelos más relevantes de cada tipo de LM, desde el nacimiento de los más básicos (N-grams) hasta la actualidad (GPT-4).

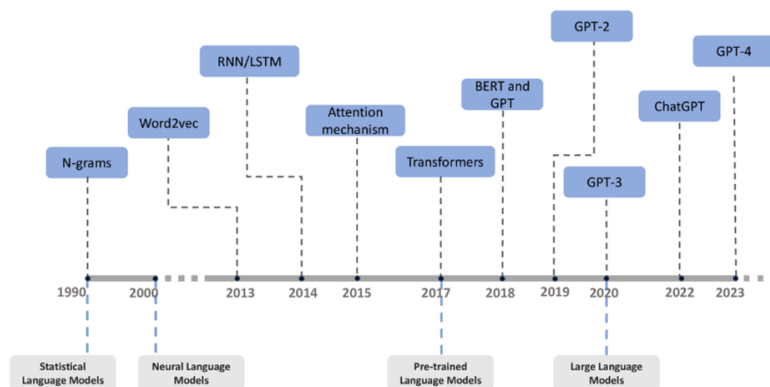


Figura 2.1: Historia y evolución de los distintos modelos de procesamiento de lenguaje natural. Extraída de (1).

- **Statistical Language Models (SLMs):** Son modelos matemáticos que emplean las propiedades contextualmente relevantes del lenguaje natural desde una perspectiva estadística probabilística. Su esencia radica en determinar la probabilidad de que una oración aparezca dentro de un contexto a partir de las probabilidades condicionales consecutivas de las unidades lingüísticas simples (palabras o letras) (24). Su mayor limitación radica en que, al tener solo en cuenta las dos primeras palabras precedentes a cada término, se pierde significativamente su precisión y contexto.
- **Neural Language Models (NLM):** Emplean redes neuronales para predecir las probabilidades de palabras posteriores dentro de secuencias lingüísticas, y son capaces de comprender el concepto de las palabras mediante vectorización y su ubicación. Es decir, convierten las palabras a lenguaje binario, se transforman en vectores con una ubicación fija, de manera que pueden simular las relaciones entre palabras a través de los ángulos entre los vectores. Su mayor limitación es que disponen de un contexto limitado y, pese a generar un texto coherente, no poseen una comprensión real del significado, llegando a generar respuestas sintácticamente correctas pero sin sentido real (25; 26).
- **Pre-trained Language Models (PLM):** Estos modelos se someten a un entrenamiento inicial con un gran volumen de texto no etiquetado, permitiendo captar estructuras lingüísticas fundamentales, como la sintaxis o la semántica, para posteriormente, realizar un segundo entrenamiento con conjuntos de datos más pequeños para “afinar” el conocimiento (fine tuning), de manera que se pueden realizar tareas más específicas sin perder calidad. Sin embargo, al depender del pre-entrenamiento, no se puede actualizar dinámicamente con nueva información sin ser reentrenados, además están enfocados en tareas más específicas en las que son entrenadas.
- **Large Language Models (LLM):** Son los modelos más modernos. Se entrenan con corpus de texto con decenas de miles de millones de parámetros para comprender los comandos humanos. Igual que los PLM, pasan por un pre-entrenamiento inicial, para posteriormente ser alineados con los valores humanos en lugar de especializarse en un

dominio específico como los PLMs. Destacan por su adaptabilidad y capacidad de aprovechar el contexto, pero su crecimiento depende del número de parámetros y datos de entrenamiento.

2.1.2. LLM: fundamentos, estado actual y familias

En los últimos años, ha habido diferentes factores que han permitido un rápido avance en los LLM. La disponibilidad de grandes volúmenes de texto de diferentes fuentes ha permitido mejorar su capacidad para generalizar y realizar múltiples tareas. Además, el crecimiento exponencial de la potencia computacional ha permitido entrenar modelos cada vez más grandes, ya que el desarrollo de hardware especializado como GPU ha acelerado su crecimiento (GPT-3 habría tardado 355 años en entrenarse en una sola GPU, pero con 1024Xa100 GPUs solo tardó 34 días) (1). El desarrollo de la arquitectura Transformer superó enfoques anteriores como redes neuronales recurrentes y convolucionales, optimizando la atención, el preentrenamiento y la eficiencia computacional (23).

Tomando como ejemplo a GPT-3, se pueden observar cuatro grandes principios que siguen los LLM para generar sus respuestas:

- **Entrada y codificación:** Reciben secuencias de palabras (tokens), convirtiendo cada palabra a un vector numérico (one-hot) dentro de un vocabulario de términos.
- **Embedding:** Para mejorar la eficiencia, los vectores one-hot se transforman en representaciones más compactas.
- **Codificación posicional:** Al carecer de estructura secuencial, se agregan codificaciones posicionales usando funciones trigonométricas para capturar el orden de las palabras en una oración.
- **Matriz de entrada:** La combinación de embeddings y codificaciones posicionales generará la matriz de entrada, que se procesará en las capas de transformadores.

Los atributos y comportamientos de los LLM están profundamente vinculados con los procesos del entrenamiento, caracterizado por tres etapas principales: preentrenamiento, ajuste fino supervisado (Supervised Fine-Tuning (SFT)) y aprendizaje de refuerzo por feedback humano (Reinforcement Learning from Human Feedback (RLHF)) (27).

- **Pre-entrenamiento:** Permite adquirir conocimientos y habilidades fundamentales, entrenando mediante la predicción autorregresiva de tokens dentro de secuencias de texto. De esta manera, desarrollan una comprensión profunda de la sintaxis y habilidades de razonamiento, estableciendo una base sólida para su posterior ajuste y refinamiento (28).
- **SFT:** Entrena el modelo con un conjunto de datos anotado compuesto por pares de instrucciones y respuestas, mejorando la capacidad de los modelos para seguir instrucciones específicas, ofreciendo respuestas más precisas y útiles. Se ha confirmado la eficacia de este paso para un rendimiento adecuado en tareas no vistas previamente, mostrando capacidad de generalización (29; 30).

- **RLHF:** Se utiliza la evaluación de respuestas por parte de humanos para optimizar el comportamiento del modelo, para maximizar la recompensa determinada, usando algoritmos de aprendizaje por refuerzo. Esto permite que el modelo genere respuestas más alineadas con las expectativas humanas, mejorando la calidad de las respuestas y reduciendo la generación de contenido sesgado (31).

2.1.3. Limitaciones de los LLM

Los modelos de LLM han demostrado ventajas significativas frente a otros modelos de lenguaje. Sin embargo, a pesar de su eficiencia en aplicaciones generales, los modelos disponibles comercialmente presentan limitaciones en el ámbito clínico y médico, generando “alucinaciones”, es decir, respuestas con referencias incompletas o incluso completamente inventadas. Esta falta de precisión compromete la fiabilidad de los resultados, lo que, en un contexto de diagnóstico médico, puede tener consecuencias graves. [Improving Automated Deep Phenotyping Through Large Language] (6).

A pesar de su capacidad de generar texto coherente y realizar tareas complejas, los LLM tienen deficiencias estructurales, como la falta de comprensión real, ya que no disponen de un razonamiento ni conocimiento como un humano, sino que identifican patrones en datos, por lo que pueden llegar a generar información inconsistente.

Sin embargo, se deben reconocer las desventajas que acompañan estos modelos.

- **Sesgo:** Las decisiones pueden presentar sesgos en contra de poblaciones o grupos más marginados, ya que durante el entrenamiento con datos extensos y no estructurados, se absorben representaciones erróneas y comportamientos excluyentes. Esta problemática se mitiga eliminando los datos claramente sesgados del conjunto de entrenamiento, pero no se elimina completamente y reduce la efectividad del modelado de lenguaje (32). El sesgo puede deberse a varios factores, como cuando el modelo genera el siguiente token basándose en sus propias predicciones previas, lo que puede dar lugar a errores acumulativos (sesgo de exposición), o cuando prioriza el conocimiento aprendido durante el entrenamiento sobre la información proporcionada en la entrada (sesgo de conocimiento paramétrico) (3).
- **Seguridad y alucinaciones:** Los LLM tienen diversas aplicaciones en industrias como el ámbito médico, legal o de finanzas, por lo que la veracidad de la información es muy importante. En ocasiones, pueden generar resultados que difieren del contexto proporcionado o del conocimiento factual, denominados *alucinaciones*. (3)
- **Privacidad:** Los modelos de lenguaje a gran escala pueden implicar riesgos para la privacidad, especialmente si se entrenan con datos sensibles o sin un adecuado control de filtrado. La exposición de información personal o confidencial puede representar un problema crítico en su implementación (33).

De hecho, las alucinaciones en los LLM fundacionales como ChatGPT o Gemini pueden mostrar capacidades de razonamiento muy elevadas, por lo que las alucinaciones pueden pasar desapercibidas y dar lugar a resultados falsos o sin fundamento. Esto causa una desconfianza y poca capacidad de usarse en ciertos ámbitos como el legal o médico (34).

2.2. Sistemas RAG en LLM

Tanto los LLM como IA han demostrado un gran potencial para mejorar la eficiencia de la interacción de los médicos y los pacientes con los sistemas de atención médica (35). Aún así, debido a que toda la información procede de los datos proporcionados por el entrenamiento, esta información no se actualiza regularmente, por lo que deben acceder a fuentes externas no verificadas para poder realizar las tareas.

Para abordar esta problemática, se ha propuesto un sistema avanzado de recuperación de información que evita la generación de respuestas sin una base de conocimiento preexistente. Este enfoque, conocido como Retrieval-Augmented Generation (RAG), combina un modelo generativo basado en LLM con un mecanismo de recuperación de información, restringiendo las respuestas a una base de datos específica y enriqueciendo su precisión y fiabilidad.

La generación aumentada por recuperación permite la incorporación de datos personalizados en los LLM, especializando su contenido y reduciendo las alucinaciones (36), sin necesidad de volver a entrenar el modelo de nuevo.

Por lo tanto, la combinación de LLM con RAG presenta una evolución significativa en la generación de texto, permitiendo que los modelos de lenguaje accedan a información externa en tiempo real.

Los modelos RAG en conjunto con LLM se ha incrementado recientemente al demostrarse como, con un correcto pre-procesamiento de los datos han podido alcanzar una precisión cerca del 99.0 % (37).

2.2.1. Arquitectura de RAG. Workflow

La arquitectura Retrieval-Augmented Generation combina la generación de un texto mediante un LLM con la recuperación de información externa para mejorar la calidad y la precisión de las respuestas generadas. Esta arquitectura se basa en dos etapas principales: la búsqueda y recuperación de información relevante en una base de datos (retriever) y posteriormente se emplea la información recuperada para generar una respuesta más precisa y contextualizada, accediendo a información que el LLM no disponía durante su entrenamiento. De este modo, el enfoque RAG permite al modelo LLM una mayor versatilidad y eficiencia, especialmente en tareas que requieren de conocimiento específico y actualizado sin necesidad de un re-entrenamiento. (16)

En general, una arquitectura RAG inicia con la consulta de entrada, la cual pasa al módulo recuperador que accederá al vectorstore para buscar los k documentos más relevantes. Posteriormente, se procesarán junto con la consulta al módulo generador como contexto para que produzca una salida condicionada. La Figura 2.2 muestra el workflow genérico de las arquitecturas RAG.

Pre-procesados: parseo, vectorstore y chunking

Pese a que la arquitectura RAG se fundamenta en dos componentes (recuperación y generación), se debe crear un repositorio de los documentos que el modelo podrá recuperar. Por lo tanto, los documentos requieren de un preprocesado que preparará la información para ser utilizada por los dos componentes.

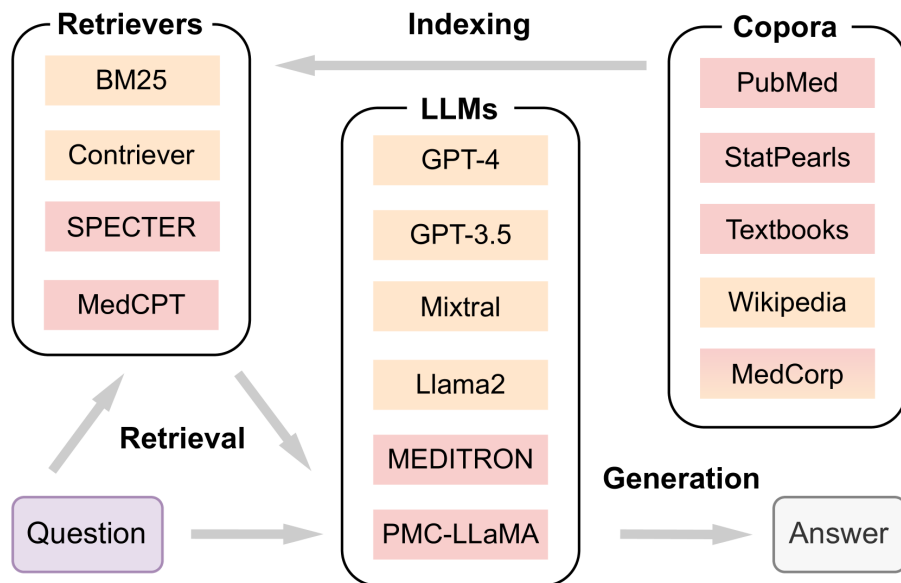


Figura 2.2: Esquema de una arquitectura RAG. Extraído de (2).

El primer paso en este proceso es el análisis de los datos o parseo, que implica extraer y organizar la información de fuentes no estructuradas, como textos, PDFs o bases de datos para convertir los datos en un formato adecuado para su posterior procesamiento. Se ha demostrado que, como mayor ha sido el procesamiento de los datos, mayor fue el rendimiento general del modelo, llegando a alcanzar un 99 % de la precisión después de procesar texto, tablas e imágenes (6).

El siguiente paso será el chunking, en que el texto se dividirá en fragmentos (chunks) más pequeños, que pueden ser desde párrafos completos a oraciones o unidades lingüísticas más simples. El chunking permitirá la recuperación más precisa y específica, ya que el modelo se podrá enfocar en unidades de texto más manejables y relevantes. Además, al gestionar fragmentos más pequeños, se optimiza el modelo en términos de memoria y permita una mejor paralelización de las tareas de procesamiento (38).

Existen múltiples formas de fragmentar los datos, variando en función de diferentes parámetros, como los tamaños de los chunks o los tokens necesarios para procesarlos. Cada método de chunking tiene sus beneficios y sus desventajas, ya que no solo importa la eficiencia del modelo para recuperar la información, sino que la información en sí sea ciertamente relevante para la query lanzada.

Por ejemplo, con el método de chunking de longitud fija se puede determinar un número de caracteres, ya sea especificando el número de caracteres o por tokens, en que cada chunk requerirá un coste determinado de token. Es un método simple y eficiente, pero puede cortar frases o ideas en puntos arbitrarios y que eliminen parte del contexto del chunk.

Por otro lado, se puede procesar el texto para generar chunks basados en estructuras de texto, dividiendo oraciones, párrafos o secciones. La eficacia del chunk se basará en la complejidad del texto fragmentado, ya que cuanto mayor contexto y coherencia tenga el chunk, mayor eficiente será en la recuperación pero, surgen otras desventajas dependiendo de la estructura captada por el chunking. En un mismo texto, el texto no se divide de manera regular, habiendo

secciones u oraciones de diferentes longitudes, o incluso algunos textos no tienen una estructura clara que el chunking pueda detectar.

Una estrategia empleada en conjunto con el chunking, es el over lapping, en que se superponen fragmentos, de manera que, pese a fragmentar la información, no se pierda el contexto semántico (39). Esto permite una técnica de chunking híbrida o adaptativa, de manera que, pese a que un fragmento corte el contexto semántico, la superposición de los chunks adyacentes reduce la pérdida de contexto. Aún así, se debe tener en cuenta que este método duplica el procesamiento y almacenamiento.

Una vez generados, los chunks deben ser almacenados para construir una base de datos accesible para el módulo de recuperación. Los embeddings son representaciones numéricas de fragmentos de texto, transformados en vectores dentro de un espacio multidimensional. Esta transformación captura el significado semántico del texto, lo que permite realizar comparaciones basadas en la similitud entre fragmentos. De este modo, es posible medir la relación entre distintos textos, facilitando la búsqueda y recuperación de información relevante. Es importante tener en cuenta que, cuanto más pequeño sea el chunk, mayor será el peso de cada uno de sus componentes, por lo que la determinación de su tamaño es un parámetro clave en el rendimiento del sistema.

Para almacenar y consultar los embeddings, se utilizan bases de datos vectoriales conocidas como VectorStores. Este método de almacenamiento está optimizado para la búsqueda eficiente, permitiendo recuperar rápidamente los fragmentos más relevantes según una consulta específica. En una arquitectura RAG, el proceso de indexación implica crear un VectorStore a partir del conjunto de chunks transformados en embeddings, junto con la referencia al texto original.

Una de las principales ventajas de los VectorStores es la posibilidad de modificar los documentos sin necesidad de reentrenar el modelo, gracias a funciones específicas que facilitan la actualización tanto de los documentos como de sus embeddings asociados. Además, ofrecen flexibilidad en las operaciones de comparación, como la similitud de coseno, la distancia euclidiana o el producto escalar, adaptándose a distintos formatos y necesidades.

Aunque los VectorStores implementan algoritmos de búsqueda eficientes, muchos de ellos también incluyen opciones de filtrado por metadatos, lo que permite limitar la búsqueda aplicando condiciones específicas. La combinación de estas técnicas enriquece el proceso de recuperación de documentos y, en consecuencia, mejora significativamente la calidad y precisión de las respuestas generadas.

Retrieval

En esta fase, los retrievers buscarán la información más relevante dentro de la base de datos proporcionada en el vectorstore, que contendrá los fragmentos de información y sus embeddings asociados. De este modo, la query realizada por el usuario sirve como input para el modelo de recuperación y buscará la información más relevante.

Cabe destacar que la consulta (query) se somete a un proceso similar al de la indexación de los documentos, vectorizándose usando el mismo modelo de embedding, por lo que el sistema puede calcular una similitud semántica entre la consulta y la información indexada. De este modo, al comparar los embeddings de la consulta con los almacenados, se determinan los fragmentos más relevantes tanto en contexto, significado y similitud de palabras.

La información recuperada será la base para los posteriores procesos, por lo que se deben

tener en cuenta las limitaciones que afecten a la calidad de la respuesta generada, por lo que debemos asegurar que el LLM generador de la respuesta se integre eficientemente con el módulo recuperador.

Durante la configuración del módulo de recuperación, el parámetro k define la cantidad de documentos que se recuperarán en el proceso, y su ajuste tiene un impacto crucial en la calidad de las respuestas generadas. Recuperar demasiados documentos puede introducir información irrelevante, confundiendo al modelo, reduciendo la precisión y aumentando de forma innecesaria la carga computacional. Por el contrario, si se recuperan muy pocos documentos, el modelo podría no contar con el contexto suficiente, lo que aumenta el riesgo de generar respuestas inexactas o alucinadas. Además, una recuperación limitada podría reflejar información sesgada, sin representar adecuadamente la diversidad del conocimiento disponible, dando lugar a respuestas parciales o incompletas.

Augmented

A partir de los fragmentos recuperados, el módulo generador integrará y sintetizará toda la información disponible para enriquecer la consulta que se enviará al LLM. Para ello, se empleará ingeniería de prompts, definiendo parámetros y directrices que guiarán el comportamiento del modelo en la generación de respuestas.

El uso de un módulo de aumento aporta ventajas clave, como el acceso a información en tiempo real, lo que resulta fundamental en campos de rápida evolución, como la investigación médica, mejorando así la precisión y actualidad de las respuestas. Además, permite adaptar el sistema a distintos contextos y dominios sin necesidad de reentrenar el modelo, ya que el conocimiento se recupera dinámicamente en cada consulta (40).

No obstante, la selección de fuentes de información fiables es un aspecto crítico, ya que influye directamente en la calidad de las respuestas generadas. Asimismo, el procesamiento de grandes volúmenes de datos puede incrementar los tiempos de respuesta y exigir altos recursos computacionales, lo que plantea desafíos en términos de eficiencia y escalabilidad.

Generation

Finalmente, el LLM utilizará la información aumentada para generar una respuesta coherente y precisa. El módulo generador será el encargado de producir respuestas a partir de los documentos recuperados, sintetizando el conocimiento relevante y combinándolo con la capacidad generativa del modelo.

Dado que la generación se basa en documentos verificados, el riesgo de alucinaciones se reduce significativamente. Sin embargo, el módulo generador sigue enfrentando desafíos inherentes a la calidad de la información recuperada. La integración de fuentes diversas puede dar lugar a inconsistencias o conflictos de conocimiento, lo que requiere mecanismos adicionales para evaluar la fiabilidad y coherencia de los datos antes de generar una respuesta.

2.2.2. Tipos de RAG en la actualidad

Los LLM están diseñados principalmente para procesar los datos en formato de texto, pero hay modelos similares que trabajan con otros tipos de datos, como audio o video. Los LLM más conocidos comercialmente, como GPT-4 de OpenAI, LLaMA de Meta AI o Claude de

Anthropic son modelos que pueden realizar tareas y generar un texto coherente, trabajando solo con entradas y salidas textuales.

Del mismo modo, los modelos de RAG se basaron inicialmente en estudios de texto, pero han evolucionado para abarcar modalidades como audio, video incluso multimodal, permitiendo aplicaciones como el reconocimiento de voz o análisis de video (41).

Por ejemplo, los modelos de audio extienden los principios al procesamiento del habla, permitiendo la transcripción o los asistentes de voz inteligentes. Los datos se representan mediante embeddings de modelos preentrenados y se emplean tanto para la recuperación de la información como la generación de respuestas contextuales. A inicios de 2025, (42) evaluó audios de enseñanza simulada con un modelo RAG local, obteniendo puntuaciones relativamente altas en precisión (4.13 sobre 5.00) y lenguaje (4.37 sobre 5.00). Este modelo presentó limitaciones técnicas derivadas del tipo de dato, ya que se detectaron faltas de precisión en el reconocimiento multilingüe.

Por otro lado, también existen modelos RAG que basan su recuperación en datos extraídos de imágenes, combinando transformadores de visión con la generación aumentada para mejorar la generación de respuestas a partir de imágenes, como se comprobó en el estudio de (43), en el que se emplearon Vision Transformers y variantes para procesar imágenes médicas y extraer información visual relevante, empleando GPT-2 como decodificador. Este estudio superó los modelos recurrentes en la generación de informes médicos, demostrando el potencial de los sistemas RAG en la mejora de eficiencia y precisión en el diagnóstico.

Unificando ambos tipos de datos, se puede asumir que la aplicación de RAG en video presenta un desafío significativo debido a la naturaleza multimodal del contenido audiovisual, necesitando de recuperación eficiente de fragmentos de vídeo, pero también generando respuestas basadas en los metadatos de los mismos. El estudio de (44) permite identificar las limitaciones más relevantes de un sistema RAG multimodal, siendo la más relevante que no existen métricas estandarizadas que midan la capacidad de los LLM para seleccionar segmentos de video adecuados a partir de descripciones textuales.

El estudio de (6) demostró que la calidad de los datos disponibles del sistema, mayor rendimiento muestra. Concretamente, el estudio de (6) realizó una transformación progresiva de los datos, desde una simple limpieza inicial de los datos en formato textual, a la conversión de imágenes a texto y tablas en formato tabular, obteniendo una precisión significativamente superior cuanto mayor calidad mostraba la conversión,

Se puede observar, entonces, que los modelos RAG están evolucionando para convertirse en herramientas multimodales capaces de manejar todo tipo de datos. Aunque existen desafíos técnicos y metodológicos, los avances en este campo prometen mejorar significativamente sectores como el diagnóstico médico.

2.2.3. Retos y límites de RAG, direcciones de futuro

La capacidad para combinar la recuperación de información y la generación de texto ha permitido avances significativos en tareas complejas, como la generación de informes a partir de imágenes, la creación de resúmenes o la respuesta de preguntas. Sin embargo, pese a su creciente evolución y aplicabilidad, los sistemas RAG enfrentan varios retos y limitaciones técnicas, quedando potencial por explotar.

En los sistemas RAG, las respuestas incompletas o erróneas pueden deberse a diversos fac-

tores dentro de su arquitectura, y no únicamente a la falta de contenido, lo que puede resultar en respuestas sin referencias sólidas que las respalden. En la Figura 2.3 se observan los posibles fallos en la recuperación de documentos. Uno de los fallos más comunes ocurre cuando el sistema no logra identificar información relevante en los documentos recuperados, a pesar de que esta esté presente, debido a errores en la clasificación que impiden destacar dicha información como significativa. Otro posible problema surge cuando, aun habiendo detectado la documentación adecuada, el modelo de lenguaje (LLM) no la incorpora correctamente en la respuesta, afectando negativamente la calidad de la información proporcionada. En situaciones donde se manejan numerosos documentos con contenido similar, la presencia de ruido en la información recuperada puede llevar a respuestas imprecisas o confusas. Asimismo, cuando la pregunta es demasiado general, el modelo puede generar respuestas que no abordan adecuadamente la necesidad de la consulta. Finalmente, también es posible que se generen respuestas incompletas que, aunque no sean incorrectas, omitan detalles relevantes presentes en el contexto, reduciendo así la utilidad de la información ofrecida (32).

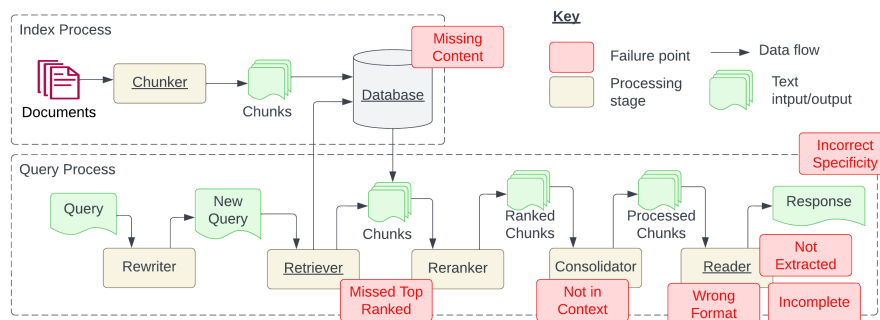


Figura 2.3: Workflow de arquitectura RAG. Extraído de (3)

A partir de estos errores, se derivan una serie de desafíos técnicos que impactan directamente en el rendimiento y la fiabilidad de los sistemas RAG:

- **Dependencia de la calidad de los documentos recuperados:** La precisión de los modelos RAG se basa, principalmente, en la calidad de los documentos recuperados, ya que en caso de incluir datos inexactos o incorrectos se puede producir una recuperación sesgada o "lucinaciones", en que las respuestas generadas por el modelo se basan en referencias incompletas o completamente inventadas pero tomadas como reales. Esto compromete la fiabilidad de los resultados, influenciando a su aplicación práctica en ámbitos en que la precisión es fundamental (16). Del mismo modo, los modelos pueden "heredar" sesgos presentes en los documentos recuperados, siendo uno de los desafíos más críticos, ya que pese a que se han propuesto diferentes técnicas, como la recuperación justa (), la completa eliminación del sesgo sigue siendo una tarea complicada.
- **Alta demanda computacional:** Uno de los principales retos de los modelos RAG radica en su alta demanda computacional, ya que requieren de dos módulos, la recuperación de información y la generación del texto. Solamente la latencia en la recuperación de documentos relevantes puede ser altamente costosa, afectando a la eficiencia del sistema si se manejan grandes cantidades de datos. Además, la creación de un repositorio de

los vectores, los embeddings y la generación de respuesta impactan directamente en la capacidad del modelo RAG y la posibilidad de aplicarlo al mundo real.

- **Seguridad del sistema:** Las arquitecturas RAG presentan ciertas vulnerabilidades de seguridad, como los ataques de recuperación y los ataques generativos con el objetivo de manipular la información recuperada o la respuesta generada (45). Aun así, estos ataques requieren de condiciones específicas que pueden evitarse, como el acceso único a bases de datos locales (46).

Capítulo 3

Marco teórico

3.1. Pre-procesamiento

3.1.1. Introducción

Para el eficiente procesamiento de datos en la recuperación de información, las arquitecturas RAG requieren de un preprocesamiento, que convertirá la información desde un formato no estructurado, como es el formato PDF, a un formato estructurado y accesible para su análisis computacional. Este proceso implicará la extracción de texto desde los archivos PDF de las guías clínicas reumatológicas, la normalización y la limpieza de los datos, así como la transformación de la información textual en vectores numéricos a través de técnicas de embedding y su almacenamiento en un vectorstore. Este procesamiento permitirá gestionar grandes volúmenes de información, facilitando la búsqueda, comparación y análisis semántico para la recuperación de documentos relevantes.

3.1.2. Framework

Durante el desarrollo de sistemas de RAG, el uso de frameworks especializados facilita la integración entre los modelos de lenguaje y la bases de datos vectoriales, optimizando la eficiencia y precisión en la recuperación y generación de respuestas. Actualmente existen múltiples herramientas especializadas, destacando LangChain, LlamaIndex y Haystack, cada una con sus características. Estos frameworks proporcionan una infraestructura que permite la implementación de técnicas avanzadas como la búsqueda semántica, re-ranking y generación de texto en base a un contexto, imprescindibles en un modelo de RAG.

La elección de uno de estos frameworks dependerá de sus características como la escalabilidad, compatibilidad con los LLM y la flexibilidad con la personalización del workflow diseñado.

- **LangChain (47):** Está diseñado para facilitar la integración de LLM con fuentes de datos externas, bases de datos vectoriales y herramientas adicionales. Permite personalizar cada componente del pipeline, además de herramientas nativas como la memoria conversacional, haciéndola una gran opción en modelos de RAG. Su estructura básica se centra en los retrievers (módulo de recuperación basada en embeddings), chains (flujos de trabajo), memoria y agentes (que ejecutan workflows dinámicos con toma de decisiones).

- **LlamaIndex (48)**: Está especializado en la indexación y recuperación eficiente de información desde documentos estructurados (y semiestructurados), optimizando la integración con modelos de LLM para generar respuestas basadas en información. Es compatible con múltiples LLM y fuentes de datos. Su arquitectura se basa en cargadores de documentos (Document Loaders), Indexadores (estructurando la información de bases vectoriales) y Query Engine (el motor de búsqueda semántica).
- **Haystack (49)**: Es un framework de código abierto que permite crear aplicaciones LLM y canales generativos con recuperación mejorada y sistema de búsqueda en grandes colecciones de documentos. Su estructura modular permite combinar diferentes motores de búsqueda, con pipelines híbridos e integración con bases de datos vectoriales. Su arquitectura se basa en Document Stores (almacenajes basados en SQL-NoSQL), Retrievers (módulo de recuperación), Readers (Modelo de NLP para la extracción) y el Generator (Modelo de LLM para la generación de respuesta).

3.1.3. Parseo de Guías Clínicas

Debido al formato de las guías clínicas (PDF), el parseo es una etapa crítica en la construcción de arquitecturas RAG. Estos documentos requieren de una correcta extracción, limpieza y segmentación para garantizar una recuperación efectiva de información.

Existen múltiples herramientas que permiten extraer el contenido de documentos en formatos diferentes, como Extensible Markup Language (XML) o Markdown. El formato PDF en específico es complejo de extraer, ya que carece de una estructura semántica clara como otros formatos como XML o HyperText Markup Language (HTML), que definen los datos con una estructura jerárquica personalizada.

Los métodos de parseo se pueden clasificar en dos enfoques principales:

El primer enfoque sigue un conjunto de reglas predefinidas para extraer la información de los documentos, basándose en patrones o estructuras bien definidas dentro del documento. De esta manera, emplea expresiones regulares o delimitadores ya presentes en los documentos, siendo altamente preciso en documentos con formatos uniformes y bien estructurados, como XML. Este enfoque es rápido y eficiente, ya que no requiere de entrenamiento de modelos de Machine Learning (ML), sin embargo, son dependientes de etiquetas de estructura bien definidas, lo que los hacen menos flexible ante variaciones de formato. Un ejemplo de este enfoque es PDFminer, que extrae texto de archivos PDF.

El segundo enfoque, emplea modelos de Natural Language Processing (NLP) y ML para interpretar y estructurar el contenido del documento sin seguir las reglas rígidamente. Esto permite manejar documentos con estructuras complejas, como los artículos científicos (con más de una columna, gráficos y tablas), de manera que extrae no solo el texto, sino que emplea técnicas avanzadas para extraer las tablas y las referencias del texto. Al emplear métodos más avanzados, regularmente son más costosos computacionalmente, como GeneRation Of Bibliographic Data (GROBID), que estructura artículos científicos en un formato

Actualmente existen múltiples herramientas comerciales disponibles, siendo capaces de extraer la información de los documentos PDF en diferentes formatos (50). Las herramientas más populares son las siguientes:

- **PyMuPDF**: Se trata de una librería de Python ligera y rápida que permite la extracción

de texto y estructuración de documentos. Permite navegar y extraer elementos específicos de las páginas del documento a través de las diferentes clases y métodos que ofrece el paquete. Los formatos de salida pueden ser desde texto plano sin estructura (TXT) hasta HTML o XML con información jerárquica.

- **Pdfplumber:** Es una librería de Python diseñada para la extracción avanzada de datos desde archivos PDF. Se especializa en la extracción de texto, tablas e imágenes enfocado en la precisión y la estructuración. Es especialmente práctico en documentos complejos, con múltiples columnas, tablas embebidas o textos con formatos irregulares.
- **GROBID:** Es una herramienta de código abierto diseñada para la extracción y estructuración automática de información en documentos científicos. A diferencia de los modelos anteriores, emplea técnicas de NLP y ML para transformar artículos académicos en formatos estructurados como XML y Text Encoding Initiative (TEI). Además del texto, también permite la extracción de metadatos, como los títulos, referencias o palabras clave.
- **LlamaParse:** Emplea una combinación de IA, visión computacional y técnicas de NLP para extraer información de documentos complejos. Se destaca por su capacidad de manejar documentos con objetos embebidos como tablas, proporcionando una representación en formato Markdown.

Pese a que hay herramientas significativamente eficaces en el parseo, todas presentan ciertas limitaciones que imposibilitan una única selección con eficacia absoluta. Si, además, añadimos un conjunto de documentos que no tienen la misma estructura en todos los documentos, la tarea de extracción puede hacerse aún más compleja con solo una herramienta de parseo. Por lo tanto, para optimizar dicha tarea, se debe realizar una combinación de dos herramientas que se complementen en sus características, como GROBID, que permite identificar entre columnas, pero dificulta la extracción de tablas, y LlamaParse, que permite la extracción de tablas pero tiene dificultades con la diferenciación de columnas.

3.1.4. Chunking, la fragmentación de documentos

Se ha demostrado que los sistemas RAG que emplean LLM suelen generar respuestas inexactas al recuperar todo un documento, ya que se incorpora información innecesaria que introduce ruido en el sistema (4). Esto se debe a que la información innecesaria y excesiva distorsiona el resultado de dichos modelos, reduciendo la confiabilidad del sistema, sobre todo en tareas críticas como el asesoramiento clínico. En la Figura 3.1 se puede observar el rendimiento de dos modelos, detallando el efecto del chunking, demostrando la mejora del rendimiento de la recuperación y de la generación de la respuesta gracias a la fragmentación.

De hecho, los modelos RAG basados en fragmentos (chunks) como ChunkRAG han demostrado una eficiencia mucho más elevada respecto a los modelos basados en recuperación de documentos, sobre todo en documentos largos y complejos (4; 51). Los modelos basados en chunks realizan una jerarquización de los fragmentos, de manera que puede considerar aquellos más relevantes para recuperar, dando lugar a un incremento de la precisión general del modelo.

Además de la jerarquización de los fragmentos, el sistema también será capaz de comparar fragmentos entre ellos por similitud, filtrando los resultados para evitar información repetida

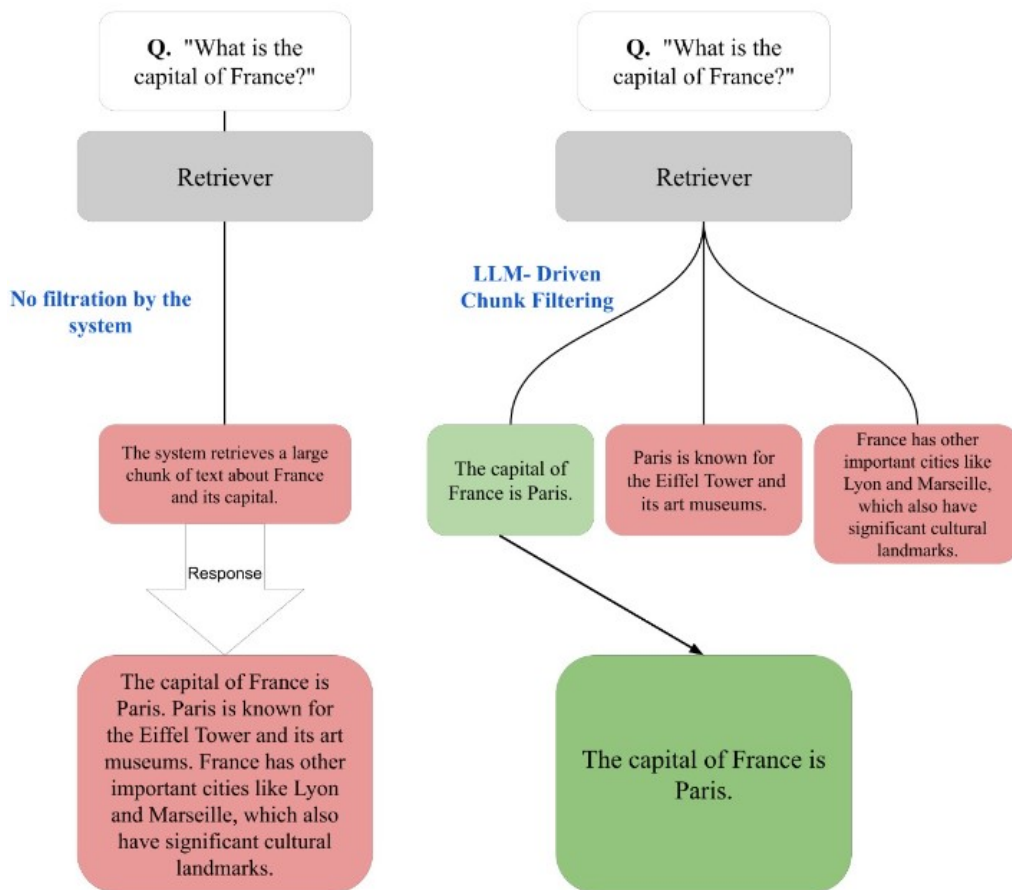


Figura 3.1: Comparación del efecto del chunking. Extraído de (4)

o redundante que dificultan la capacidad del modelo para generar repuestas coherentes y únicas. Existen múltiples enfoques que analizan la similitud entre fragmentos, como el cálculo de similitud de coseno, que calcula el ángulo entre dos vectores distintos de cero en un espacio de producto interno, cuantificando la similitud entre dos vectores independientemente de su magnitud. (52)

También existen modelos que, para optimizar la recuperación de fragmentos, también emplean una búsqueda en base al caché. De este modo, se llegó a reducir el cálculo redundante en un 51% en cargas de trabajo de producción reales (53). Esto refleja la importancia del almacenamiento de memoria, pero también implica una limitación en el almacenamiento de caché.

Se debe tener en cuenta que optimizar el tamaño de los fragmentos es esencial para la precisión y la recuperación eficiente, sin embargo, determinar el tamaño adecuado depende de la estructura de los datos y la densidad de información e incluso, en la misma base se pueden encontrar diferentes grados de granularidad (39). De hecho, otros factores como la misma pregunta pueden afectar al tamaño del chunk óptima, ya que en función de la granularidad de la query, el tamaño del chunk puede variar (si se solicita una información más amplia, se prefiere una granularidad más gruesa). En conclusión, se debe encontrar un equilibrio en la granularidad,

ya que una granularidad gruesa proporciona más información con menor precisión, mientras que la fina ofrece información completa a costa de la eficiencia.

Actualmente, se disponen de múltiples herramientas para la fragmentación de información. Cada framework (Langchain, LlamaIndex...) tienen sus propios splitters nativos, pero actúan de manera similar. (54)

- **TokenTextSplitter:** Divide el texto en fragmentos de tamaño especificado según los recuentos de tokens sin procesar, permitiendo definir una ventana de solapamiento (*overlap*) para respetar el contexto.
- **SpacyTextSplitter:** Analiza el texto con preferencia por oraciones completas, intentando mantener unidas las oraciones y los párrafos. Esto reduce la probabilidad de obtener oraciones inconexas al final del fragmento.
- **RecursiveCharacterTextSplitter:** Divide el texto de manera recursiva hasta obtener fragmentos del tamaño adecuado, procurando mantener párrafos completos el mayor tiempo posible para preservar la relación semántica. Además, permite establecer limitadores, lo que incrementa su versatilidad.

En modelos RAG, la precisión de la recuperación depende en gran medida de los fragmentos de información. Por ello, es importante no solo el tamaño del chunk, si no que también es relevante que no se pierda el contexto dentro del mismo fragmento, por lo que **RecursiveCharacterTextSplitter** destaca entre el resto de técnicas.

3.1.5. Representación vectorial: embeddings y vectorstores

Vectorización: embedding

Una de las desventajas de los LLM es que al añadir más contexto de entrada se produce un cuello de botella en la generación de la inferencia, ya que la ralentiza la generación además de requerir más memoria. Para evitar estas limitaciones, se han desarrollado técnicas de compresión de contexto, reduciendo la cantidad de información que el modelo necesita procesar, siendo posible emplear dos técnicas: la filtración de palabras irrelevantes del texto antes de pasarlo al modelo de LLM, y usando embeddings, que convierten el contexto en una representación más compacta, pero manteniendo gran parte del significado.

Los inicios de los modelos de embedding se remontan a los enfoques basados en frecuencia que permitían calcular la relevancia de términos en documentos, pero sin capturar las relaciones semánticas entre las palabras del contexto. Posteriormente, se introdujo la idea de representar palabras como vectores en un espacio continuo, de manera que términos con significado similar mantuvieran la similitud en las representaciones mediante distancia. La revolución llegó cuando la arquitectura Transformer permitió crear modelos como BERT y GPT, que permitieron generar embeddings dinámicos, adaptándose al contexto de cada palabra en una oración, permitiendo su incorporación en tareas de recuperación de información y en el desarrollo de arquitecturas RAG.

Una incrustación (embedding) es una representación densa de objetos en un espacio continuo de menor dimensión, capturando similitudes semánticas al asignar elementos similares vectores cercanos en el espacio de la representación. De esta manera, pueden representar palabras o

frases reduciendo la dimensionalidad del problema y mejorando la eficiencia computacional en comparación con enfoques más ingenuos (55).

Los enfoques de compresión basados en embeddings se centran en comprimir el contexto en incrustaciones (embeddings) que el modelo decodificador del LLM puede interpretar directamente. Esta técnica permite comprimir la información contextual segmentándola en fragmentos empleando herramientas de clustering, autoencoders o mean pooling para reducir dimensionalidad, que posteriormente se agregan en incrustaciones de resumen mediante un modelo iterativo hasta alcanzar el tamaño de incrustación objetivo.

De este modo, palabras estructuralmente diferentes como “perro” y “cachorro” serían palabras cercanas por su semántica.

Modelos como OpenAI Embedding convierten fragmentos de texto en vectores densos que pueden ser empleados para buscar y recuperar información relevante en bases de datos vectoriales. Esto permite mejorar la recuperación de documentos relevantes, de manera que el modelo recibe el contexto más relevante para generar respuestas informadas.

Los modelos de embeddings, pese a su efectividad, a su vez, tienen limitaciones (56):

- **Dependencia de modelos de compresión grandes:** Estos métodos se basan en modelos de compresión grandes para lograr una efectividad, incrementando los costos computacionales y dificultando su implementación en entornos con recursos limitados.
- **Baja efectividad en la generación de respuestas:** Actualmente no aprovechan completamente el potencial de los LLM, ya que optimizan solo ciertos componentes del sistema, dejando el decodificador del modelo sin modificaciones.
- **Tasa de compresión fija:** Los métodos actuales no ofrecen diferentes tasas de compresión con respecto a la longitud del contexto de entrada, impidiendo optimizar el equilibrio entre velocidad de inferencia y la calidad de generación.
- **Limitación a un solo documento:** Los métodos efectivos actuales admiten en su mayoría la generación de respuestas a partir de un único documento, restringiendo la utilidad en tareas de análisis e integración de información de múltiples fuentes, como la generación de respuestas complejas basadas en diversos documentos.

Durante el modelo, la información que proporcionará el contexto se convirtió en vectores incrustados en una base de datos vectorial, conteniendo la información contextual codificada. Del mismo modo, cuando el usuario ingresa la pregunta al modelo, se realiza un embedding de la misma pregunta empleando el mismo modelo, realizando comparaciones entre los vectores de la base de datos y obteniendo aquellos con menor distancia. Esta distancia, codificada como contexto, implica que aquellos que se obtengan con menor distancia serán, a su vez, los más similares a nivel contexto.

Los documentos recuperados se utilizan como contexto para generar una respuesta, permitiendo al modelo generar respuestas más precisas e informativas al basar sus resultados en datos reales.

Actualmente, existen múltiples herramientas para realizar el proceso de embedding. La plataforma HuggingFace (<https://huggingface.co/spaces/mteb/leaderboard>) proporciona una tabla de clasificación en Massive Text Embedding Benchmark (MTEB) Leaderboard, donde se comparan diferentes modelos de embedding en tareas del MTEB.

Esta clasificación ayuda a evaluar el rendimiento de los diferentes modelos en diferentes escenarios, permitiendo seleccionar el modelo más adecuado en función de las necesidades. La tabla proporciona las características y métricas evaluadas en diferentes contextos. En el contexto de RAG y la recuperación de información, las métricas más destacables serían:

- **Retrieval (Recuperación):** Evalúa qué tan bien el embedding recupera información relevante.
- **Semantic Textual Similarity (STS):** Mide qué tan bien el embedding entiende relaciones entre textos similares.
- **Re-ranking:** Ayuda a mejorar los documentos recuperados en RAG, priorizando ciertos documentos.
- **Bitext Mining:** Capacidad de identificar pares de textos equivalentes en distintos idiomas (para RAG multilingüe).

Se extrajeron las 4 mejores opciones en base a estos parámetros, teniendo en cuenta su compatibilidad con un buen rendimiento en RAG con LangChain. De este modo, se plantearon estas opciones:

- **Alibaba NLP (gte-Qwen2-7B-instruct):** Tiene una gran recuperación y STS, además de que es open-source (gratuito). Soporta preguntas complejas en RAG, funcionando bien con retrieval y generación de contexto relevante. No tiene integración directa con LangChain.
- **Intfloat (e5-mistral-7b-instruct):** Tiene un buen equilibrio entre Retrieval y re-ranking, y está optimizado para tareas de NLP, pero tiene un consumo mayor que otros modelos, además de una integración menos directa con LangChain.
- **Gemini (gemini-embedding-exp-03-07):** Se destaca tanto en re-ranking, retrieval y STS, optimizado para comprensión semántica profunda y tareas de recuperación de información. Tiene acceso restringido y falta de integración optimizada para LangChain.
- **OpenAI (text-embedding-3-large):** Tiene una buena optimización para retrieval, STS y re-ranking. Es totalmente compatible con LangChain e incluso con el vectorstore de ChromaDB. Tiene una eficiencia y costo superior a otros modelos, sin requerir de una infraestructura propia.

Entre las opciones disponibles, OpenAI destaca entre ellas al disponer de integración nativa con LangChain y ChromaDB, simplificando su implementación. Además ofrece un rendimiento óptimo en los parámetros de retrieval, STS y re-ranking, garantizando una recuperación de información eficiente. También evita la necesidad de infraestructura propia para la inferencia, optimizando tanto en costos como eficiencia. Finalmente, su escalabilidad y persistencia asegura que el sistema, en caso de ser necesario, pueda crecer con el tiempo.

El resto de alternativas, pese a tener ventajas en ciertos aspectos, presentan también limitaciones clave en integración, costos o flexibilidad.

Base de datos vectoriales: vectorstores

Una base de datos vectorial (vectorstore) se utiliza para almacenar datos de alta dimensión que no han podido ser caracterizados por sistemas de gestión de bases de datos tradicionales. Estos vectores pueden tener una cantidad variable de dimensiones, dependiendo de la granularidad de los datos, y representan matemáticamente las características o atributos de los fragmentos de texto (57).

Estas bases de datos especializadas presentan ciertas ventajas respecto a las bases de datos tradicionales:

- **Búsqueda y recuperación por similitud:** Pueden encontrar los datos más relevantes según distancia en el espacio vectorial, siendo especialmente práctica en aplicaciones de NLP, a diferencia de las bases de datos tradicionales, que no pueden captar el significado contextual o semántico en el texto.
- **Soporte para datos complejos y no estructurados:** Permiten almacenar y buscar datos con alta complejidad y granularidad, sin necesidad de estar estructurados como en las bases de datos tradicionales.
- **Escalabilidad y rendimiento:** Pueden manejar análisis y procesamiento de datos a gran escala y en tiempo real, esencial para aplicaciones de IA. Pueden distribuir la carga de trabajo y reducir la latencia en consultas a gran escala mediante técnicas avanzadas de indexación y paralelización.

Actualmente, los tres modelos de VectorStore más populares son ChromaDB, MongoDB (Atlas Vector Search) y Facebook AI Similarity Search (FAISS), cada una con sus características y limitaciones (58; 59).

- **MongoDB** soporta búsquedas vectoriales dentro de una base de datos documental, y permite combinar consultas estructuradas y vectoriales (60).
- **FAISS** es una librería optimizada para búsqueda rápida en colecciones de vectores, soportando una indexación eficiente y búsqueda aproximada (lo que reduce la latencia en consultas grandes) (61).
- **ChromaDB** es una base de datos optimizada para búsquedas por similitud, permitiendo almacenar y consultar metadatos adicionales. Tiene una integración nativa con el framework de LangChain, y se puede emplear en entornos locales y escalables en la nube (62).

Para determinar la base de datos vectorial adecuada, se debe tener en cuenta ciertos aspectos clave: la integración con el framework utilizado (LangChain), la posibilidad de almacenar metadatos y la facilidad para implementar el modelo y escalarlo.

De este modo, tras seleccionar las opciones se determinó que ChromaDB puede ofrecer el mejor equilibrio entre estas características.

Respecto a la integración con LangChain, MongoDB no tiene una integración nativa con el framework (a diferencia de ChromaDB), así como FAISS, que pese a ser eficiente en búsquedas vectoriales, tampoco tiene un soporte nativo.

En cuestión de soporte para el almacenamiento y metadatos, MongoDB puede almacenar los metadatos en documentos JavaScript Object Notation (JSON) dentro de una base de datos documental, pero requiere de indexación específica para la búsqueda por similitud (algo que ChromaDB hace de forma automática). FAISS, por otro lado, no puede almacenar los metadatos dentro de la base de datos, requiriendo un almacenaje externo.

En términos de facilidad de implementación, MongoDB requiere el uso de Atlas Vector Search, una funcionalidad de pago, requiriendo de una implementación compleja de comprender. FAISS, al ser una librería optimizada en búsquedas, no es una base de datos persistente, por lo que para mantener los datos a largo plazo requiere de un almacenamiento externo.

Los vectores incrustados, en un modelo RAG, se recuperarán mediante similitud con la pregunta del usuario. De este modo, uno de los factores clave en la selección de la base de datos vectorial será su optimización para búsquedas por similitud. Chroma, como se ha mencionado anteriormente, está diseñado específicamente para este propósito. MongoDB, pese a soportar estas búsquedas, su rendimiento es inferior ya que está diseñado como base de datos documental. FAISS, aunque es altamente eficiente en búsquedas vectoriales y está optimizado para grandes volúmenes de datos, carece de almacenamiento persistente.

Finalmente, debido a que las guías clínicas que se emplearán en este modelo pueden aumentar con el tiempo (con nuevas publicaciones, por ejemplo), el modelo seleccionado debe tener una escalabilidad y flexibilidad adecuada. MongoDB tiene una alta escalabilidad, pero depende del uso de Atlas Vector Search (de pago), limitando su uso en grandes volúmenes de datos. FAISS, pese a ser eficiente en búsquedas a gran escala, no está diseñado para ser una solución completa. ChromaDB, en cambio, se puede ejecutar tanto localmente como en la nube, con opciones de persistencia que permite escalar en función de las necesidades del modelo.

(¿tabla resumen?)

Tras evaluar las opciones y las limitaciones de cada una, ChromaDB es la mejor alternativa al ofrecer integración nativa con LangChain, almacenamiento de metadatos sin necesidad de configuraciones adicionales, permite persistencia de datos y tiene una implementación sencilla y sin costos adicionales.

3.1.6. RAG: Retrievers y Generators

(63)

Capítulo 4

Materiales y métodos

En esta sección se detallan los procedimientos y herramientas utilizados para desarrollar el modelo de RAG aplicado a guías clínicas. Se describen los pasos empleados en la adquisición y preprocesamiento de los documentos, las técnicas de fragmentación y la conversión de los archivos en formatos estructurados. Además, se especifican los algoritmos y modelos empleados para la recuperación de la información, junto a las métricas utilizadas para evaluar su rendimiento.

El proceso metodológico se ha diseñado con el objetivo de garantizar la eficiencia y precisión en la extracción y recuperación de contenidos relevantes dentro de las guías clínicas reumatológicas.

4.1. Materiales

En esta sección se presentan los datos y herramientas empleadas en el desarrollo del modelo. Se describe el conjunto de guías clínicas utilizadas, especificando su formato, origen y características, así como las herramientas de software y librerías utilizadas para el procesamiento de los documentos y la implementación del modelo.

4.1.1. Conjuntos de datos

Para la realización de este trabajo, se ha utilizado un conjunto de datos de documentos de guías clínicas en reumatología obtenidos de la ACR sobre patologías reumatológicas. Estos documentos engloban 17 patologías reumatológicas, como artritis reumatoide o lupus, presentando una gran variedad de recomendaciones repartidas en 23 documentos. Se encuentran en documentos PDF, cada una con una configuración diferente (diferente número de columnas, diferentes formatos de estructuración...).

Cada patología cuenta con un documento de recomendaciones clínicas detalladas, respaldadas por evidencia científica, junto con tablas y figuras que permiten entenderlas, en diferentes grados de complejidad.

Dado a que estas guías no siguen un formato homogéneo, requerirán de un proceso de preprocesamiento inicial previo a la incorporación en el modelo RAG, implicando una extracción, una limpieza y normalización de la información. Se puede observar en tabla siguiente la información sobre las guías clínicas empleadas en el trabajo.

****AQUI VA LA TABLA QUE NO SE COMO PONER****

4.1.2. Herramientas y tecnologías

El desarrollo de este trabajo ha requerido de diversas herramientas y tecnologías para la implementación, procesamiento y almacenamiento de datos.

Respecto al lenguaje de programación y bibliotecas, se ha empleado el lenguaje de programación **Python** debido a su versatilidad y amplio abanico de bibliotecas especializadas, como pandas para la manipulación de datos o langchain para la integración del modelo con la arquitectura RAG.

Se ha empleado el modelo de lenguaje [**GPT-4o mini**] de OpenAI como base para la recuperación aumentada de información. Se ha seleccionado este modelo por su capacidad para procesar grandes volúmenes de texto y generar respuestas coherentes en contexto de recuperación de contexto clínico, considerando también su bajo valor económico.

Para almacenar y gestionar la información extraída de los documentos se ha empleado **ChromaDB**, elegida por su eficiencia demostrada en la gestión de bases de datos vectoriales, facilitando la búsqueda semántica y la integración con modelos de lenguaje.

Para el diseño de arquitecturas RAG existen varios frameworks especializados. Entre LangChain, LlamaIndex y Haystack, todos tienen características distintas tanto en la integración de modelos de LLM, la estrategia de recuperación que siguen y el grado de modularidad. Para este trabajo, se ha seleccionado LangChain por su flexibilidad, además de la integración nativa con ChromaDB como base vectorial y OpenAI GPT como LLM y su capacidad para gestionar interacciones complejas en sistemas de recuperación de información.

Los otros dos frameworks, pese a ser buenas opciones, fueron descartadas por diferentes motivos. LlamaIndex fue descartado ya que se centra en la integración de múltiples componentes en un workflow modular, pero su enfoque en la optimización de la indexación no es el adecuado para un modelo de RAG. Respecto a Haystack, se descartó porque su enfoque en pipelines híbridos y NLP tradicionales no es necesario para el modelo RAG, basado en embeddings y LLMs, donde LangChain permite una integración más directa.

4.2. Métodos

En esta sección se detallan los procedimientos de cada etapa del desarrollo del modelo, describiendo el flujo de trabajo aplicado en las etapas de preprocesamiento y la posterior implementación del modelo de recuperación de información y generación de respuesta. Asimismo, se explicarán los criterios de evaluación empleados para analizar el rendimiento del sistema.

4.2.1. Preprocesamiento de los documentos

4.2.2. Representación y almacenamiento de la información

4.2.3. Recuperación de información

4.2.4. Generación de respuestas

4.2.5. Evaluación de sistema

La evaluación de una arquitectura RAG es bastante compleja, ya que interfieren múltiples componentes independientes, cada uno con métricas y limitaciones propias. Por ello, se establecerán parámetros de evaluación de las métricas.

Interacción Retriever-Generator

El rendimiento del sistema depende, en mayor medida, de la calidad de la recuperación y de la generación del LLM, ya que si el retriever devuelve documentos irrelevantes, el LLM generará respuestas incorrectas o incompletas, mientras que si la generación del LLM no sigue el contexto proporcionado por el retriever, la recuperación puede ser inútil.

Variabilidad de la respuesta

Estudio de las alucinaciones

Efecto del re-ranker

Capítol 5

Valoració econòmica

Consideración previa:

El uso de Application Programming Interface (API)s de LLM como OpenAI requieren del uso de tokens, por lo que tendrán un coste asociado. El coste depende del modelo de LLM empleado en el modelo. Se barajaron las opciones de GPT-3.5 turbo y GPT-4o mini, buscando la optimización del

Capítulo 6

Referencias

Bibliografia

- [1] Wang Z, Chu Z, Doan TV, Ni S, Yang M, Zhang W. History, Development, and Principles of Large Language Models-An Introductory Survey; 2024. Available from: <https://arxiv.org/abs/2402.06853>.
- [2] Xiong G, Jin Q, Lu Z, Zhang A. Benchmarking Retrieval-Augmented Generation for Medicine; 2024. Available from: <https://arxiv.org/abs/2402.13178>.
- [3] Ji Z, Lee N, Frieske R, Yu T, Su D, Xu Y, et al. Survey of Hallucination in Natural Language Generation. *ACM Computing Surveys*. 2023 Mar;55(12):1–38. Available from: <http://dx.doi.org/10.1145/3571730>.
- [4] Singh IS, Aggarwal R, Allahverdiyev I, Taha M, Akalin A, Zhu K, et al.. ChunkRAG: Novel LLM-Chunk Filtering Method for RAG Systems; 2024. Available from: <https://arxiv.org/abs/2410.19572>.
- [5] Garcia BT, Westerfield L, Yelemali P, Gogate N, Rivera-Munoz EA, Du H, et al. Improving Automated Deep Phenotyping Through Large Language Models Using Retrieval Augmented Generation. *medRxiv*. 2024 Dec. Preprint. Available from: <https://doi.org/10.1101/2024.12.01.24318253>.
- [6] Kresevic S, Giuffre M, Ajcevic M, Accardo A, Croce LS, Shung DL. Optimization of hepatological clinical guidelines interpretation by large language models: a retrieval augmented generation-based framework. *NPJ Digital Medicine*. 2024 April 23;7(1):102. Available from: <https://www.nature.com/articles/s41746-024-01091-y>.
- [7] Murugan M, Yuan B, Venner E, Ballantyne CM, Robinson KM, Coons JC, et al. Empowering personalized pharmacogenomics with generative AI solutions. *Journal of the American Medical Informatics Association*. 2024 May 20;31(6):1356-66. Available from: <https://doi.org/10.1093/jamia/ocae039>.
- [8] Delanoe P, Tchuenté D, Colin G. Method and evaluations of the effective gain of artificial intelligence models for reducing CO2 emissions. *Journal of Environmental Management*. 2023;331:117261. Available from: <https://www.sciencedirect.com/science/article/pii/S030147972300049X>.
- [9] Yokoi T. GenAI: Ventaja competitiva versus coste medioambiental; 2024. Accessed: 2025-02-21. Available from: <https://www.imd.org/ibyimd/artificial-intelligence/genai-competitive-advantage-versus-environmental-cost/>.

- [10] Li P, Yang J, Islam MA, Ren S. Making AI Less "Thirsty": Uncovering and Addressing the Secret Water Footprint of AI Models; 2025. Available from: <https://arxiv.org/abs/2304.03271>.
- [11] Programa de las Naciones Unidas para el Medio Ambiente (PNUMA). La IA plantea problemas ambientales. Esto es lo que el mundo puede hacer al respecto. Programa de las Naciones Unidas para el Medio Ambiente. 2024 September 21. Reportaje Medio ambiente bajo revisión. Available from: <https://www.unep.org/es/noticias-y-reportajes/reportajes/la-ia-plantea-problemas-ambientales-esto-es-lo-que-el-mundo-puede>.
- [12] Abi-Rafteh J, Henry N, Xu HH, Bassiri-Tehrani B, Arezki A, Kazan R, et al. Utility and Comparative Performance of Current Artificial Intelligence Large Language Models as Postoperative Medical Support Chatbots in Aesthetic Surgery. *Aesthetic Surgery Journal*. 2024 02;44(8):889-96. Available from: <https://doi.org/10.1093/asj/sjae025>.
- [13] Morley J, Machado CCV, Burr C, Cowls J, Joshi I, Taddeo M, et al.. The ethics of AI in health care: A mapping review; 2020. Available from: <https://www.sciencedirect.com/science/article/pii/S0277953620303919>.
- [14] Küster D, Schultz T. Künstliche Intelligenz und Ethik im Gesundheitswesen - Spagat oder Symbiose? *Bundesgesundheitsblatt, Gesundheitsforschung, Gesundheitsschutz*. 2023 Feb;66(2):176-83. Available from: <https://link.springer.com/article/10.1007/s00103-022-03653-5>.
- [15] Sohn J, Park Y, Yoon C, Park S, Hwang H, Sung M, et al.. Rationale-Guided Retrieval Augmented Generation for Medical Question Answering; 2024. Available from: <https://arxiv.org/abs/2411.00300>.
- [16] Ge J, Sun S, Owens J, Galvez V, Gologorskaya O, Lai JC, et al. Development of a liver disease-specific large language model chat interface using retrieval-augmented generation. *Hepatology*. 2024 November;80(5):1158-68. Available from: <https://doi.org/10.1097/HEP.0000000000000834>.
- [17] Ong CS, Obey NT, Zheng Y, Cohan A, Schneider EB. SurgeryLLM: a retrieval-augmented generation large language model framework for surgical decision support and workflow enhancement. *npj Digital Medicine*. 2024;7(1):364. Available from: <https://doi.org/10.1038/s41746-024-01391-3>.
- [18] Zhang C, Zhang H, Khan A, Kim T, Omoleye O, Abiona O, et al.. Lightweight Mobile Automated Assistant-to-physician for Global Lower-resource Areas; 2021. Available from: <https://arxiv.org/abs/2110.15127>.
- [19] Chu CH, Nyrup R, Leslie K, Shi J, Bianchi A, Lyn A, et al. Digital Ageism: Challenges and Opportunities in Artificial Intelligence for Older Adults. *The Gerontologist*. 2022 August;62(7):947-55. Available from: <https://doi.org/10.1093/geront/gnab167>.

- [20] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 (General Data Protection Regulation); 2016. Available from: <https://eur-lex.europa.eu/eli/reg/2016/679/oj>.
- [21] Health Insurance Portability and Accountability Act of 1996 (HIPAA); 1996. Available from: <https://www.hhs.gov/hipaa/for-individuals/index.html>.
- [22] Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, et al.. Language Models are Few-Shot Learners; 2020. Available from: <https://arxiv.org/abs/2005.14165>.
- [23] Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding; 2019. Available from: <https://arxiv.org/abs/1810.04805>.
- [24] Wang F, Zhang Z, Zhang X, Wu Z, Mo T, Lu Q, et al.. A Comprehensive Survey of Small Language Models in the Era of Large Language Models: Techniques, Enhancements, Applications, Collaboration with LLMs, and Trustworthiness; 2024. Available from: <https://arxiv.org/abs/2411.03350>.
- [25] Bengio Y, Ducharme R, Vincent P. A Neural Probabilistic Language Model. vol. 3; 2000. p. 932-8.
- [26] Mikolov T, Karafiát M, Burget L, Cernocký J, Khudanpur S. Recurrent neural network based language model. Proceedings of Interspeech. 2010 01;2.
- [27] Huang L, Yu W, Ma W, Zhong W, Feng Z, Wang H, et al. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. ACM Transactions on Information Systems. 2025 Jan;43(2):1–55. Available from: <http://dx.doi.org/10.1145/3703155>.
- [28] Zhou Y, Muresanu AI, Han Z, Paster K, Pitis S, Chan H, et al.. Large Language Models Are Human-Level Prompt Engineers; 2023. Available from: <https://arxiv.org/abs/2211.01910>.
- [29] Chung HW, Hou L, Longpre S, Zoph B, Tay Y, Fedus W, et al.. Scaling Instruction-Finetuned Language Models; 2022. Available from: <https://arxiv.org/abs/2210.11416>.
- [30] Iyer S, Lin XV, Pasunuru R, Mihaylov T, Simig D, Yu P, et al.. OPT-IML: Scaling Language Model Instruction Meta Learning through the Lens of Generalization; 2023. Available from: <https://arxiv.org/abs/2212.12017>.
- [31] Ouyang L, Wu J, Jiang X, Almeida D, Wainwright CL, Mishkin P, et al.. Training language models to follow instructions with human feedback; 2022. Available from: <https://arxiv.org/abs/2203.02155>.
- [32] Barnett S, Kurniawan S, Thudumu S, Brannelly Z, Abdelrazek M. Seven Failure Points When Engineering a Retrieval Augmented Generation System; 2024. Available from: <https://arxiv.org/abs/2401.05856>.

- [33] Bruckhaus T. RAG Does Not Work for Enterprises; 2024. Available from: <https://arxiv.org/abs/2406.04369>.
- [34] Farquhar S, Kossen J, Kuhn L, et al. Detecting hallucinations in large language models using semantic entropy. *Nature*. 2024;630:625-30.
- [35] Abi-Rafeh J, Xu HH, Kazan R, Tevlin R, Furnas H. Large Language Models and Artificial Intelligence: A Primer for Plastic Surgeons on the Demonstrated and Potential Applications, Promises, and Limitations of ChatGPT. *Aesthetic Surgery Journal*. 2024 Feb 15;44(3):329-43.
- [36] Ge J, Sun S, Owens J, Galvez V, Gologorskaya O, Lai JC, et al. Development of a Liver Disease-Specific Large Language Model Chat Interface using Retrieval Augmented Generation. *medRxiv*. 2023 Nov. Preprint. Available from: <https://doi.org/10.1101/2023.11.10.23298364>.
- [37] Wang Y, Leutner S, Ingrisich M, Klein C, Hinske LC, Danhauser K. Optimizing Data Extraction: Harnessing RAG and LLMs for German Medical Documents; 2024.
- [38] Yepes AJ, You Y, Milczek J, Laverde S, Li R. Financial Report Chunking for Effective Retrieval Augmented Generation; 2024. Available from: <https://arxiv.org/abs/2402.05131>.
- [39] Zhong Z, Liu H, Cui X, Zhang X, Qin Z. Mix-of-Granularity: Optimize the Chunking Granularity for Retrieval-Augmented Generation; 2025. Available from: <https://arxiv.org/abs/2406.00456>.
- [40] Yang R, Ning Y, Keppo J, et al. Generación aumentada por recuperación para inteligencia artificial generativa en el cuidado de la salud. *npj Health Systems*. 2025;2:2. Available from: <https://doi.org/10.1038/s44401-024-00004-1>.
- [41] Gupta S, Ranjan R, Singh SN. A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions; 2024. Available from: <https://arxiv.org/abs/2410.12837>.
- [42] Fang K, Tang C, Wang J. Evaluating simulated teaching audio for teacher trainees using RAG and local LLMs. *Scientific Reports*. 2025 January 29;15(1):3633.
- [43] Raminedi S, Shridevi S, Won D. Multi-modal transformer architecture for medical image analysis and automated report generation. *Scientific Reports*. 2024 August;14(1):19281.
- [44] Tevissen Y, Guetari K, Petitpont F. Towards Retrieval Augmented Generation over Large Video Libraries; 2024. Available from: <https://arxiv.org/abs/2406.14938>.
- [45] Su C, Wen J, Kang J, Wang Y, Su Y, Pan H, et al.. Hybrid RAG-empowered Multi-modal LLM for Secure Data Management in Internet of Medical Things: A Diffusion-based Contract Approach; 2024. Available from: <https://arxiv.org/abs/2407.00978>.

- [46] Xue J, Zheng M, Hu Y, Liu F, Chen X, Lou Q. BadRAG: Identifying Vulnerabilities in Retrieval Augmented Generation of Large Language Models; 2024. Available from: <https://arxiv.org/abs/2406.00083>.
- [47] LangChain. LangChain Documentation; 2025. Último acceso: 20 de marzo de 2025. Available from: <https://python.langchain.com/docs/introduction/>.
- [48] LlamaIndex. LlamaIndex Documentation; 2025. Último acceso: 20 de marzo de 2025. Available from: <https://docs.llamaindex.ai/en/stable/>.
- [49] deepset ai. Haystack Documentation; 2025. Último acceso: 20 de marzo de 2025. Available from: <https://docs.haystack.deepset.ai/docs/intro>.
- [50] Perez A, Vizcaino X. Advanced ingestion process powered by LLM parsing for RAG system; 2024. Available from: <https://arxiv.org/abs/2412.15262>.
- [51] In: Zong FyLWyNR Chengqing y Xia, editor. Hallazgos de la Asociación de Lingüística Computacional: ACL-IJCNLP 2021; Available from: <https://aclanthology.org/2021.hallazgos-acl.29/>.
- [52] Chen LC, Pardeshi MS, Liao YX, Pai KC. Application of retrieval-augmented generation for interactive industrial knowledge management via a large language model. Computer Standards Interfaces. 2025;94:103995. Available from: <https://www.sciencedirect.com/science/article/pii/S0920548925000248>.
- [53] Agarwal S, Sundaresan S, Mitra S, Mahapatra D, Gupta A, Sharma R, et al.. Cache-Craft: Managing Chunk-Caches for Efficient Retrieval-Augmented Generation; 2025. Available from: <https://arxiv.org/abs/2502.15734>.
- [54] LangChain. Document Transformers - LangChain; 2025. Accedido el: 20 de marzo de 2025. Available from: https://python.langchain.com/v0.1/docs/modules/data_connection/document_transformers/.
- [55] Nguyen PV, Tran MN, Nguyen L, Dinh D. Advancing Vietnamese Information Retrieval with Learning Objective and Benchmark; 2025. Available from: <https://arxiv.org/abs/2503.07470>.
- [56] Rau D, Wang S, Déjean H, Clinchant S. Context Embeddings for Efficient Answer Generation in RAG; 2024. Available from: <https://arxiv.org/abs/2407.09252>.
- [57] Han Y, Liu C, Wang P. A Comprehensive Survey on Vector Database: Storage and Retrieval Technique, Challenge; 2023. Available from: <https://arxiv.org/abs/2310.11703>.
- [58] Öztürk E, Mesut A. Performance Analysis of Chroma, Qdrant, and FAISS Databases. UNITECH – Selected Papers. 2024. Available from: https://unitechsp.tugab.bg/images/2024/4-CST/s4_p72_v3.pdf.
- [59] Inc M. What are Vector Databases?; 2024. Accessed: March 23, 2025. Available from: https://www.mongodb.com/es/resources/basics/databases/vector-databases?utm_source=chatgpt.com.

- [60] MongoDB. MongoDB Atlas Vector Search; 2025. Último acceso: 23 de marzo de 2025. Available from: <https://www.mongodb.com/es/products/platform/atlas-vector-search>.
- [61] Facebook AI Research. FAISS: A Library for Efficient Similarity Search; 2025. Último acceso: 23 de marzo de 2025. Available from: <https://github.com/facebookresearch/faiss>.
- [62] ChromaDB. ChromaDB Documentation: Introduction; 2025. Último acceso: 23 de marzo de 2025. Available from: <https://docs.trychroma.com/docs/overview/introduction>.
- [63] LangChain. Retrievers - LangChain Documentation; 2025. Último acceso: 20 de marzo de 2025. Available from: <https://python.langchain.com/docs/integrations/retrievers/>.

.1. Anexo 1. Preguntas PEC2

El grado de cumplimiento de los objetivos ha sido adecuado. Durante el desarrollo del PEC2, se han consolidado los conocimientos sobre los modelos de LLM y la arquitectura RAG, logrando completar las tareas establecidas previamente en el cronograma de Gantt. Se han explorado diferentes enfoques para el preprocesamiento de los documentos de las guías clínicas, incluyendo el análisis de los metadatos, diferentes técnicas de chunking y estrategias de vectorización, embedding y almacenamiento de vectrostore. Además, se han evaluado distintos modelos en entornos clínicos, permitiendo comparar el rendimiento de diferentes retrievers y generators con el objetivo de seleccionar la mejor arquitectura RAG para el caso de estudio. Las principales actividades realizadas durante el período del PEC2 han incluido:

- Exploración de metadatos procedentes de las guías clínicas obtenidos durante el proceso de parseo.
- Implementación y ajuste de estrategias de chunking para la fragmentación eficiente de los documentos clínicos.
- Aplicación de técnicas de vectorización y almacenamiento de datos en la base de datos vectorial.
- Validación preliminar de la calidad del vectorstore mediante métricas que evalúan la recuperación y evaluación de los embeddings empleados.
- Selección de modelos de lenguaje para el sistema RAG y análisis de sus capacidades en entornos clínicos.
- Selección de la arquitectura más adecuada, basada en experimentos y evaluaciones realizadas.

Como parte del desarrollo del PEC2, se ha creado un repositorio de GitHub que centraliza el código, los experimentos y la documentación del proyecto. De este modo, el repositorio facilita

la trazabilidad del trabajo y permite futuras mejoras. El repositorio consta de un notebook con el código empleado, así como el repositorio de la guías clínicas empleadas, los documentos parseados y limpiados, los vectores y el vectorstore.

Repositorio de GitHub: <https://github.com/dcamacmon/TFM—RAG/>

.2. Anexo 2. Tabla de metadatos