

UNIVERSIDAD DE VALLADOLID

MÁSTER UNIVERSITARIO
Ingeniería Informática



TRABAJO FIN DE MÁSTER

Implementación de técnicas de RAG (Retrieval Augmented Generation) sobre LLM (Large Language Models) para la extracción y generación de documentos en las entidades públicas

Realizado por **Miguel Ángel Collado Alonso**



Universidad de Valladolid

19 de septiembre de 2024

Tutor: David Escudero Mancebo

Universidad de Valladolid



Máster universitario en Ingeniería Informática

D. David Escudero Mancebo, profesor del departamento de Informática, área de Ciencia de la Computación e Inteligencia Artificial.

Expone:

Que el alumno D. Miguel Ángel Collado Alonso, ha realizado el Trabajo final de Máster en Ingeniería Informática titulado "Implementación de técnicas de RAG (Retrieval Augmented Generation) sobre LLM (Large Language Models) para la extracción y generación de documentos en las Entidades Públicas".

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Valladolid, 19 de septiembre de 2024

Vº. Bº. del Tutor:

D. David Escudero Mancebo

Agradecimientos

A María Jesús, mi pareja, por haber tenido tanta paciencia conmigo durante todos los meses que ha durado el Máster. Te quiero.

A mis niños de tres y cuatro patas, que les he tenido un poco olvidados este tiempo.

A David, mi tutor, por haber tenido la genial idea de realizar este trabajo y haber confiado en mí para hacerlo.

Al profesor Valentín, por haberme ayudado con la configuración inicial del proyecto.

Al grupo de investigación ECA-SIMM por dejarme utilizar uno de sus equipos.

Por último, y no por ello menos importante, a mi familia, gracias por estar siempre ahí.

*Nuestra inteligencia es lo que nos hace humanos,
y la inteligencia artificial es una extensión de esa cualidad*

Yann LeCun

Resumen

Un viejo desafío en el campo de la inteligencia artificial es “enseñar” a las máquinas a entender cómo hablan y escriben los humanos, conocido como procesamiento de lenguaje natural (PLN). Sin embargo, desde hace poco más de dos años, estamos asistiendo a la caída de este antiguo bastión con la llegada de los modelos grandes del lenguaje (LLM) y los interfaces conversacionales. Los LLM son modelos de inteligencia artificial que se entrenan utilizando algoritmos de Deep Learning sobre conjuntos enormes de información generada por humanos. De esta manera, una vez entrenados, han aprendido la forma en la que los humanos utilizamos la palabra hablada y escrita, así que son capaces de ofrecernos respuestas generales y con un patrón muy parecido a nuestra forma de contestar a las preguntas que les hacemos. Sin embargo, si buscamos respuestas precisas en un contexto determinado, los LLM por sí solos no proporcionarán respuestas concretas o habrá una alta probabilidad de que se inventen completamente la respuesta. En este trabajo, se explica y desarrolla una de las técnicas clave que hace posible que estos sistemas nos respondan con relativa precisión a las preguntas que les hacemos, esta tecnología se denomina Generación Aumentada por Recuperación o RAG, del inglés Retrieval Augmented Generation. Para ello se comparan diversos modelos LLM, se configuran un conjunto de herramientas para lograr inferir los modelos, y se desarrolla un código fuente para, finalmente, lograr realizar una interfaz web, a modo de chatbot, que permita interactuar con el usuario y realizar RAG sobre un documento elegido por el mismo.

Descriptores

Inteligencia Artificial (IA), Procesamiento del Lenguaje Natural (PLN), Modelo Grande del Lenguaje (LLM), Generación Aumentada por Recuperación (RAG), Deep Learning.

Abstract

A long-standing challenge in the field of artificial intelligence is “teaching” machines to understand how humans speak and write, known as natural language processing (NLP). However, for just over two years, we have been witnessing the fall of this old bastion with the arrival of large language models (LLM) and conversational interfaces. LLMs are artificial intelligence models that are trained using Deep Learning algorithms on huge sets of human-generated data. In this way, once trained, they have learned the way in which humans use the spoken and written word, so they are able to offer us general answers with a pattern very similar to our way of answering the questions we ask them. However, if we are looking for precise answers in a given context, LLMs alone will not provide concrete answers or there will be a high probability that they will completely make up the answer. In this work, one of the key techniques that makes it possible for these systems to respond with relative precision to the questions we ask them is explained and developed. This technology is called Retrieval Augmented Generation or RAG. To do this, various LLM models are compared, a set of tools are configured to infer the models, and a source code is developed to, finally, create a web interface, like a chatbot, that allows interacting with the user and performing RAG on a document chosen by him.

Keywords

Artificial Intelligence (AI), Natural Language Processing (NLP), Large Language Model (LLM), Retrieval Augmented Generation (RAG), Deep Learning.

Índice General

Índice General	III
Índice de Figuras	V
Índice de Tablas	VI
1. Introducción	1
1.1 - Objetivo del proyecto	2
1.2 – Estructura del documento	3
2. Contexto	4
2.1 - Modelo Llama 3	4
2.2 - Modelo Gemma 2	5
2.3 - Modelo Phi 3	6
2.4 - Evolución de RAG	7
3. Plan de trabajo.....	11
4. Metodología	13
4.1 Fases de desarrollo de sistemas RAG	15
4.1.1 - Fase de Recuperación.....	15
4.1.2 - Alinear consultas y documentos.....	15
4.1.3 - Alineación del Retriever y LLM	16
4.1.4 - Fase de Generación	16
4.1.5 - Fase de Aumento.....	17
4.2 – Implementación del sistema RAG	17

4.3 – Evaluación de calidad	20
4.3.1 - Métricas	20
4.3.2 – Test de evaluación	23
5. Resultados obtenidos	29
5.1 – Demostración del sistema	33
6. Discusión y conclusiones	35
Apéndices	37
Apéndice A - Instalación y configuración del sistema	38
A.1 - Docker	38
A.2 - Ollama	39
A.3 - Python y módulo VENV de Python	44
A.4 - Librerías y paquetes Python necesarios	45
Apéndice B - Código fuente del sistema	49
B.1 - Fichero app.py (aplicación principal)	49
B.2 - Fichero chainlit.md (texto del Readme o Léeme)	55
B.3 - Fichero config.toml (parámetros de configuración y estilo)	56
B.4 - Fichero es-Es.json (traducción al castellano)	58
B.5 - Código auxiliar del Cuaderno Jupyter para el cálculo de las métricas.	63
Apéndice C - Respuestas del sistema a las preguntas planteadas	66
Apéndice D - Presupuesto para implementar el sistema RAG en una Admon. Pública	79
Referencias y Bibliografía	82

Índice de Figuras

Figura 2.1 – Benchmark del modelo Meta LLama 3 frente a otros	5
Figura 2.2 - Benchmark del modelo Google Gemma 2 frente a otros	6
Figura 2.3 - Benchmark del modelo Microsoft Phi-3 frente a otros	7
Figura 2.4 - Evolución de RAG	8
Figura 3.1 - Diagrama Gantt con la planificación del trabajo	11
Figura 4.1 - Diagrama de la metodología RAG	14
Figura 4.2 - Esquema de cómo se implementa el sistema RAG de forma práctica.	20
Figura 5.1 - Interfaz de usuario. Pregunta al sistema con el modo RAG desactivado	33
Figura 5.2 - Interfaz de usuario. Pregunta al sistema con el modo RAG activado.	34
Figura A.1 - Estructura de carpetas y ficheros de chainlit	46
Figura A.2 - Diagrama de base de datos Chroma	48

Índice de Tablas

Tabla 4.1 – Batería de preguntas realizadas al sistema	23
Tabla 4.2 – Ficheros que contienen el contexto para las respuestas	23
Tabla 4.3 – Respuestas correctas o esperadas	25
Tabla 4.4 – Métricas de la P1 sin RAG	25
Tabla 4.5 – Métricas de la P2 sin RAG	25
Tabla 4.6 – Métricas de la P3 sin RAG	26
Tabla 4.7 – Métricas de la P4 sin RAG	26
Tabla 4.8 – Métricas de la P1 con RAG	26
Tabla 4.9 – Métricas de la P2 con RAG	26
Tabla 4.10 – Métricas de la P3 con RAG	27
Tabla 4.11 – Métricas de la P4 con RAG	27
Tabla 4.12 – Tiempo en contestar la respuesta sin RAG	27
Tabla 4.13 – Tiempo en contestar la respuesta con RAG	27
Tabla 4.14 – Tiempo que tarda en procesar el fichero	28
Tabla 4.15 – Consumo de memoria GPU de los modelos	28
Tabla 5.1 – Media aritmética de métricas sin RAG	29
Tabla 5.2 – Media de las métricas con RAG	31
Tabla C.1 – Respuestas a la P1 sin RAG	68
Tabla C.2 – Respuestas a la P2 sin RAG	69
Tabla C.3 – Respuestas a la P3 sin RAG	71

Tabla C.4 – Respuestas a la P4 sin RAG	73
Tabla C.5 – Respuestas a la P1 con RAG	75
Tabla C.6 – Respuestas a la P2 con RAG	75
Tabla C.7 – Respuestas a la P3 con RAG	77
Tabla C.8 – Respuestas a la P4 con RAG	78
Tabla D.1 – Presupuesto con implantación On-Premise.....	80
Tabla D.2 – Presupuesto con implantación en la nube de AWS.....	81

1. Introducción

Un modelo de lenguaje es un tipo de modelo de inteligencia artificial diseñado para entender y generar texto en lenguaje natural. En esencia, el modelo aprende las características estadísticas de una gran cantidad de texto y utiliza ese conocimiento para predecir o generar texto coherente y relevante en función de las entradas que recibe.

A su vez, un modelo de lenguaje grande o LLM (siglas en inglés para Large Language Model), también llamado modelo de lenguaje de gran tamaño, es un modelo de lenguaje que consta de una red neuronal con billones de parámetros, entrenados con grandes cantidades de texto sin etiquetar mediante aprendizaje auto supervisado. La aparición de estos modelos en el año 2018 está cambiando el enfoque de la investigación del procesamiento del lenguaje natural (PLN) alejándose del paradigma anterior de entrenar modelos supervisados especializados para tareas específicas. Los LLM son modelos de propósito general que destacan en una amplia gama de tareas, en lugar de estar capacitados para una tarea específica, como el análisis de sentimientos, el reconocimiento de entidades nombradas o el razonamiento matemático. De esta manera, una vez entrenados, han aprendido la forma en la que los humanos hablan y escriben, así que son capaces de ofrecer respuestas generales, y con un patrón muy parecido al lenguaje natural, a las preguntas que se les planteen [1]. Algunos de los modelos más conocidos actualmente son los GPT-3.5 y GPT-4 de OpenAI, Gemini y Gemma de Google, LLAMA-2 y LLAMA-3 de Meta AI, Mistral de Mistral AI, etc.

Frente a las aproximaciones convencionales de aprendizaje supervisado, en los modelos LLM se emplea el aprendizaje auto supervisado, de esta forma, el modelo se entrena utilizando datos no etiquetados, pero se utiliza alguna forma de señal interna o estructura del conjunto de datos para generar etiquetas implícitas. Por ejemplo, en el procesamiento del lenguaje natural, se puede entrenar un modelo para predecir la siguiente palabra en una oración, donde la oración completa se considera la entrada, pero se oculta una palabra y se entrena al modelo para

predecirla. Esto proporciona una forma de aprender representaciones útiles del lenguaje sin depender de etiquetas externas [2].

Si se buscan respuestas precisas en un contexto determinado, los LLM por sí solos no proporcionarán respuestas específicas o habrá una alta probabilidad de que “alucinen” y se inventen completamente la respuesta. Que los LLM “alucinen” significa que generan texto inexacto, sin sentido o desconectado. Por ejemplo, si se pregunta a un LLM sobre datos actuales, como quién es el alcalde de una ciudad o el presidente de un gobierno, probablemente su respuesta no será correcta, ya que el modelo ha sido entrenado en una fecha en concreto y la actualidad cambia muy rápidamente. O si se pregunta al LLM sobre datos concretos o privados de una organización su respuesta será demasiado generalista y poco exacta. Se pueden mejorar levemente las respuestas gracias a las técnicas de fine tuning sobre los LLM, que consisten en ajustar un modelo pre-entrenado con un conjunto de datos específico o tarea, modificando ligeramente los pesos y parámetros del modelo, así los LLM pueden adaptarse a necesidades particulares, mejorando su rendimiento y precisión en tareas concretas [3].

Para solventar este problema se han desarrollado las técnicas RAG, que mejoran la calidad de las respuestas en contextos específicos, en una disciplina concreta o en base a repositorios de conocimiento privados como bases de datos de empresas. El objetivo de las técnicas RAG es recuperar hechos de una base de conocimientos externa para garantizar que los modelos de lenguaje devuelven información precisa y actualizada.

La generación mejorada por recuperación (RAG), del inglés “Retrieval Augmented Generation” es el proceso de optimización de la salida de un modelo lingüístico de gran tamaño (LLM), de modo que haga referencia a una base de conocimientos autorizada fuera de los orígenes de datos de entrenamiento antes de generar una respuesta. RAG extiende las ya poderosas capacidades de los LLM a dominios específicos o a la base de conocimientos interna de una organización, todo ello sin la necesidad de volver a entrenar el modelo [4].

1.1 - Objetivo del proyecto

El proyecto busca desarrollar un sistema con técnicas RAG sobre LLM para mejorar la extracción y generación automática de información en una entidad pública, optimizando la gestión diaria de numerosos documentos. La aplicación de técnicas de inteligencia artificial, especialmente en el campo del procesamiento del lenguaje natural (PLN), puede ofrecer soluciones significativas para la automatización de tareas relacionadas con documentos.

Los chatbot basados en modelos LLM que existen actualmente, como ChatGPT [5] o Bing [6], ya implementan opciones de hacer RAG sobre un documento, limitados generalmente en sus versiones gratuitas, realizan esta técnica sobre un documento que se anexa al chatbot. El problema está en que si se requiere que se utilice una gran cantidad de documentos o si se tienen que cumplir normas de privacidad con el contenido de estos documentos, no se podrían utilizar estos chatbots semi-comerciales.

La motivación de este proyecto radica en la necesidad de modernizar y optimizar los sistemas de gestión documental en las entidades públicas, facilitando el acceso a la información, reduciendo los tiempos de respuesta y mejorando la calidad de los documentos generados. La implementación de estas tecnologías contribuye a una mayor transparencia y eficiencia administrativa, beneficiando tanto a los empleados públicos como a los ciudadanos.

1.2 – Estructura del documento

El presente documento desarrolla las actividades sobre las técnicas de implementar RAG realizadas para el Trabajo de Fin de Máster en Ingeniería Informática.

Los capítulos en los que se divide la memoria son los siguientes:

- **Capítulo 1. Introducción.** En el que se explica que son los modelos grandes del lenguaje (LLM), la problemática que surge con las respuestas a un contexto específico y la técnica de hacer RAG para solventar el problema. También se plantean los objetivos del proyecto, hasta donde se pretende llegar.
- **Capítulo 2. Contexto.** Se muestra el estado del arte de los LLM y de las técnicas para la implementación de RAG.
- **Capítulo 3. Plan de trabajo.** Se enumeran las tareas que se han desarrollado y el tiempo necesario para ello.
- **Capítulo 4. Metodología.** En el que se explica cómo se ha trabajado, que modelos se han analizado y con qué técnicas y herramientas se ha implementado el sistema. Así como las métricas empleadas para evaluar los resultados.
- **Capítulo 5. Resultados obtenidos.** En este capítulo se muestran las métricas obtenidas y una demostración uso de la aplicación de usuario.
- **Capítulo 6. Discusión y conclusiones.** Limitaciones encontradas y trabajo realizado. Así como los objetivos conseguidos y el trabajo futuro para el proyecto.

2. Contexto

La gestión de documentos en las entidades públicas es una tarea que afecta a la eficiencia administrativa, la transparencia y la calidad del servicio ofrecido a los ciudadanos. El volumen de datos y documentos que estas entidades manejan es inmenso, lo que dificulta su gestión y acceso eficiente. Las técnicas tradicionales de procesamiento de documentos suelen ser insuficientes para manejar esta complejidad y volumen de información.

La implementación de técnicas avanzadas como la Recuperación Aumentada por Generación (RAG) sobre Modelos de Lenguaje Grandes (LLM) se plantea para mejorar significativamente estos procesos. Estas técnicas combinan la capacidad de recuperación de información relevante con la generación de textos coherentes y contextualmente adecuados, permitiendo no solo extraer información de grandes bases de datos, sino también generar documentos precisos y personalizados.

A continuación, se detallan algunos de los modelos LLM más utilizados actualmente.

2.1 - Modelo Llama 3

Son una familia de modelos desarrollados por Meta Inc., la última versión es la tercera, disponibles en tamaños de parámetros 8B y 70B (preentrenados o ajustados por instrucciones). Los modelos ajustados por instrucciones de Llama 3 están ajustados y optimizados para casos de uso de diálogo/chat y superan a muchos de los modelos de chat de código abierto disponibles en los benchmark.

Meta Llama 3 Instruct model performance

	Meta Llama 3 8B	Gemma 7B - It Measured	Mistral 7B Instruct Measured
MMLU 5-shot	68.4	53.3	58.4
GPQA 0-shot	34.2	21.4	26.3
HumanEval 0-shot	62.2	30.5	36.6
GSM-BK 8-shot, CoT	79.6	30.6	39.9
MATH 4-shot, CoT	30.0	12.2	11.0

	Meta Llama 3 70B	Gemini Pro 1.5 Published	Claude 3 Sonnet Published
MMLU 5-shot	82.0	81.9	79.0
GPQA 0-shot	39.5	41.5 CoT	38.5 CoT
HumanEval 0-shot	81.7	71.9	73.0
GSM-BK 8-shot, CoT	93.0	91.7 11-shot	92.3 0-shot
MATH 4-shot, CoT	50.4	58.5 Minerva prompt	40.5

Figura 2.1 – Benchmark del modelo Meta LLama 3 frente a otros. [7]

En la figura 4.1 se puede comprobar como destaca el modelo Llama 3 en los distintos benchmark frente a modelos de código abierto como la primera versión del modelo Gemma de Google [8] o Mistral [35] así como a los modelos comerciales de Google Gemini Pro 1.5 [36] o el menos conocido Claude 3 de Anthropic [37].

2.2 - Modelo Gemma 2

Diseñados por Google, son una familia de modelos abiertos ligeros y de última generación creados a partir de la misma investigación y tecnología que se usaron para crear los modelos de Gemini. El modelo Gemma 2 de Google está disponible en dos tamaños, 9B y 27B. Presenta una arquitectura completamente nueva diseñada para brindar rendimiento y eficiencia líderes en su clase.

	BENCHMARK	METRIC	Gemma 2		Llama 3		Grok-1
			9B	27B	8B	70B	314B
General	MMLU	5-shot, top-1	71.3	75.2	66.6	79.5	73.0
Reasoning	BBH	3-shot, CoT	68.2	74.9	61.1	81.3	–
	HellaSwag	10-shot	81.9	86.4	82	–	–
Math	GSM8K	5-shot, maj@1	68.6	74.0	45.7	–	62.9 (8-shot)
	MATH	4-shot	36.6	42.3	–	–	23.9
Code	HumanEval	pass@1	40.2	51.8	–	–	63.2 (0-shot)

Figura 2.2 - Benchmark del modelo Google Gemma 2 frente a otros. [8]

En la figura 4.2 se aprecia como Google ha mejorado su modelo Gemma con la versión 2. En los benchmark gana posiciones frente a Llama 3 de Meta [7] o el modelo de LLM abierto más grande hasta el momento, con 314 billones de parámetros, el Grok-1 de xAI [38], propiedad del famoso empresario Elon Musk.

2.3 - Modelo Phi 3

Phi 3: Diseñados por Microsoft, familia de modelos abiertos Phi-3, son los modelos de lenguaje pequeños más capaces y rentables disponibles, según asegura la propia compañía [34]. Los modelos Phi-3 superan a los modelos del mismo tamaño y del siguiente tamaño en una variedad de puntos de referencia que evalúan las capacidades de lenguaje, codificación y matemáticas, gracias a las innovaciones de entrenamiento desarrolladas por los investigadores de Microsoft.

Hay dos versiones, phi3-mini de 3B de parámetros y phi3-medium con 14B de parámetros.

Category	Benchmark	Phi-3-Medium		Mistral-8x22B	Llama-3-70B-Instruct	GPT3.5-Turbo-1106	Claude-3 Sonnet	Gemini 1.0 Pro
		Phi-3-Medium-4K-In	Phi-3-Medium-128K-In					
Popular Aggregate Benchmarks	MMLU (5-shot)	78.0	76.6	76.2	80.2	71.4	73.9	66.7
Language Understanding	HellaSwag (5-shot)	82.4	81.6	79.0	82.6	78.8	79.2	76.2
Reasoning	WinoGrande (5-shot)	81.5	78.9	75.3	83.3	68.8	81.4	72.2
	Social IQA (5-shot)	80.2	79.0	78.2	81.1	68.3	80.2	75.4
	TruthfulQA (MC2) (10-shot)	75.1	74.3	67.4	81.9	67.7	77.8	72.6
	MedQA (2-shot)	69.9	67.6	67.9	78.5	63.4	67.9	58.2
Factual Knowledge	TriviaQA (5-shot)	73.9	73.9	84.5	78.5	85.8	65.7	80.2
Math	GSM8K CoT (8-shot)	91.0	87.5	83.8	93.5	78.1	79.1	80.4
Code generation	HumanEval (0-shot)	62.2	58.5	39.6	78.7	62.2	65.9	64.4
	MBPP (3-shot)	75.2	73.8	70.7	81.3	77.8	79.4	73.2

Figura 2.3 - Benchmark del modelo Microsoft Phi-3 frente a otros. [9]

Como refleja la figura 4.3, el modelo Phi-3 obtiene muy buenos resultados en los distintos benchmark frente a los ya mencionados modelos de código abierto Mistral [35], Llama3 de Meta [7] o GPT3.5-Turbo de OpenAI [39] así como a los modelos comerciales de Google Gemini Pro 1.0 [36] o Claude 3 de Anthropic [37].

2.4 - Evolución de RAG

En los últimos años, los sistemas RAG han evolucionado de Naive RAG a Advanced RAG y Modular RAG. Esta evolución se ha producido para abordar ciertas limitaciones en torno al rendimiento, el costo y la eficiencia. [10]

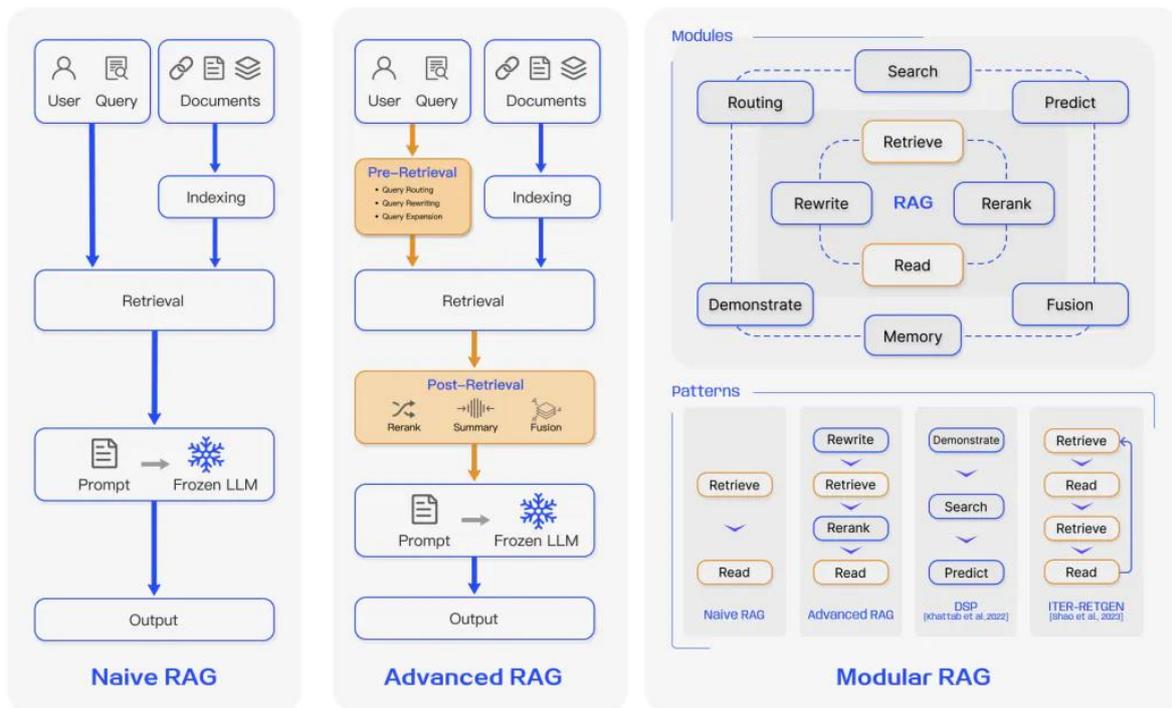


Figura 2.4 - Evolución de RAG [10]

Naive RAG sigue el proceso tradicional de indexación, recuperación y generación. Se utiliza una entrada del usuario para consultar documentos relevantes que luego se combinan con un mensaje y se pasan al modelo para generar una respuesta final. El historial conversacional se puede integrar en el mensaje si la aplicación implica interacciones de diálogo de varios turnos.

Naive RAG tiene limitaciones como baja precisión (fragmentos recuperados desalineados) y baja recuperación (no se pueden recuperar todos los fragmentos relevantes). También es posible que el LLM reciba información desactualizada, que es uno de los principales problemas que un sistema RAG debería intentar resolver inicialmente. Esto conduce a problemas de alucinaciones y respuestas deficientes e inexactas.

Cuando se aplica el aumento, también podría haber problemas de redundancia y repetición. Cuando se utilizan varios pasajes recuperados, la clasificación y la conciliación de estilo/tono también son clave. Otro desafío es garantizar que la tarea de generación no dependa demasiado de la información aumentada, lo que puede llevar a que el modelo simplemente reitere el contenido recuperado.

Advanced RAG ayuda a abordar los problemas presentes en Naive RAG, como mejorar la calidad de la recuperación, lo que podría implicar la optimización de los procesos de pre-recuperación, recuperación y pos-recuperación.

El proceso de recuperación previa implica la optimización de la indexación de datos, cuyo objetivo es mejorar la calidad de los datos que se indexan a través de cinco etapas: mejorar la granularidad de los datos, optimizar las estructuras del índice, agregar metadatos, optimización de la alineación y recuperación mixta.

La etapa de recuperación se puede mejorar aún más optimizando el propio modelo de embedding, lo que afecta directamente la calidad de los fragmentos que componen el contexto. Esto se puede hacer ajustando la incrustación para optimizar la relevancia de la recuperación o empleando embedding dinámicos que capturen mejor la comprensión contextual.

La optimización de la recuperación posterior se centra en evitar los límites de la ventana de contexto y tratar con información ruidosa o que pueda distraer. Un enfoque común para abordar estos problemas es la reclasificación, que podría implicar enfoques como la reubicación del contexto relevante en los bordes del mensaje o el recalcular de la similitud semántica entre la consulta y los fragmentos de texto relevantes.

Por último, Modular RAG mejora los módulos funcionales, como la incorporación de un módulo de búsqueda para la recuperación de similitudes y la aplicación de fine-tuning en el recuperador. Tanto Naive RAG como Advanced RAG son casos especiales de Modular RAG y están formados por módulos fijos. Los módulos RAG extendidos incluyen búsqueda, memoria, fusión, enrutamiento, predicción y adaptador de tareas que resuelven diferentes problemas. Estos módulos se pueden reorganizar para adaptarse a contextos de problemas específicos. Por lo tanto, Modular RAG se beneficia de una mayor diversidad y flexibilidad, ya que puede agregar o reemplazar módulos o ajustar el flujo entre módulos según los requisitos de la tarea.

Dada la mayor flexibilidad en la construcción de sistemas RAG, se han propuesto otras técnicas de optimización importantes para optimizar las tuberías RAG, que incluyen:

- Hybrid Search Exploration [40]: este enfoque aprovecha una combinación de técnicas de búsqueda como la búsqueda basada en palabras clave y la búsqueda semántica para recuperar información relevante y rica en contexto; esto es útil cuando se trata de diferentes tipos de consultas y necesidades de información.

- Recursive Retrieval y Query Engine [41]: implica un proceso de recuperación recursiva que puede comenzar con pequeños fragmentos semánticos y posteriormente recuperar fragmentos más grandes que enriquecen el contexto; esto es útil para equilibrar la eficiencia y la información rica en contexto.
- StepBack-prompt [42]: una técnica de prompting que permite a los LLM realizar abstracciones que producen conceptos y principios que guían el razonamiento; esto conduce a respuestas mejor fundamentadas cuando se adopta en un marco RAG porque el LLM se aleja de instancias específicas y se le permite razonar de manera más amplia si es necesario.
- Sub-Queries [43]: existen diferentes estrategias de consulta, como consultas de árbol o consultas secuenciales de fragmentos, que se pueden utilizar para diferentes escenarios.
- Hypothetical Document Embeddings [44]: HyDE genera una respuesta hipotética a una consulta, la transforma en embeddings y la utiliza para recuperar documentos similares a la respuesta hipotética en lugar de utilizar la consulta directamente.

3. Plan de trabajo

Este trabajo de fin de máster ha sido desarrollado acorde a la siguiente planificación:

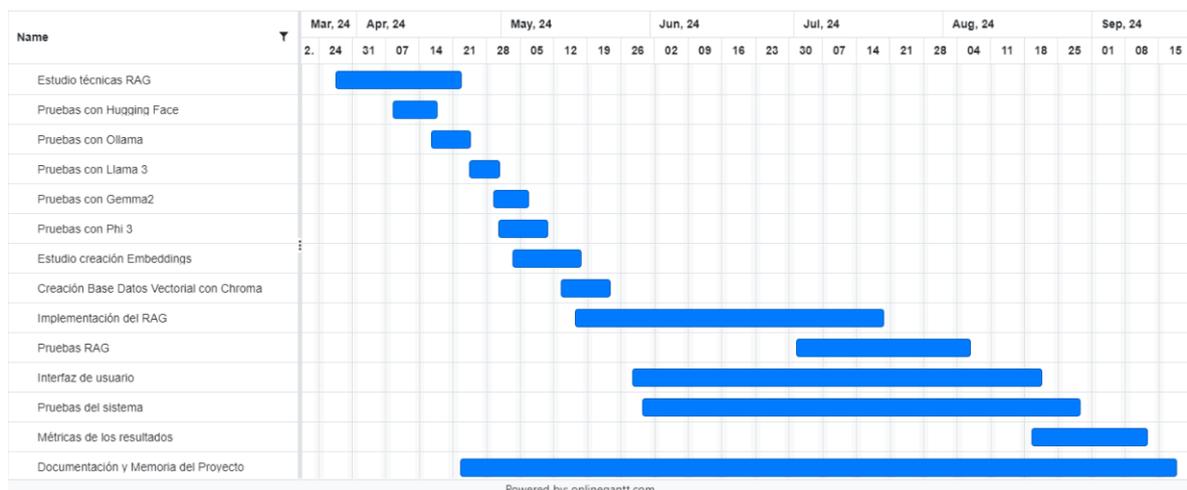


Figura 3.1 - Diagrama Gantt con la planificación del trabajo [54]

Inicialmente se comenzó con el estudio de las técnicas de RAG, aprender en qué consiste, cómo se utiliza y la finalidad del mismo.

Posteriormente se hicieron pruebas con dos plataformas para implementar el LLM, se examinó Hugging Face [47] y Ollama [15], siendo esta última la empleada para desarrollar el proyecto. Dentro de Ollama se estudió cómo inferir los modelos Llama3, Gemma2 y Phi3.

Una vez que se entendió el funcionamiento de los modelos, se comenzó a estudiar la creación de Embeddings y de la base de datos vectorial con Chroma.

Con todo lo anterior claro, se comienza a implementar el sistema RAG, con varias aproximaciones y realizando las pruebas suficientes para conocer cómo funciona.

Para mostrar el funcionamiento del sistema RAG, se desarrolló un prototipo con una interfaz de usuario amigable para poder realizar todas las pruebas del sistema.

Por último, se calcularon las métricas de los resultados obtenidos.

La memoria y documentación del proyecto se ha venido realizando casi desde el inicio de este Trabajo Fin de Máster hasta su presentación para la defensa.

4. Metodología

Utilizando Naive RAG, se toma la pregunta introducida por el usuario y se recupera un conjunto de documentos relevantes de una fuente dada, como un gestor documental o una base de conocimiento. Los documentos se concatenan como contexto con el mensaje de entrada original y se envían al generador de texto que produce el resultado final. Esto hace que RAG se adapte a situaciones en las que los hechos podrían evolucionar con el tiempo. Es muy útil ya que el conocimiento paramétrico de los LLM es estático. RAG permite que los modelos de lenguaje eviten el reentrenamiento, permitiendo el acceso a la información más reciente para generar resultados confiables a través de la generación basada en la recuperación.

Para implementar un sistema RAG es necesario realizar los diferentes pasos o componentes de la siguiente manera:

- **Entrada:** la pregunta a la que responde el sistema LLM se denomina entrada. Si no se utiliza ningún RAG, el LLM se utiliza directamente para responder a la pregunta.
- **Indexación:** si se utiliza RAG, se indexa una serie de documentos relacionados fragmentándolos primero, generando los embeddings de los fragmentos después e indexándolos en un almacén de vectores. Por inferencia, la consulta también se integra de forma similar.
- **Recuperación:** Los documentos relevantes se obtienen comparando la consulta con los vectores indexados, manteniendo la información del resto de documentos para posibles futuras consultas.

- **Generación:** los documentos relevantes se combinan con el mensaje original como contexto adicional. Después, el texto y el mensaje combinados se pasan al modelo para generar la respuesta, que luego se prepara como resultado final del sistema para el usuario.

Una base de datos vectorial es un tipo especializado de base de datos diseñada para almacenar y gestionar datos en forma de vectores numéricos, conocidos como embeddings. Estos embeddings codifican los significados semánticos y la información contextual de cualquier tipo de datos. Los datos pueden ser texto, imágenes o incluso audio. Las bases de datos vectoriales almacenan eficientemente estos embeddings y proporcionan una recuperación rápida de los embeddings a través de una búsqueda de similitud.

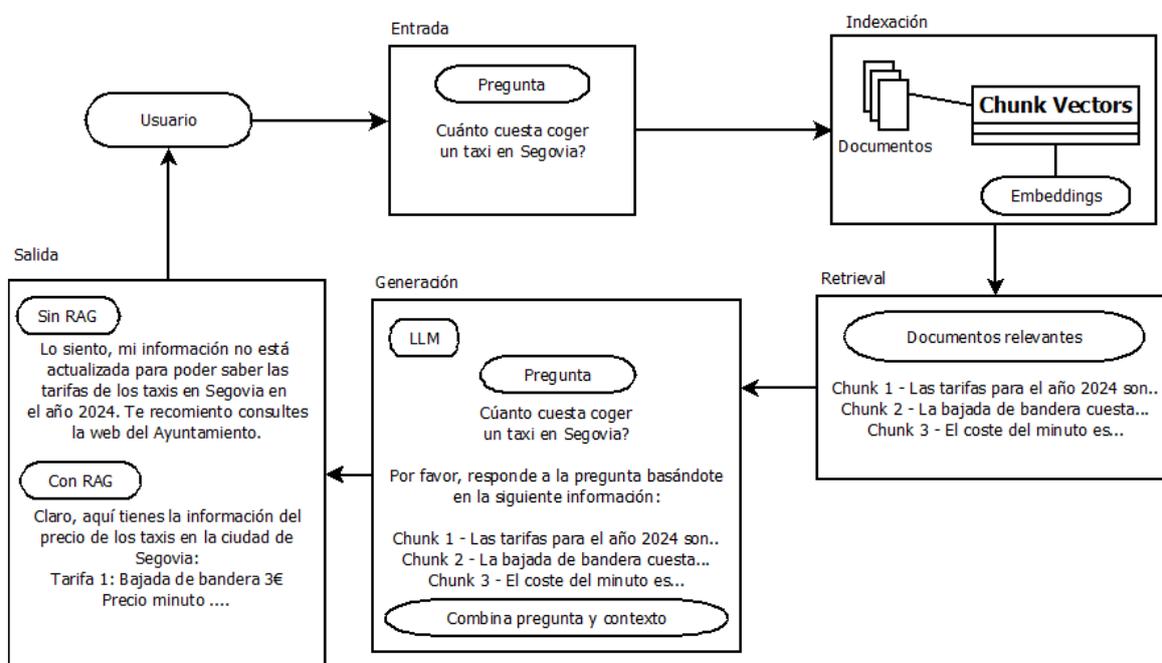


Figura 4.1 - Diagrama de la metodología RAG

En el ejemplo proporcionado en la figura 6.1, el uso directo del modelo no responde a la pregunta debido a la falta de conocimiento de los acontecimientos actuales. En cambio, cuando se utiliza RAG, el sistema puede extraer la información relevante necesaria para que el modelo responda la pregunta de manera adecuada.

4.1 Fases de desarrollo de sistemas RAG

4.1.1 - Fase de Recuperación

La recuperación es el componente de RAG que se ocupa de recuperar contexto altamente relevante. Un retriever o recuperador se puede mejorar de muchas maneras, entre ellas:

Mejora de las representaciones semánticas. Este proceso implica mejorar directamente las representaciones semánticas que impulsan al retriever. Aquí hay algunas consideraciones:

- **Chunking o Fragmentación:** un paso importante es elegir la estrategia de fragmentación adecuada, que depende del contenido con el que se trabaja y de la aplicación para la que se generan respuestas. Los diferentes modelos también muestran distintas fortalezas con las variaciones en los tamaños de bloques. Los transformadores de oraciones funcionarán mejor en oraciones individuales, pero algunos modelos funcionarán mejor con bloques que contengan 256 o 512 tokens. Otros aspectos a considerar incluyen la extensión de las preguntas de los usuarios, la aplicación y los límites de tokens, pero es común experimentar con diferentes estrategias de fragmentación para ayudar a optimizar la recuperación en su sistema RAG.
- **Fine-tuned Embeddings models:** una vez que se haya determinado una estrategia de fragmentación eficaz, es posible que sea necesario ajustar el modelo de embeddings si se está trabajando con un dominio específico. De lo contrario, es posible que las consultas de los usuarios sean completamente malinterpretadas en su aplicación.

4.1.2 - Alinear consultas y documentos

Este proceso se ocupa de alinear las consultas del usuario con las de los documentos en el espacio semántico. Esto puede ser necesario cuando la consulta de un usuario puede carecer de información semántica o contener una redacción imprecisa. Se puede mejorar la consulta del usuario utilizando técnicas como la reescritura de esta consulta o transformando los embeddings de forma optimizada.

4.1.3 - Alineación del Retriever y LLM

Este proceso se ocupa de alinear los resultados del recuperador con las preferencias de los LLM.

- Fine-tuning Retrievers o Recuperadores de ajuste fino: utiliza las señales de retroalimentación de un LLM para refinar los modelos de recuperación.
- Adaptadores: Incorpora adaptadores externos para ayudar en el proceso de alineación, por ejemplo, con un adaptador Pluggable Reward-Driven Contextual Adapter (PRCA) [45].

4.1.4 - Fase de Generación

El generador en un sistema RAG es el responsable de convertir la información recuperada en un texto coherente que formará el resultado final del modelo. Este proceso involucra diversos datos de entrada que a veces requieren trabajos para refinar la adaptación del modelo de lenguaje a los datos de entrada derivados de consultas y documentos. Esto se puede solucionar mediante el proceso de post-retrieval y fine-tuning:

- Post-retrieval con Frozen LLM: el procesamiento posterior a la recuperación deja el LLM intacto y, en cambio, se centra en mejorar la calidad de los resultados de la recuperación a través de operaciones como la compresión de información y la reclasificación de resultados. La compresión de información ayuda a reducir el ruido, abordar las restricciones de longitud del contexto de un LLM y mejora los efectos de generación. La reclasificación tiene como objetivo reordenar los documentos para priorizar los elementos más relevantes en la parte superior.
- Fine-tuning de LLM para RAG: para mejorar el sistema RAG, el generador se puede optimizar o ajustar aún más para garantizar que el texto generado sea natural y aproveche eficazmente los documentos recuperados.

4.1.5 - Fase de Aumento

El aumento es el proceso de integrar de manera efectiva el contexto de las partes de los documentos recuperados con la tarea de la generación actual.

La eficacia de un modelo RAG se ve muy afectada por la elección de la fuente de datos de aumento. Los datos se pueden clasificar en datos no estructurados, estructurados y generados por LLM.

En muchos procesos de aumento una sola recuperación no es suficiente, por lo que se han propuesto algunos métodos:

- La recuperación iterativa permite que el modelo realice múltiples ciclos de recuperación para mejorar la profundidad y relevancia de la información.
- La recuperación recursiva itera recursivamente en la salida de un paso de recuperación como entrada para otro paso de recuperación; esto permite profundizar en información relevante para consultas complejas y de varios pasos (por ejemplo, investigación académica y análisis de casos legales).
- La recuperación adaptativa transforma el proceso de recuperación a demandas específicas al determinar los momentos y el contenido óptimos para la recuperación.

4.2 – Implementación del sistema RAG

Para implementar un sistema RAG, al menos, se han de incluir los siguientes elementos:

- Modelo LLM: Será el encargado de analizar la pregunta y generar las respuestas.
- Técnica de Procesamiento de Lenguaje Natural a Vectores Matemáticos (Embedding): Para transformar los datos externos “en bruto” a formato entendible por el modelo [11].
- Base de Datos Vectorial: Para almacenar convenientemente los datos externos y una serie de comandos o preguntas [12].
- Interfaz para establecer el diálogo entre el sistema y el usuario.

Los pasos, de forma simplificada, serán los siguientes [13]:

- El usuario realiza la consulta, también denominada técnicamente prompt.
- El RAG se encarga de enriquecer ese prompt o pregunta con datos y hechos que ha obtenido de una base de datos externa que contiene información relevante relativa a la pregunta que ha realizado el usuario. A esta etapa se le denomina retrieval.
- El RAG es el encargado de enviar el prompt del usuario enriquecido o aumentado al LLM que se encarga de generar una respuesta en lenguaje natural aprovechando toda la potencia del lenguaje humano que ha aprendido con sus datos de entrenamiento genéricos, pero también con los datos específicos proporcionados en la etapa de retrieval.

Más en detalle, el prototipo desarrollado en Python[22] para implementar RAG sobre un documento, se implementa de la siguiente manera:

- Se cargan las librerías y herramientas necesarias.
- El sistema pregunta al usuario si desea hacer RAG o utilizar el chatboot con las respuestas que da el modelo LLM directamente.
- Se le ofrece al usuario la opción de elegir un modelo para trabajar entre los tres elegidos para este proyecto.
- Si el usuario ha seleccionado trabajar en modo RAG, se le pide que suba un documento en formato PDF para que éste sea analizado.
- El sistema lee el PDF y extrae el contenido de texto. Realiza un particionado o chunks del texto para que posteriormente pueda ser tratado. Se crean los metadatos y los embeddings y se almacena todo en la base de datos vectorial.
- A continuación, se genera el buffer de memoria para almacenar el contexto de la conversación.
- Posteriormente se crea el chat con el modelo seleccionado, la base de datos vectorial con los datos anteriormente descritos sobre el documento para hacer RAG y el buffer de memoria.
- El usuario puede realizar cualquier cuestión al modelo LLM y éste responderá basándose en el documento facilitado utilizando la técnica de RAG. También se le mostrará al usuario la fuente de dónde ha obtenido la respuesta en el documento.

- Si el usuario no ha seleccionado el modo RAG, simplemente se genera el buffer de memoria para almacenar el contexto de la conversación y se crea el chat con el modelo seleccionado. Las respuestas se generarán directamente por el modelo LLM.

Para la implementación práctica se han utilizado las siguientes tecnologías:

- Contenedor Docker [14]: Se ha implementado un contenedor con Ollama [15] para inferir el modelo LLM y para transformar los documentos externos en embeddings.
- Modelo LLM: Se han estudiado y comparado los siguientes modelos LLM:
 - LLAMA-3 [16] en sus versiones 8B y 70B
 - GEMMA-2 [17] en su versión 9B
 - PHI-3 [18] en su versión medium de 14B
- Modelo para Embedding: Se han estudiado y comparado los siguientes modelos:
 - Nomic-embed-text [19]
 - Mxbai-embed-large [20]
 - Snowflake-arctic-embed [21]
- Python [22]: Para el desarrollo del código fuente necesario para implementar el sistema RAG y para el interfaz de usuario.
- Módulo VENV de Python [23]: Para la creación de un entorno virtual específico para el proyecto.
- Chroma [24]: Base de datos vectorial para almacenar los embeddings.
- Framework LangChain [25]: Para el desarrollo de la aplicación en Python.
- Plataforma Google Colaboratory [46]: para la ejecución del cuaderno Jupyter [48] que calcula las métricas de los resultados obtenidos.
- Paquete Chainlit [26]: Para generar el interfaz de usuario en modo Chatbot.
- Librería PyPDF2 [27]: Para trabajar y transformar los ficheros PDF con los que el sistema RAG trabajará.
- Servidor con GPU [28]: Para la realización de este trabajo, el Grupo de Investigación Reconocido ECA-SIMM ha habilitado la máquina Beta, que es un servidor Ubuntu Linux 22.04 con procesador Intel(R) Xeon(R) Gold 6326 CPU @ 2.90Ghz con 8 cores

(4 núcleos x 2 socket), con 62Gb de memoria RAM y una GPU Nvidia A40 con 46Gb de memoria.

RETRIEVAL AUGMENTED GENERATION O RAG

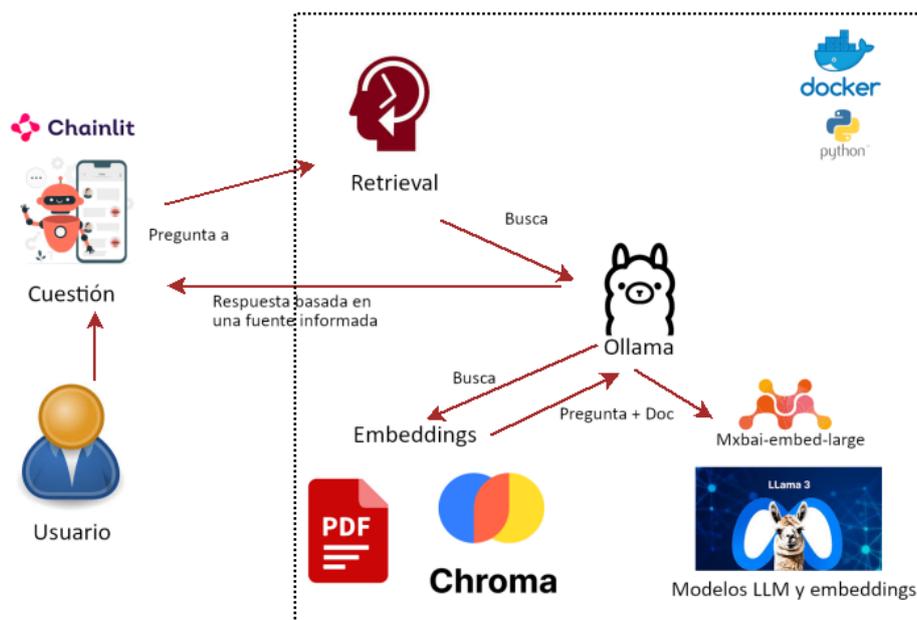


Figura 4.2 - Esquema de cómo se implementa el sistema RAG de forma práctica.

En el [apéndice A](#) se detalla la instalación y configuración de cada uno de los elementos descritos.

4.3 – Evaluación de calidad

4.3.1 - Métricas

Para evaluar las respuestas que devuelve el sistema a las preguntas que plantea el usuario se han usado las siguientes métricas:

- **ROUGE (Recall-Oriented Understudy for Gisting Evaluation)** [49]

Se utiliza para evaluar tanto la precisión como la exhaustividad de las respuestas generadas y es sensible al orden y la repetición de palabras.

Mide la superposición de n-grams entre la respuesta generada por el LLM y la de referencia o respuesta correcta. Evalúa tanto la exactitud (cuántos n-grams de la referencia están presentes en la respuesta generada) como la exhaustividad (cuántos n-grams de la respuesta generada coinciden con la referencia).

Tipos de ROUGE calculados:

- ROUGE-1: Mide la coincidencia de unigramas (palabras individuales) entre la respuesta generada y la de referencia.
- ROUGE-2: Mide la coincidencia de bigramas (pares consecutivos de palabras) entre ambas respuestas.
- ROUGE-L: Evalúa la longitud de la subsecuencia más larga de palabras coincidentes entre ambas respuestas, sin necesidad de que estén en la misma posición.

Rango: Varía entre 0 y 1, siendo 1 una coincidencia perfecta.

- **Precisión [50]**

La precisión es una métrica que evalúa qué porcentaje de las palabras generadas por el chatbot están presentes en la respuesta correcta.

Se calcula dividiendo el número de palabras correctamente generadas (palabras en común entre la respuesta generada y la de referencia) por el número total de palabras generadas por el chatbot.

Rango: Va de 0 a 1, donde 1 significa que todas las palabras generadas son correctas (presentes en la respuesta correcta).

- **Recall (Exhaustividad) [50]**

Mide qué porcentaje de las palabras de la respuesta correcta están presentes en la respuesta generada.

Se calcula dividiendo el número de palabras correctamente generadas por el número total de palabras que hay en la respuesta correcta.

Rango: Oscila entre 0 y 1, donde 1 significa que todas las palabras de la respuesta correcta están presentes en la respuesta generada.

- **F1-Score [50]**

Es una métrica que combina la precisión y el recall en un solo valor para equilibrar ambos factores. Es útil cuando se necesita tener en cuenta tanto la precisión como la exhaustividad.

El F1-Score es la media armónica de la precisión y el recall. Pondera los dos valores, dando un valor más bajo si uno de ellos es bajo, lo que obliga a que ambos valores sean altos para que el F1-Score también lo sea.

Fórmula:

$$F1 = 2 \times \left(\frac{\text{Precisión} \times \text{Recall}}{\text{Precisión} + \text{Recall}} \right)$$

Rango: Va de 0 a 1, donde 1 significa que tanto la precisión como el recall o exhaustividad son perfectos

- **Similitud coseno [50]**

La similitud coseno es una métrica que se utiliza para medir la similitud entre dos vectores de un espacio vectorial. En el contexto del procesamiento de lenguaje natural (NLP), los textos se convierten en vectores, y la similitud coseno mide cuán similares son esos textos en función del ángulo entre los vectores que los representan.

A diferencia de métricas como ROUGE o F1-Score, que se basan en coincidencias de palabras o secuencias de palabras, la similitud coseno mide el ángulo entre dos vectores en un espacio vectorial. Es útil porque evalúa cómo de alineados están dos textos en términos de contenido, sin importar la longitud o la escala.

Fórmula:

$$\text{Similitud Coseno } (A, B) = \frac{A \cdot B}{\|A\| \times \|B\|}$$

Dónde:

- A y B son los vectores que representan las respuestas.
- $A \cdot B$ es el producto punto de los dos vectores.
- $\|A\|$ y $\|B\|$ son las magnitudes de los vectores.

Su valor oscila entre -1 y 1:

- 1 significa que los vectores son idénticos o están alineados (máxima similitud).
- 0 significa que no tienen correlación alguna (vectores ortogonales).
- -1 indica que los vectores están completamente opuestos.

4.3.2 – Test de evaluación

Para realizar el test de evaluación del sistema y facilitar el cálculo de las métricas mencionadas en el anterior apartado, se ha implementado un cuaderno Jupyter [48] auxiliar donde se introducen los textos generados por el chatbot y el texto de referencia o respuesta correcta. Se adjunta el código fuente a esta memoria en el [Apéndice B5](#).

Se detallan las pruebas realizadas en los distintos modelos y los resultados y tiempos obtenidos, para ello se han utilizado la siguiente batería de preguntas.

Identificador	Pregunta
P1	¿Cuál es el precio de los taxis en la ciudad de Segovia?
P2	¿Se puede subir al Acueducto de Segovia?
P3	¿Qué oposiciones hay en el Ayuntamiento de Segovia?
P4	¿Qué partidos políticos componen la corporación actual del Ayuntamiento de Segovia?

Tabla 4.1 – Batería de preguntas realizadas al sistema

De la siguiente relación de ficheros se puede obtener el contexto para dar la mejor respuesta mediante las técnicas de RAG.

Identificador	Fichero que contiene el contexto para las respuestas	Tamaño Mb	Nº páginas
F1	AYS_PLN_20231124_ACT.pdf <i>(Acta del Pleno del Ayuntamiento de Segovia con las tasas de taxi)</i>	1.4	118
F2	Ordenanza de Convivencia Ciudadana 2022.pdf <i>(Ordenanza donde se reflejan las sanciones por actos incívicos en la ciudad de Segovia)</i>	0.6	28
F3	Publicación_BOP_SEGOVIA_78_28 Jun.pdf <i>(Boletín Oficial de la Provincia de Segovia donde se publica la oferta de empleo público del Ayuntamiento de Segovia)</i>	1.4	40
F4	AYS_PLN_20230707_ACT.pdf <i>(Acta del Pleno del Ayuntamiento de Segovia donde figura cómo se compone la corporación municipal)</i>	0.7	113

Tabla 4.2 – Ficheros que contienen el contexto para las respuestas

A continuación, se detallan las respuestas a las preguntas que figuran dentro de los documentos proporcionados al sistema para hacer RAG.

Identificador	Pregunta																										
R1	<p>TARIFAS 2024</p> <p>TARIFA 1.- Se aplicará esta tarifa todos los días del año de 07:00 horas a 23:00 horas excepto cuando sea de aplicación la tarifa 2</p> <table> <tr> <td>Bajada de Bandera</td> <td>1,90 €</td> </tr> <tr> <td>Percepción mínima.</td> <td>3,95 €</td> </tr> <tr> <td>Metros</td> <td>1.830,36</td> </tr> <tr> <td>Segundos</td> <td>359,12</td> </tr> <tr> <td>Kilómetro recorrido</td> <td>1,12 €</td> </tr> <tr> <td>Hora parada</td> <td>20,55 €</td> </tr> </table> <p>TARIFA 2.- Se aplicará esta tarifa en horario nocturno de 23:00 a 7:00 horas; domingos y festivos de 7:00 a 23:00 horas; periodo de feria y fiesta local de 15:00 a 23:00 horas (que comenzará a computarse desde el primer día de su inicio); así como sábados y días 24 y 31 de diciembre de 16:00 a 23:00 h.</p> <table> <tr> <td>BAJADA DE BANDERA</td> <td>2,65 €</td> </tr> <tr> <td>Percepción mínima.</td> <td>5,00 €</td> </tr> <tr> <td>Metros</td> <td>1.598,64</td> </tr> <tr> <td>Segundos</td> <td>293,24</td> </tr> <tr> <td>Kilómetro recorrido</td> <td>1,47 €</td> </tr> <tr> <td>Hora parada</td> <td>28,85 €</td> </tr> </table> <p>SUPLEMENTOS</p> <table> <tr> <td>Bulto o maleta</td> <td>0,65 €</td> </tr> </table>	Bajada de Bandera	1,90 €	Percepción mínima.	3,95 €	Metros	1.830,36	Segundos	359,12	Kilómetro recorrido	1,12 €	Hora parada	20,55 €	BAJADA DE BANDERA	2,65 €	Percepción mínima.	5,00 €	Metros	1.598,64	Segundos	293,24	Kilómetro recorrido	1,47 €	Hora parada	28,85 €	Bulto o maleta	0,65 €
Bajada de Bandera	1,90 €																										
Percepción mínima.	3,95 €																										
Metros	1.830,36																										
Segundos	359,12																										
Kilómetro recorrido	1,12 €																										
Hora parada	20,55 €																										
BAJADA DE BANDERA	2,65 €																										
Percepción mínima.	5,00 €																										
Metros	1.598,64																										
Segundos	293,24																										
Kilómetro recorrido	1,47 €																										
Hora parada	28,85 €																										
Bulto o maleta	0,65 €																										
R2	<p>No están permitidos los siguientes usos impropios en relación con el Acueducto de Segovia:</p> <p>Escalar, subir, trepar, descolgarse o cualquier otra acción similar en el Acueducto, sin autorización municipal, autorización que deberá estar basada en el interés público o por motivos de conservación o mantenimiento del monumento.</p>																										
R3	<p>Bases específicas y convocatoria para la cobertura en propiedad, mediante oposición libre, de ocho plazas de operarios de cometidos varios (3 reserva discapacidad) en el Ayuntamiento de Segovia, grupo e, escala administración especial, subescala servicios especiales, personal de oficios, pertenecientes a las ofertas públicas de empleo ordinarias de 2023 y 2024</p>																										

R4	<ul style="list-style-type: none"> - Concejales de la candidatura del Partido Popular (PP): 12 concejales. - Concejales de la candidatura del Partido Socialista Obrero Español (PSOE): 7 concejales. - Concejales de la candidatura del Partido Vox: 2 concejales. - Concejales de la candidatura del Partido Izquierda Unida (I.U.): 2 concejales. - Concejales de la candidatura del Partido Podemos – Alianza Verde (Podemos AV): 1 concejal. - Concejales de la candidatura del Partido Ciudadanos (Cs): 1 concejal
----	--

Tabla 4.3 – Respuestas correctas o esperadas

Para facilitar la lectura de esta memoria, en el [apéndice C](#) se pueden consultar las respuestas que ha dado el chatbot a cada una de las preguntas, con cada uno de los modelos estudiados y con el modo RAG inicialmente desactivado y posteriormente activado.

A continuación, se pueden ver los resultados de las métricas sin utilizar RAG

Modelo	ROUGE-1	ROUGE-2	ROUGE-L	Precisión	Recall	F1-Score	Similitud Coseno
Llama3	0.16	0.0	0.1	0.09	0.13	0.11	0.35
Llama3:70b	0.32	0.05	0.17	0.11	0.17	0.14	0.58
Gemma2	0.13	0.0	0.06	0.21	0.08	0.12	0.27
Phi3	0.22	0.01	0.13	0.15	0.14	0.14	0.36

Tabla 4.4 – Métricas de la P1 sin RAG

Modelo	ROUGE-1	ROUGE-2	ROUGE-L	Precisión	Recall	F1-Score	Similitud Coseno
Llama3	0.20	0.03	0.15	0.12	0.31	0.17	0.43
Llama3:70b	0.25	0.07	0.13	0.14	0.38	0.21	0.47
Gemma2	0.15	0.08	0.11	0.25	0.05	0.09	0.27
Phi3	0.27	0.05	0.15	0.16	0.23	0.19	0.39

Tabla 4.5 – Métricas de la P2 sin RAG

Modelo	ROUGE-1	ROUGE-2	ROUGE-L	Precisión	Recall	F1-Score	Similitud Coseno
Llama3	0.18	0.03	0.13	0.12	0.26	0.17	0.67
Llama3:70b	0.16	0.04	0.11	0.1	0.23	0.14	0.74
Gemma2	0.24	0.07	0.17	0.17	0.28	0.21	0.42
Phi3	0.24	0.04	0.12	0.13	0.33	0.19	0.51

Tabla 4.6 – Métricas de la P3 sin RAG

Modelo	ROUGE-1	ROUGE-2	ROUGE-L	Precisión	Recall	F1-Score	Similitud Coseno
Llama3	0.53	0.21	0.41	0.39	0.5	0.44	0.75
Llama3:70b	0.44	0.13	0.28	0.22	0.5	0.3	0.63
Gemma2	0.22	0.0	0.19	0.06	0.09	0.08	0.41
Phi3	0.16	0.05	0.14	0.1	0.41	0.16	0.32

Tabla 4.7 – Métricas de la P4 sin RAG

Ahora, se procede a mostrar las métricas de las respuestas a las mismas preguntas, pero utilizando RAG.

Modelo	ROUGE-1	ROUGE-2	ROUGE-L	Precisión	Recall	F1-Score	Similitud Coseno
Llama3	0.67	0.51	0.60	0.50	0.45	0.48	0.69
Llama3:70b	0.43	0.25	0.33	0.36	0.29	0.32	0.60
Gemma2	0.36	0.16	0.25	0.19	0.15	0.17	0.61
Phi3	0.60	0.52	0.54	0.38	0.59	0.46	0.73

Tabla 4.8 – Métricas de la P1 con RAG

Modelo	ROUGE-1	ROUGE-2	ROUGE-L	Precisión	Recall	F1-Score	Similitud Coseno
Llama3	0.66	0.46	0.60	0.54	0.51	0.53	0.72
Llama3:70b	0.67	0.46	0.57	0.63	0.49	0.55	0.73
Gemma2	0.28	0.14	0.24	0.45	0.13	0.20	0.42
Phi3	0.36	0.09	0.21	0.25	0.36	0.29	0.45

Tabla 4.9 – Métricas de la P2 con RAG

Modelo	ROUGE-1	ROUGE-2	ROUGE-L	Precisión	Recall	F1-Score	Similitud Coseno
Llama3	0.29	0.10	0.18	0.15	0.16	0.15	0.56
Llama3:70b	0.27	0.11	0.18	0.19	0.14	0.16	0.47
Gemma2	0.23	0.10	0.18	0.23	0.14	0.17	0.50
Phi3	0.20	0.05	0.13	0.10	0.26	0.15	0.56

Tabla 4.10 – Métricas de la P3 con RAG

Modelo	ROUGE-1	ROUGE-2	ROUGE-L	Precisión	Recall	F1-Score	Similitud Coseno
Llama3	0.57	0.40	0.50	0.45	0.84	0.59	0.61
Llama3:70b	0.56	0.38	0.50	0.41	0.69	0.51	0.67
Gemma2	0.45	0.25	0.41	0.50	0.50	0.50	0.46
Phi3	0.50	0.23	0.41	0.46	0.53	0.49	0.52

Tabla 4.11 – Métricas de la P4 con RAG

A continuación, se detallan los tiempos de respuesta del sistema.

Tiempo en contestar la respuesta sin RAG (segundos)				
Pregunta / Modelo	Llama3	Llama3:70b	Gemma2	Phi3
P1	6,75	28,07	0,94	5,81
P2	4,43	23,20	1,36	1,27
P3	4,59	40,49	2,56	14,80
P4	2,15	14,89	2,55	2,65

Tabla 4.12 – Tiempo en contestar la respuesta sin RAG

Tiempo en contestar la respuesta con RAG activado (segundos)				
Pregunta / Modelo	Llama3	Llama3:70b	Gemma2	Phi3
P1	6,4	32,68	8,71	9,65
P2	3,33	6,13	4,64	5,08
P3	1,86	35,43	6,88	13,08
P4	4,97	21,49	4,77	4,48

Tabla 4.13 – Tiempo en contestar la respuesta con RAG

Tiempo que tarda en procesar el fichero (segundos)			
Fichero / Modelo	Nomic-embed-text	Mxbai-embed-large	Snowflake-arctic-embed
F1	40.67	44.24	53.43
F2	8.62	11.41	14.37
F3	9.80	11.25	13.79
F4	34.36	38.08	39.11

Tabla 4.14 – Tiempo que tarda en procesar el fichero

Y, por último, el consumo de memoria de los diferentes modelos.

Consumo de memoria GPU de los modelos (MiB)	
Llama3	6.348
Llama3:70b	41.820
Gemma2	8.988
Phi3:medium	10.352
Nomic-embed-text	2.002
Mxbai-embed-large	1.404
Snowflake-arctic-embed	1.980

Tabla 4.15 – Consumo de memoria GPU de los modelos

5. Resultados obtenidos

En la evaluación de modelos de LLM se han hecho una serie de pruebas preliminares de su capacidad para generar respuestas coherentes y precisas en base a la información recuperada. De los tres modelos evaluados, el más fiable ha sido Llama3 en su versión de 70 billones de parámetros. El problema de utilizar este modelo reside en el requerimiento de GPU, con un número tan elevado de parámetros, las respuestas son más precisas, pero bastante más lentas que en la versión de 8 billones. Respecto al ajuste de hiperparámetros, sólo se ha modificado la temperatura del modelo, se ha fijado en 0.4, consiguiendo así un cierto grado de creatividad en las respuestas, pero manteniendo una base confiable. Cuanto más bajo sea el valor (cerca de 0), más conservador es el modelo y cuanto más alto (cerca de 1) más creativo e impredecible.

Para conocer cuál de los modelos es más preciso y coherente en sus respuestas, se ha procedido a realizar el cálculo de la media aritmética de las métricas de las respuestas a las 4 preguntas planteadas. Inicialmente se hace el análisis sin utilizar RAG.

Modelo	ROUGE-1	ROUGE-2	ROUGE-L	Precisión	Recall	F1-Score	Similitud Coseno
Llama3	0,27	0,07	0,20	0,18	0,30	0,22	0,55
Llama3:70b	0,29	0,07	0,17	0,14	0,32	0,20	0,61
Gemma2	0,19	0,04	0,13	0,17	0,13	0,13	0,34
Phi3	0,22	0,04	0,14	0,14	0,28	0,17	0,40

Tabla 5.1 – Media aritmética de métricas sin RAG

Para la métrica ROUGE-1, el modelo Llama3:70b es el que obtiene mayor valor (0.29), lo que indica una mayor coincidencia de unigramas con las secuencias de referencia. Respecto a ROUGE-2, tanto Llama3 como Llama3:70b tienen valores similares (0.07), lo que significa que la calidad en la coincidencia de bigramas es comparable entre ellos. Para ROUGE-L, que mide la coincidencia de la sub-secuencia más larga, una vez más, Llama3 (0.20) es el que obtiene mejores resultados.

En la métrica de Precisión, Llama 3 (0.18) y Gemma2, tienen valores similares, lo que indica que estos modelos generan una mayor proporción de secuencias correctas.

Para la métrica Recall, los modelos de Llama3:70b (0.32) y Llama3 (0.30) obtienen los mejores resultados, lo que sugiere que son capaces de capturar más palabras o secuencias correctas en comparación con los demás modelos.

Respecto a F1-Score, que combina la precisión y el recall, es Llama3 (0.22) la que tiene el mejor balance entre ambas, seguido de Llama3:70b (0.20). Esto significa que ambos modelos ofrecen un rendimiento más equilibrado en términos de generar secuencias correctas y captar suficientes detalles relevantes.

Por último, la métrica Similitud Coseno, que mide la similitud semántica entre las secuencias generadas y las de referencia, son los modelos Llama3:70b (0.61) y Llama3 (0.55) los que muestran los valores más altos, lo que indica que ambos son mejores para generar texto que es semánticamente similar al texto de referencia.

Los resultados muestran que Llama3:70b es el modelo que presenta el mejor rendimiento general, destacándose en las métricas ROUGE-1 y recall, además de obtener la mayor similitud coseno. Llama3 también ofrece un desempeño competitivo, particularmente en ROUGE-L, precisión y F1-Score. En cambio, Gemma2 tiene el peor rendimiento en la mayoría de las métricas, mostrando una baja precisión y similitud semántica. Phi3 ofrece un desempeño intermedio, pero en general se ubica por debajo de los modelos de la serie Llama3.

Posteriormente, se procede a hacer el cálculo de la media aritmética de las métricas de las respuestas a las 4 preguntas planteadas para conocer cuál de los modelos obtiene mejores resultados de forma global, esta vez empleando RAG.

Modelo	ROUGE-1	ROUGE-2	ROUGE-L	Precisión	Recall	F1-Score	Similitud Coseno
Llama3	0,55	0,37	0,47	0,41	0,49	0,44	0,65
Llama3:70b	0,48	0,30	0,40	0,40	0,40	0,39	0,62
Gemma2	0,33	0,16	0,27	0,34	0,23	0,26	0,50
Phi3	0,42	0,22	0,32	0,30	0,44	0,35	0,57

Tabla 5.2 – Media de las métricas con RAG

En la métrica ROUGE-1, es el modelo Llama3 el que obtiene el valor más alto (0.55), lo que indica que sus unigramas (palabras individuales) coinciden más con el texto de referencia en comparación con los demás modelos. Para ROUGE-2, el modelo de Llama3 sigue liderando con 0.37, seguido de Llama3:70b (0.30). Por último, para ROUGE-L, vuelve a ser Llama3 el modelo que tiene un claro liderazgo con un valor de 0.47, lo que indica que este modelo captura mejor las subsecuencias más largas que otros, lo que demuestra que los modelos de la serie Llama tienden a generar textos con una mayor estructura coherente.

Respecto a la Precisión, son los modelos Llama3 (0.41) y Llama3:70b (0.40) los que obtienen los mejores resultados, lo que significa que producen una mayor proporción de secuencias correctas con respecto a las palabras generadas.

Para Recall, Llama3 (0.49) vuelve a lidera en esta métrica, lo que indica que es capaz de recuperar la mayor parte de las palabras relevantes del texto de referencia. Phi3 (0.44) obtiene un buen valor, seguido por Llama3:70b (0.40), mientras que Gemma2 tiene el peor desempeño (0.23).

Analizando F1-Score, que combina la precisión y el recall para ofrecer una métrica equilibrada del rendimiento. Llama3 obtiene el mejor resultado (0.44), lo que demuestra su capacidad general para generar secuencias relevantes y correctas.

Para finalizar, en la métrica de Similitud Coseno, que evalúa si el significado del texto generado es coherente con el texto de referencia, es Llama3 (0.65) nuevamente la que destaca con el valor más alto, lo que indica que es el mejor modelo en generar texto que sea semánticamente cercano al de referencia.

Los resultados muestran que Llama3 es el modelo con el mejor rendimiento en todas las métricas clave, sobresaliendo tanto en la coincidencia n-grama (ROUGE) como en la precisión, recall y similitud semántica (similitud coseno). Llama3:70b también ofrece un desempeño robusto, aunque con un menor equilibrio entre precisión y recall, lo que podría indicar que genera secuencias más dispersas.

Phi3, por su parte, obtiene resultados moderados, destacándose principalmente en el recall, pero presentando un menor rendimiento en precisión y F1-Score, lo que indica una tendencia a generar más información relevante, aunque menos precisa. Gemma2 es el modelo con el peor desempeño general, con bajos valores en todas las métricas, particularmente en recall y ROUGE-2, lo que sugiere una menor capacidad para generar secuencias coherentes y estructuradas.

En conjunto, los modelos Llama son los más efectivos para tareas de generación de texto tanto cuando se incorpora la técnica de RAG como cuando no se incorpora.

Para los modelos de embeddings se han hecho pruebas de funcionamiento de los tres que tiene Ollama, el más rápido es nomic-embed-text, pero también es el menos preciso. Se ha optado por usar mxbai-embed-large que está en el punto medio en rapidez y precisión. El tiempo en transformar un documento de 100 páginas aproximadamente es de 40 segundos.

También se han evaluado los tiempos de respuesta de los distintos modelos, los tiempos que tarda en procesar el documento y el consumo de memoria GPU, se puede observar en las tablas 4.12, 4.13, 4.14 y 4.15 respectivamente.

Es el modelo de Gemma2 es el más eficiente en términos de tiempo, mientras que Llama3:70 es el más lento. La alta complejidad de este modelo es la causa de que sus tiempos de respuesta sean más largos ya que requiere una mayor demanda de procesamiento computacional.

Analizando todos los resultados en conjunto, sin duda, los modelos de LLAMA3 han demostrado ser los mejores entre los analizados, tanto en precisión y coherencia en sus respuestas, como en la similitud a los textos de referencia que contienen los documentos analizados.

5.1 – Demostración del sistema

Se ha desarrollado un interfaz de usuario con la ayuda del framework Chainlit, a modo de chatbot o sistema conversacional, en el que se puede seleccionar entre modo normal o modo RAG. El código fuente se puede consultar en el [apéndice A](#).

Lo primero que pregunta el sistema al usuario es si desea utilizar el modo RAG o no. Si selecciona que no y le realiza una pregunta muy concreta de un tema específico, por ejemplo, el precio de las tarifas de taxis en la ciudad de Segovia, el resultado es el siguiente:

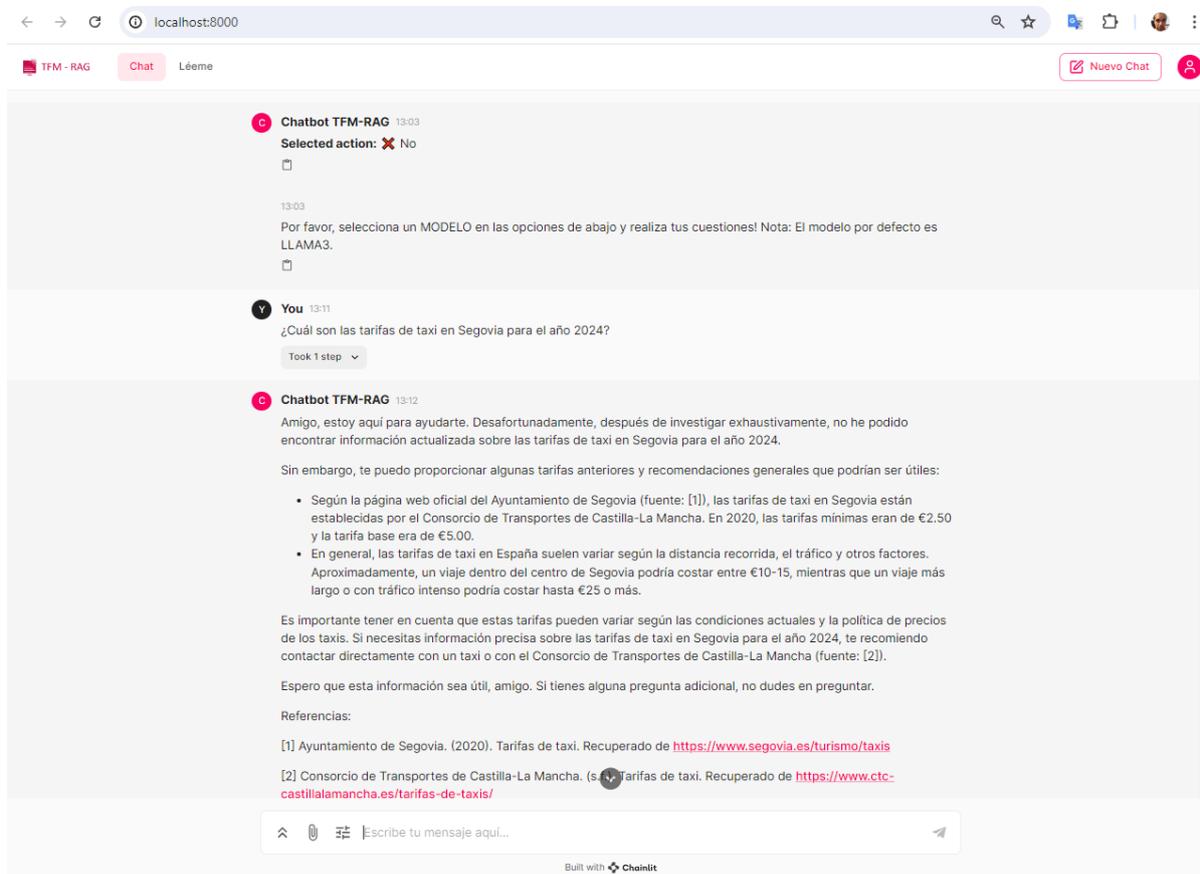
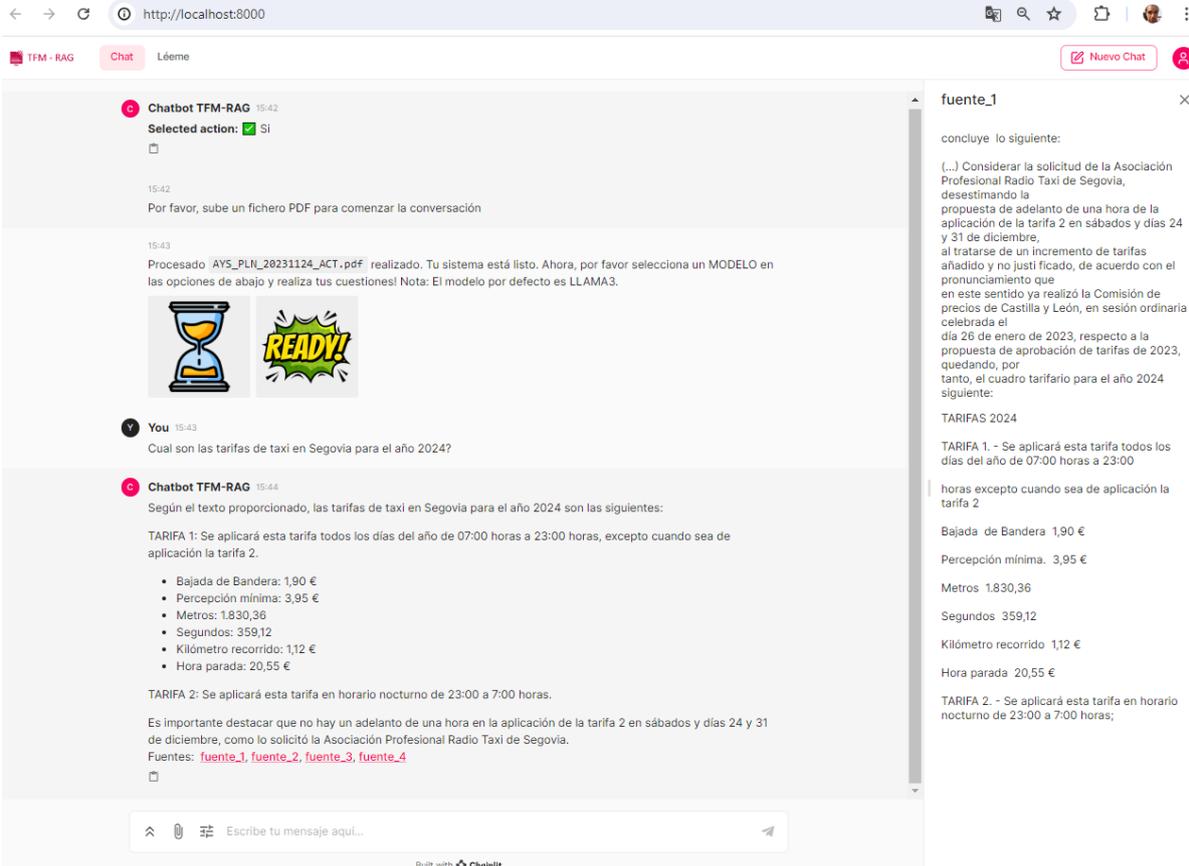


Figura 5.1 - Interfaz de usuario. Pregunta al sistema con el modo RAG desactivado.

Si se escoge el modo RAG, el sistema le pide al usuario que envíe un documento en formato PDF para transformarlo y enviárselo al modelo para aumentar y generar la respuesta más precisa basando su respuesta en el contenido del documento.

Se sube el acta del pleno del Ayuntamiento de Segovia donde se aprobaron las tarifas de taxi para el año 2024 y se realiza la misma cuestión.



The screenshot shows a web browser window with a chat interface. The browser address bar shows 'http://localhost:8000'. The chat interface has a header with 'TFM - RAG', 'Chat', and 'Léeme'. A 'Nuevo Chat' button is visible in the top right. The chat history shows:

- Chatbot TFM-RAG (15:42):** Selected action: Si. Por favor, sube un fichero PDF para comenzar la conversación.
- Chatbot TFM-RAG (15:43):** Procesado AYS_PLIN_20231124_ACT.pdf realizado. Tu sistema está listo. Ahora, por favor selecciona un MODELO en las opciones de abajo y realiza tus cuestiones! Nota: El modelo por defecto es LLAMA3. (Includes an hourglass icon and a 'READY!' speech bubble icon).
- You (15:43):** Cual son las tarifas de taxi en Segovia para el año 2024?
- Chatbot TFM-RAG (15:44):** Según el texto proporcionado, las tarifas de taxi en Segovia para el año 2024 son las siguientes:
 - TARIFA 1: Se aplicará esta tarifa todos los días del año de 07:00 horas a 23:00 horas, excepto cuando sea de aplicación la tarifa 2.
 - Bajada de Bandera: 1,90 €
 - Percepción mínima: 3,95 €
 - Metros: 1.830,36
 - Segundos: 359,12
 - Kilómetro recorrido: 1,12 €
 - Hora parada: 20,55 €
 - TARIFA 2: Se aplicará esta tarifa en horario nocturno de 23:00 a 7:00 horas.

Es importante destacar que no hay un adelanto de una hora en la aplicación de la tarifa 2 en sábados y días 24 y 31 de diciembre, como lo solicitó la Asociación Profesional Radio Taxi de Segovia.

Fuentes: [fuente_1](#), [fuente_2](#), [fuente_3](#), [fuente_4](#)

The right sidebar shows a source window titled 'fuente_1' with the following text:

concluye lo siguiente:

(...) Considerar la solicitud de la Asociación Profesional Radio Taxi de Segovia, desestimando la propuesta de adelanto de una hora de la aplicación de la tarifa 2 en sábados y días 24 y 31 de diciembre, al tratarse de un incremento de tarifas añadido y no justificado, de acuerdo con el pronunciamiento que en este sentido ya realizó la Comisión de precios de Castilla y León, en sesión ordinaria celebrada el día 26 de enero de 2023, respecto a la propuesta de aprobación de tarifas de 2023, quedando, por tanto, el cuadro tarifario para el año 2024 siguiente:

TARIFAS 2024

TARIFA 1. - Se aplicará esta tarifa todos los días del año de 07:00 horas a 23:00 horas excepto cuando sea de aplicación la tarifa 2

- Bajada de Bandera 1,90 €
- Percepción mínima. 3,95 €
- Metros 1.830,36
- Segundos 359,12
- Kilómetro recorrido 1,12 €
- Hora parada 20,55 €

TARIFA 2. - Se aplicará esta tarifa en horario nocturno de 23:00 a 7:00 horas;

Figura 5.2 - Interfaz de usuario. Pregunta al sistema con el modo RAG activado.

Haciendo la misma pregunta al sistema alternando entre activar y desactivar el modo RAG se puede comprobar el funcionamiento de la técnica de RAG para la generación de respuestas del modelo LLM

6. Discusión y conclusiones

A lo largo de este proyecto, se ha demostrado la capacidad de los sistemas de Generación Aumentada por Recuperación (RAG) para mejorar la extracción y generación de documentos, en general en cualquier entorno y específicamente en entornos de gestión pública. La implementación de estos sistemas ha permitido no solo obtener respuestas más precisas y adaptadas al contexto, sino también superar algunas limitaciones inherentes a los Modelos de Lenguaje Grande (LLM), como la tendencia a "alucinar" cuando se enfrentan a consultas específicas.

En cuanto a los resultados obtenidos, el análisis de las métricas de rendimiento de los distintos modelos ha demostrado que Llama3 en su versión de 70 billones de parámetros ofrece las mejores respuestas tanto en precisión como en similitud semántica, aunque con una clara desventaja en términos de tiempo de procesamiento debido a la alta demanda de recursos computacionales. En cambio, modelos como Gemma2 y Phi3 presentaron un comportamiento más eficiente en términos de tiempo, pero con una precisión y coherencia menores.

Esto plantea una interesante discusión sobre el equilibrio entre rendimiento y eficiencia en la implementación de RAG. Se podría solventar si los recursos hardware fueran ilimitados, sobre todo los recursos computacionales, como la GPU, pero la escalabilidad requiere un mayor coste económico que hay que estar dispuesto a asumir. En el [apéndice D](#) se muestra un presupuesto de lo que podría costar, aproximadamente, implementar este proyecto en una Administración Pública.

Un aspecto relevante a destacar es la capacidad de las técnicas de RAG para enriquecer el contexto de las consultas del usuario, lo que se reflejó en las métricas, que fueron significativamente superiores cuando se activó esta técnica en comparación con los resultados obtenidos sin su uso.

En este trabajo se ha logrado desarrollar e implementar un sistema RAG efectivo que mejora los procesos de extracción y generación de información en entornos de gestión documental pública. El sistema desarrollado ha cumplido con los objetivos propuestos, proporcionando una solución viable para la automatización de tareas relacionadas con la gestión de documentos, con lo que se consigue una mayor eficiencia administrativa y una mejora en la calidad de los servicios prestados.

Los resultados obtenidos muestran que los modelos Llama3 y el uso de la técnica RAG son la mejor opción para la generación de respuestas más precisas y contextualizadas. A su vez, el uso de herramientas como Ollama, Docker, y LangChain ha facilitado la implementación de un sistema modular y escalable, lo que permite su adaptación futura a entornos más complejos o con mayor volumen de documentos.

Para un trabajo futuro, se plantea explorar las siguientes líneas de trabajo:

- Optimización de recursos: Investigar cómo implementar el sistema en entornos con menor capacidad computacional, por ejemplo, mediante la utilización de técnicas de compresión de modelos o la distribución de la carga de procesamiento.
- Aumento de la capacidad de recuperación: Mejorar la fase de recuperación y alineación de documentos, quizá incorporando técnicas más avanzadas como la búsqueda híbrida o la recuperación recursiva.
- Procesamiento en segundo plano: Implementar un modo off-line donde el procesamiento de grandes volúmenes de documentos se realice como tarea en segundo plano, permitiendo una interacción más fluida entre el usuario y el sistema sin tiempos de espera prolongados.

Para concluir, este trabajo ha sentado las bases para la implementación de sistemas avanzados de generación aumentada en entornos públicos, abriendo nuevas oportunidades para la automatización y mejora de procesos en la gestión documental, con el potencial de escalar a aplicaciones más amplias en otros sectores.

Apéndice

Apéndice A

Instalación y configuración del sistema

Todo el sistema se ha instalado y configurado en el servidor BETA del Grupo de Investigación Reconocido ECA-SIMM, ya que posee una potente GPU [21], necesaria para que el sistema funcione de forma fluida. Es un servidor Ubuntu Linux 22.04 con procesador Intel(R) Xeon(R) Gold 6326 CPU @ 2.90Ghz con 8 cores (4 núcleos x 2 socket), con 62Gb de memoria RAM y una GPU Nvidia A40 con 46Gb de memoria.

A.1 - Docker [14]

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

Se creará un contenedor configurado con una imagen de Ollama, que se explicará en el siguiente apartado, así se consigue que el sistema esté aislado del resto de software del equipo.

Instalación de Docker en Ubuntu Linux [29]

1. Configurar el repositorio APT para Docker.

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
```

```
echo \  
  "deb [arch=$(dpkg --print-architecture) signed-  
by=/etc/apt/keyrings/docker.asc]  
https://download.docker.com/linux/ubuntu \  
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \  
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null  
sudo apt-get update
```

2. Instalar la última versión de los paquetes de Docker.

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-  
buildx-plugin docker-compose-plugin
```

3. Verificar que la instalación de Docker está correcta ejecutando la imagen de hello-world.

```
sudo docker run hello-world
```

Este comando descarga una imagen de prueba y la ejecuta en un contenedor. Cuando el contenedor se ejecuta, imprime un mensaje de confirmación y se cierra.

A.2 - Ollama [15]

Ollama es una herramienta de código abierto que permite ejecutar modelos de lenguaje en equipos de forma local y de manera sencilla. Su principal propósito es facilitar el acceso a modelos de gran tamaño sin necesidad de utilizar la nube. Con Ollama, se puede descargar y ejecutar multitud de modelos, incluyendo LLaMA-3, Gemma-2, Phi-3 y Mistral, entre otros. Esta herramienta es especialmente útil para explorar y experimentar con modelos de lenguaje sin complicaciones.

Instalación de Ollama en un contenedor Docker [30]

1. Para poder utilizar los beneficios de la GPU del servidor, se debe instalar y configurar la herramienta específica de NVIDIA para contenedores Docker [31].

1. Configurar el repositorio.

```
curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey
| sudo gpg --dearmor -o /usr/share/keyrings/nvidia-container-
toolkit-keyring.gpg \
  && curl -s -L https://nvidia.github.io/libnvidia-
container/stable/deb/nvidia-container-toolkit.list | \
  sed 's#deb https://#deb [signed-
by=/usr/share/keyrings/nvidia-container-toolkit-keyring.gpg]
https://#g' | \
  sudo tee /etc/apt/sources.list.d/nvidia-container-
toolkit.list
```

2. Actualizar la lista de paquetes desde el repositorio.

```
sudo apt-get update
```

3. Instalar el paquete de NVIDIA Container ToolKit.

```
sudo apt-get install -y nvidia-container-toolkit
```

4. Es necesario configurar el runtime de Docker para que use el runtime para contenedores de NVIDIA, para ello se ejecuta el comando nvidia-ctk.

```
sudo nvidia-ctk runtime configure --runtime=docker
```

5. Por último, se reinicia el servicio de Docker.

```
sudo systemctl restart docker
```

2. Ejecutar Ollama dentro de un contenedor Docker.

```
docker run -d --gpus=all -v ollama:/root/.ollama -p 11434:11434 --name
ollama ollama/ollama
```

Levanta un contenedor de nombre ollama en el puerto tcp 11434, lo que nos permite realizar llamadas externas al contenedor.

3. Ejecutar un modelo dentro del contenedor.

```
docker exec -it ollama ollama run llama3
```

La primera vez que se ejecuta un modelo, Ollama lo descarga en el contenedor.

4. Para saber el identificador del contenedor de Ollama, se ejecuta el comando `docker ps`.

```
macoll@beta:~$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED
0a6c4a3ae0b5	ollama/ollama	"/bin/ollama serve"	46 hours ago
Up 46 hours	0.0.0.0:11434->11434/tcp	ollama	

Con este comando se obtiene información del contenedor, como el identificador, puertos tcp, nombre, etc.

5. Si se quieren descargar más modelos para tenerlos accesibles es necesario ejecutar el comando `pull`.

```
macoll@beta:~$ docker exec 0a6c4a3ae0b5 ollama pull gemma2
pulling manifest
pulling ffd1d1fc78170... 100% ██████████ 5.4 GB
pulling 109037bec39c... 100% ██████████ 136 B
pulling 097a36493f71... 100% ██████████ 8.4 KB
pulling 2490e7468436... 100% ██████████ 65 B
pulling 10aa81da732e... 100% ██████████ 487 B
verifying sha256 digest
writing manifest
removing any unused layers
success
```

6. Si se quiere saber qué modelos están descargados en el docker, es necesario ejecutar el comando *list*.

```
macoll@beta:~$ docker exec 0a6c4a3ae0b5 ollama list
NAME                                ID                                SIZE    MODIFIED
gemma2:latest                       ff02c3702f32                    5.4 GB  21 minutes ago
llama3:70b                           be39eb53a197                    39 GB   2 months ago
llama3:latest                        a6990ed6be41                    4.7 GB  2 months ago
llama3-chatqa:latest                 b37a98d204b2                    4.7 GB  2 days ago
mxbai-embed-large:latest             468836162de7                    669 MB  7 weeks ago
nomic-embed-text:latest              0a109f422b47                    274 MB  8 weeks ago
phi3:medium                          1e67dff39209                    7.9 GB  2 days ago
snowflake-arctic-embed:latest        21ab8b9b0545                    669 MB  5 weeks ago
```

7. Si se quiere borrar un modelo se ejecuta el comando *rm*.

```
macoll@beta:~$ docker exec 0a6c4a3ae0b5 ollama rm llama3-chatqa
deleted 'llama3-chatqa'
```

8. Parar y arrancar el contenedor de Ollama.

```
macoll@beta:~$ docker stop 0a6c4a3ae0b5
0a6c4a3ae0b5
macoll@beta:~$ docker start 0a6c4a3ae0b5
0a6c4a3ae0b5
```

9. Si se quiere borrar el contenedor se ejecuta el siguiente comando.

```
macoll@beta:~$ docker rm 0a6c4a3ae0b5
0a6c4a3ae0b5
```

10. Para comprobar que se está utilizando la GPU y cuánta memoria, se ejecuta el siguiente comando.

```
macoll@beta:~$ nvidia-smi
Thu Jul 11 10:09:53 2024
```

NVIDIA-SMI 535.183.01 Driver Version: 535.183.01 CUDA Version: 12.2										
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr. ECC				
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute M.				MIG M.
0	NVIDIA A40	Off	00000000:06:10.0	Off		0				
0%	37C	P0	95W / 300W	1210MiB / 46068MiB	13%	Default				N/A


```
Processes:
```

GPU	GI	CI	PID	Type	Process name	GPU Memory Usage
ID	ID	ID				
0	N/A	N/A	328311	C	...unners/cuda_v11/ollama_llama_server	1202MiB

Despliegue de los modelos necesarios en el contenedor Docker con Ollama [32]

Se necesitan desplegar los siguientes modelos LLM:

- LLAMA-3 [16] en sus versiones 8B y 70B

```
docker exec 0a6c4a3ae0b5 ollama pull llama3
```

```
docker exec 0a6c4a3ae0b5 ollama pull llama3:70b
```

- GEMMA-2 [17] en su versión 9B

```
docker exec 0a6c4a3ae0b5 ollama pull gemma2
```

- PHI-3 [18] en su versión medium de 14B

```
docker exec 0a6c4a3ae0b5 ollama pull phi3:medium
```

Se necesitan desplegar los siguientes modelos para realizar los Embedding:

- Nomic-embed-text [19]

```
docker exec 0a6c4a3ae0b5 ollama pull nomic-embed-text
```

- Mxbai-embed-large [20]

```
docker exec 0a6c4a3ae0b5 ollama pull mxbai-embed-large
```

- Snowflake-arctic-embed [21]

```
docker exec 0a6c4a3ae0b5 ollama pull snowflake-arctic-embed
```

A.3 - Python y módulo VENV de Python [22] [23]

Python es un lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.

Se utilizará Python para el desarrollo del código fuente necesario para implementar el sistema RAG y para el interfaz de usuario.

El módulo VENV de Python permite la creación de "entornos virtuales" ligeros, cada uno con su propio conjunto independiente de paquetes Python instalados en los directorios del sitio. Un entorno virtual se crea sobre una instalación de Python existente, conocida como Python "base" del entorno virtual, y opcionalmente puede aislarse de los paquetes en el entorno base, de modo que solo estén disponibles aquellos instalados explícitamente en el entorno virtual. De esta forma se podrá trabajar en el proyecto con todas los paquetes y librerías necesarias sin que esto afecte a la instalación "base" de Python instalada en el servidor BETA de ECA-SIMM.

Instalación de Python en Ubuntu [33]

Para instalar Python3 en Ubuntu 22.04 se ejecutan los siguientes comandos.

1. Actualizar la lista de paquetes desde el repositorio.

```
sudo apt-get update
```

2. Instalar el paquete de Python3.

```
sudo apt-get install python3
```

3. Verificar instalación

```
python3 --version
```

Creación de entorno virtual VENV [23]

Para crear el entorno virtual de Python con el que se trabajará se ejecuta el siguiente comando.

```
python3 -m venv /home/data/macoll/rag/frontend
```

Para activar y desactivar el entorno virtual se ejecutan estos comandos.

```
macoll@beta:~/data/rag/frontend$ source bin/activate  
(frontend) macoll@beta:~/data/rag/frontend$
```

```
(frontend) macoll@beta:~/data/rag/frontend$ deactivate
```

A.4 - Librerías y paquetes Python necesarios

- **Chainlit** [26]

Chainlit es un paquete Python de código abierto para crear chatbot de IA conversacional listos para producción. Entre sus características destaca por su facilidad para crear rápidamente, es multiplataforma, tiene persistencia de dato y se integra perfectamente con OpenAI, Llama Index o LangChain entre otros.

Para instalar Chainlit es necesario ejecutar el siguiente comando.

```
(frontend) macoll@beta:~/data/rag/frontend$ pip install chainlit
```

Una vez instalado, crea una estructura de carpetas y ficheros como la siguiente.

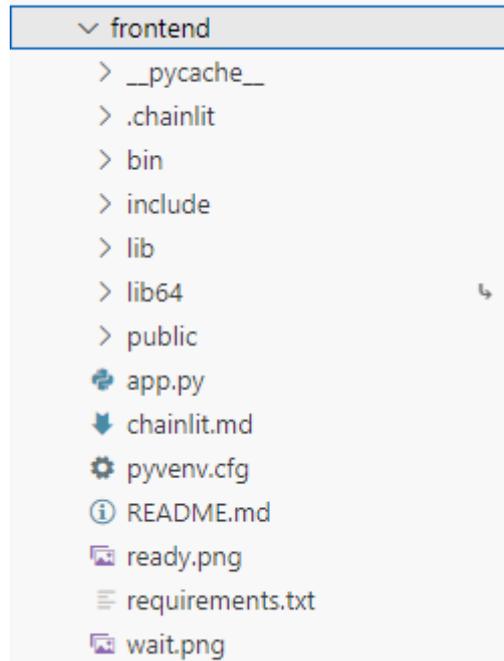


Figura A.1 - Estructura de carpetas y ficheros de chainlit

- `__pycache__` : contiene la caché de la aplicación.
- `.chainlit`: contiene el fichero de traducciones de idioma y el fichero `config.toml`, que alberga distintos parámetros de configuración.
- `public`: contiene logotipos, imágenes y hojas de estilo `css`.
- `app.py`: código fuente de la aplicación.
- `chainlit.md`: contiene el texto informativo de la aplicación.

Para ejecutar la aplicación se realiza mediante el siguiente comando.

```
(frontend) macoll@beta:~/data/rag/frontend$ chainlit run -w app.py
```

- **Framework LangChain [25]**

LangChain es un framework para desarrollar aplicaciones sobre grandes modelos de lenguaje (LLM). LangChain simplifica cada etapa del ciclo de vida de la aplicación LLM.

Para instalar LangChain y las librerías necesarias se ejecutan los siguientes comandos.

```
(frontend) macoll@beta:~/data/rag/frontend$ pip install langchain  
  
(frontend) macoll@beta:~/data/rag/frontend$ pip install langchain_community
```

- **Librería PyPDF2 [27]**

PyPDF2 es una biblioteca de PDF de Python gratuita y de código abierto capaz de dividir, fusionar, recortar y transformar las páginas de archivos PDF. También puede agregar datos personalizados, recuperar texto y metadatos, opciones de visualización y contraseñas a archivos PDF.

Para instalar PyPDF2 se ejecuta el siguiente comando.

```
(frontend) macoll@beta:~/data/rag/frontend$ pip install PyPDF2
```

- **Base de datos vectorial Chroma [24]**

Chroma es la base de datos vectorial de código abierto nativa de IA. Chroma facilita la creación de aplicaciones LLM al hacer que los conocimientos, los hechos y las habilidades se puedan conectar a los LLM. Chroma facilita las herramientas para almacenar los embeddings y sus metadatos, insertar documentos y consultas, así como los embeddings de búsqueda. Chroma prioriza la simplicidad y productividad del desarrollado.

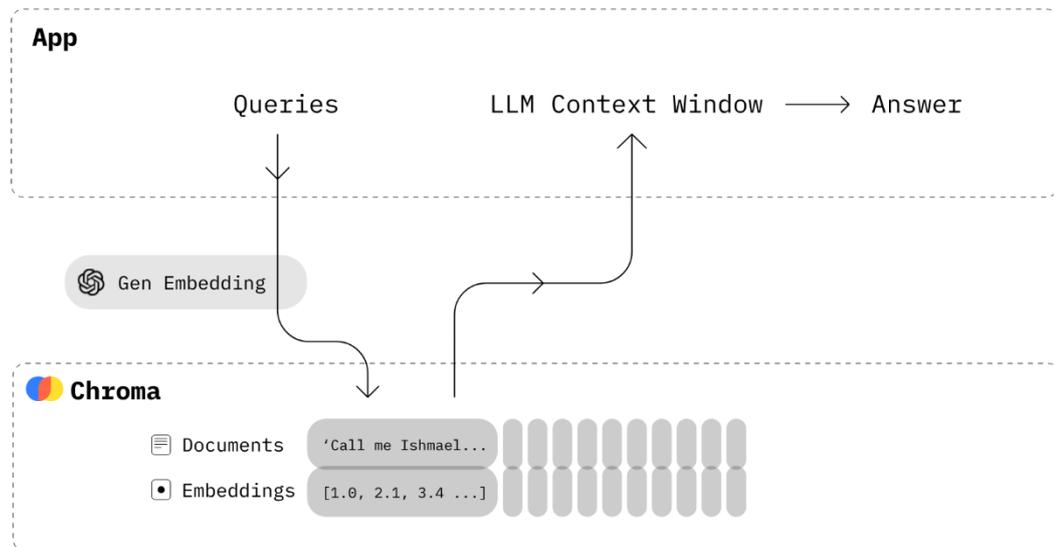


Figura A.2 - Diagrama de base de datos Chroma [24]

Para instalar la base de datos vectorial Chroma es necesario ejecutar el siguiente comando.

```
(frontend) macoll@beta:~/data/rag/frontend$ pip install chromadb
```

Apéndice B

Código fuente del sistema

Además de en esta memoria, se puede consultar en el siguiente repositorio de GitHub:

<https://github.com/boni554/tfm-rag>

B.1 - Fichero app.py (aplicación principal)

```
#####  
#           Máster en Ingeniería Informática - Universidad de Valladolid           #  
#                                                                                   #  
#                   Trabajo Fin de Máster                                         #  
#                                                                                   #  
#   Implementación de técnicas de RAG (Retrieval Augmented Generation)           #  
#   sobre LLM (Large Language Models) para la extracción y generación           #  
#                   de documentos en las Entidades Públicas                       #  
#                                                                                   #  
#                   Realizado por Miguel Ángel Collado Alonso                     #  
#                                                                                   #  
#####  
  
# Import necesarios para el proyecto  
  
from langchain_community.embeddings import OllamaEmbeddings  
from langchain_community.vectorstores import Chroma  
from langchain_community.chat_models import ChatOllama  
from langchain.text_splitter import RecursiveCharacterTextSplitter  
from langchain.chains import ConversationalRetrievalChain  
from langchain.memory import ChatMessageHistory, ConversationBufferMemory  
from langchain_community.llms import Ollama  
from langchain.prompts import ChatPromptTemplate  
from langchain.schema import StrOutputParser  
from langchain.schema.runnable import Runnable  
from langchain.schema.runnable.config import RunnableConfig
```

```

import chainlit as cl
from chainlit.input_widget import Select

import time

import PyPDF2

# Variable que actúa a modo de switch para activar o no el modo RAG
RAG = False

# Función que se ejecuta al principio de abrir el chat

@cl.on_chat_start
async def on_chat_start():

    global RAG

    # Pregunta al usuario si quiere usar RAG o no

    res = await cl.AskActionMessage(
        content="""Hola, soy el asistente virtual de TFM-RAG.
        ¿Deseas utilizar RAG?""",
        actions=[
            cl.Action(name="si", value="Si", label="✓ Si"),
            cl.Action(name="no", value="No", label="✗ No"),
        ],
    ).send()

    if res and res.get("value") == "Si":
        RAG = True
    else:
        RAG = False
        await cl.Message(
            content="Por favor, selecciona un MODELO en las opciones de
            abajo y realiza tus cuestiones! Nota: El modelo por defecto es LLAMA3.",
        ).send()

    # Define la configuración del modelo LLM, permite seleccionar entre los
    modelos Llama3, Gemma2
    # o Phi3. Por defecto está seleccionado Llama3.

    settings = await cl.ChatSettings(
        [
            Select(
                id="Model",
                label="LLM - Modelo",
                values=["llama3", "llama3:70b", "gemma2", "phi3:medium"],
                initial_index=0,
            )
        ]
    ).send()
    selected_model = settings["Model"]

```

```

# Código que se ejecuta si el modo RAG está activado

if RAG:

    # Le pide al usuario que suba un fichero para hacer RAG

    files = None

    # Espera que el usuario suba el fichero

    while files is None:
        files = await cl.AskFileMessage(
            content="Por favor, sube un fichero PDF para comenzar la
conversación",
            accept=["application/pdf"],
            max_size_mb=60, # Tamaño máximo de fichero
            timeout=180, # Define el timeout
            type="assistant_message",
        ).send()

        # La variable file almacena el fichero subido por el usuario
        file = files[0]

        # Inicio del temporizador
        start_time_rag = time.time()

        # Configura las imágenes que se visualizan en el chat mientras
espera o cuando está listo
        elements1 = [
            cl.Image(name="image", display="inline",
path="wait.png",size="small")
        ]
        elements2 = [
            cl.Image(name="image", display="inline", path="ready.png",
size="small")
        ]

        # Informa al usuario de que se está procesando el fichero
        msg = cl.Message(content=f"Procesando
`{file.name}`...",type="user_message", elements=elements1)
        await msg.send()

        # Lectura y extracción del texto del fichero
        pdf = PyPDF2.PdfReader(file.path)
        pdf_text = ""
        for page in pdf.pages:
            pdf_text += page.extract_text()

        # Particionado del fichero en chunk de tamaño 1024
        text_splitter = RecursiveCharacterTextSplitter(chunk_size=1024,
chunk_overlap=50)

```

```

texts = text_splitter.split_text(pdf_text)

# Creación de metadatos para cada pedazo de fichero o chunk
metadatos = [{"source": f"{i}-pl"} for i in range(len(texts))]

# Creación de los Embeddings y base de datos de vectores en Chroma

# Selección de modelo de embeddings con el que se desea trabajar
# embeddings = OllamaEmbeddings(model="nomic-embed-text")
embeddings = OllamaEmbeddings(model="mxbai-embed-large")
# embeddings = OllamaEmbeddings(model="snowflake-arctic-embed")

# Creación de la base de datos con el texto, los embeddings y los
metadatos
docsearch = await cl.make_async(Chroma.from_texts)(
    texts, embeddings, metadatos=metadatos
)

# Inicializa el historial de mensajes de la conversación
message_history = ChatMessageHistory()

# Buffer de memoria para almacenar el contexto de la conversación
memory = ConversationBufferMemory(
    memory_key="chat_history",
    output_key="answer",
    chat_memory=message_history,
    return_messages=True,
)

# Crea el chat con el modelo seleccionado, el documento para hacer
RAG y el buffer de memoria
chain = ConversationalRetrievalChain.from_llm(
    ChatOllama(
        model=selected_model, # selección de modelo
        temperature=0.4, # parámetro temperatura para ajustar la
creatividad
        system="You are an expert research assistant who will
analyze, correlate, and extract relevant information from the given context
and answer questions asked by the user. Your output should be precise and
accurate with source information. You will always speak in the Spanish
language."
    ),
    chain_type="stuff",
    retriever=docsearch.as_retriever(),
    memory=memory,
    return_source_documents=True,
)

# Fin del temporizador
end_time_rag = time.time()

# Calcular el tiempo transcurrido
elapsed_time_rag = end_time_rag - start_time_rag

```

```

        # Informa al usuario de que el sistema está listo
        msg.content = f"Procesado `{file.name}` realizado en
`{elapsed_time_rag:.2f}` segundos. Ahora, por favor selecciona un MODELO en
las opciones de abajo y realiza tus cuestiones! Nota: El modelo por defecto
es LLAMA3."
        msg.elements = elements2
        await msg.update()

        # Almacena el chat en la sesión de usuario
        cl.user_session.set("chain", chain)

    else:

        # Código que se ejecuta si el modo RAG está desactivado

        # Selección de modelo LLM
        model = Ollama(model=selected_model,
                       temperature=0.4) # parámetro temperatura para
ajustar la creatividad

        # Crea el prompt indicándole al modelo cómo debe comportarse
        prompt = ChatPromptTemplate.from_messages(
            [
                (
                    "system",
                    "You are a very knowledgeable robot that provides
precise and eloquent answers to any type of question. Your output should be
precise and accurate with source information. You will always speak in the
Spanish language.",
                ),
                ("human", "{question}"),
            ]
        )

        # Ejecuta el chat
        runnable = prompt | model | StrOutputParser()

        # Almacena el chat en la sesión de usuario
        cl.user_session.set("runnable", runnable)

# Función que se ejecuta cuando se recibe un mensaje del usuario

@cl.on_message
async def main(message: cl.Message):

    global RAG

    # Código que se ejecuta si el modo RAG está activado

    if RAG:

        # Recupera el chat de la sesión de usuario

```

```

chain = cl.user_session.get("chain")

# Inicializa el controlador de devolución de llamada
cb = cl.LangchainCallbackHandler()

# Inicio del temporizador
start_time_chat = time.time()

# Llama al chat con el contenido del mensaje del usuario, añadiendo
que responda en español
res = await chain.ainvoke(message.content+". Responde en español,
incluso si no encuentras respuesta.", callbacks=[cb])
answer = res["answer"]
source_documents = res["source_documents"]

text_elements = [] # Variable para almacenar el texto de las
Fuentes del documento

# Procesado del documento para referenciar las fuentes
if source_documents:
    for source_idx, source_doc in enumerate(source_documents):
        source_name = f"fuentes_{source_idx+1}"
        # Create the text element referenced in the message
        text_elements.append(
            cl.Text(content=source_doc.page_content,
name=source_name)
        )
    source_names = [text_el.name for text_el in text_elements]

# Añade las fuentes a la respuesta del chat
if source_names:
    answer += f"\nFuentes: {', '.join(source_names)}"
else:
    answer += "\nNo se ha encontrado fuentes"

# Fin del temporizador
end_time_chat = time.time()

# Calcular el tiempo transcurrido
elapsed_time_chat = end_time_chat - start_time_chat
answer += f"\n Tiempo en responder `{elapsed_time_chat:.2f}`
segundos"

# Devuelve los resultados
await cl.Message(content=answer, elements=text_elements).send()

else:

# Código que se ejecuta si el modo RAG está desactivado

```

```

# Inicio del temporizador
start_time_chat = time.time()

# Recupera el chat de la sesión de usuario
runnable = cl.user_session.get("runnable")

# Inicializa el mensaje
msg = cl.Message(content="")

# Llama al chat con el contenido del mensaje del usuario, añadiendo
que responda en español
    async for chunk in runnable.astream(
        {"question": message.content+" Responde en español, incluso si
no encuentras respuesta."},
config=RunnableConfig(callbacks=[cl.LangchainCallbackHandler()]),
    ):
        await msg.stream_token(chunk)

# Fin del temporizador
end_time_chat = time.time()

# Calcular el tiempo transcurrido
elapsed_time_chat = end_time_chat - start_time_chat
msg.content += f"\nTiempo en responder: `{elapsed_time_chat:.2f}`
segundos"

# Devuelve los resultados
await msg.send()

```

B.2 - Fichero chainlit.md (texto del Readme o Léeme)

Trabajo Fin de Máster - Universidad de Valladolid

Implementación de técnicas de RAG (Retrieval Augmented Generation) sobre LLM (Large Language Models) para la extracción y generación de documentos en las Entidades Públicas

****Realizado por Miguel Ángel Collado Alonso****

Un viejo desafío en el campo de la inteligencia artificial es “enseñar” a las máquinas a entender cómo hablan y escriben los humanos, conocido como procesamiento de lenguaje natural (PLN). Sin embargo, desde hace poco más de dos años, estamos asistiendo a la caída de este antiguo bastión con la llegada de los modelos grandes del lenguaje (LLM) y los interfaces conversacionales.

Los LLM son modelos de inteligencia artificial que se entrenan utilizando algoritmos de Deep Learning sobre conjuntos enormes de información generada por humanos. De esta manera, una vez entrenados, han aprendido la forma en la que los humanos utilizamos la palabra hablada y escrita, así que son capaces de ofrecernos respuestas generales y con un patrón muy parecido a nuestra forma de contestar a las preguntas que les hacemos. Sin embargo, si buscamos respuestas precisas en un contexto determinado, los LLM por sí solos no proporcionarán respuestas concretas o habrá una alta probabilidad de que se inventen completamente la respuesta.

En este trabajo, se va a explicar y desarrollar una de las técnicas clave que hace posible que estos sistemas nos respondan con relativa precisión a las preguntas que les hacemos, esta tecnología se denomina Generación Aumentada por Recuperación o RAG, del inglés Retrieval Augmented Generation.

B.3 - Fichero config.toml (parámetros de configuración y estilo)

```
[project]
# Whether to enable telemetry (default: true). No personal data is collected.
enable_telemetry = true

# List of environment variables to be provided by each user to use the app.
user_env = []

# Duration (in seconds) during which the session is saved when the connection
is lost
session_timeout = 3600

# Enable third parties caching (e.g LangChain cache)
cache = false

# Authorized origins
allow_origins = ["*"]

# Follow symlink for asset mount (see
https://github.com/Chainlit/chainlit/issues/317)
# follow_symlink = false

[features]
# Show the prompt playground
```

```
prompt_playground = true

# Process and display HTML in messages. This can be a security risk (see
# https://stackoverflow.com/questions/19603097/why-is-it-dangerous-to-render-
# user-generated-html-or-javascript)
unsafe_allow_html = false

# Process and display mathematical expressions. This can clash with "$"
# characters in messages.
latex = false

# Authorize users to upload files with messages
multi_modal = true

# Allows user to use speech to text
[features.speech_to_text]
  enabled = false
  # See all languages here https://github.com/JamesBrill/react-speech-
  # recognition/blob/HEAD/docs/API.md#language-string
  # language = "en-US"

[UI]
# Name of the app and chatbot.
name = "Chatbot TFM-RAG"

# Show the readme while the thread is empty.
show_readme_as_default = true

# Description of the app and chatbot. This is used for HTML tags.
# description = ""

# Large size content are by default collapsed for a cleaner ui
default_collapse_content = true

# The default value for the expand messages settings.
default_expand_messages = false

# Hide the chain of thought details from the user in the UI.
hide_cot = false

# Link to your github repo. This will add a github button in the UI's header.
# github = ""
```

```
# Specify a CSS file that can be used to customize the user interface.
# The CSS file can be served from the public directory or via an external
link.
# custom_css = "/public/test.css"

# Specify a custom font url.
# custom_font =
"https://fonts.googleapis.com/css2?family=Inter:wght@400;500;700&display=swap"

# Override default MUI light theme. (Check theme.ts)
[UI.theme]
  #font_family = "Inter, sans-serif"
[UI.theme.light]
  #background = "#FAFAFA"
  #paper = "#FFFFFF"

  [UI.theme.light.primary]
    #main = "#F80061"
    #dark = "#980039"
    #light = "#FFE7EB"

# Override default MUI dark theme. (Check theme.ts)
[UI.theme.dark]
  #background = "#FAFAFA"
  #paper = "#FFFFFF"

  [UI.theme.dark.primary]
    #main = "#F80061"
    #dark = "#980039"
    #light = "#FFE7EB"

[meta]
generated_by = "1.0.200"
```

B.4 - Fichero es-Es.json (traducción al castellano)

```
{
  "components": {
    "atoms": {
      "buttons": {
```

```

        "userButton": {
            "menu": {
                "settings": "Configuración",
                "settingsKey": "C",
                "APIKeys": "API Keys",
                "logout": "Logout"
            }
        }
    },
    "molecules": {
        "newChatButton": {
            "newChat": "Nuevo Chat"
        },
        "tasklist": {
            "TaskList": {
                "title": "\ud83d\udd22\ufe0f Lista de tareas",
                "loading": "Cargando...",
                "error": "Ha ocurrido un error"
            }
        },
        "attachments": {
            "cancelUpload": "Cancelar subida",
            "removeAttachment": "Borrar adjunto"
        },
        "newChatDialog": {
            "createNewChat": "Crear nuevo chat?",
            "clearChat": "Se borrarán los mensajes actuales y comenzará un
nuevo chat.",
            "cancel": "Cancelar",
            "confirm": "Confirmar"
        },
        "settingsModal": {
            "expandMessages": "Ampliar mensajes",
            "hideChainOfThought": "Hide Chain of Thought",
            "darkMode": "Modo oscuro"
        }
    },
    "organisms": {
        "chat": {
            "history": {
                "index": {

```

```

        "lastInputs": "Últimas entradas",
        "noInputs": "Vacío...",
        "loading": "Cargando..."
    }
},
"inputBox": {
    "input": {
        "placeholder": "Escribe tu mensaje aquí..."
    },
    "speechButton": {
        "start": "Comenzar grabación",
        "stop": "Parar grabación"
    },
    "SubmitButton": {
        "sendMessage": "Enviar mensaje",
        "stopTask": "Parar tarea"
    },
    "UploadButton": {
        "attachFiles": "Adjuntar ficheros"
    },
    "waterMark": {
        "text": "Built with"
    }
},
"Messages": {
    "index": {
        "running": "Ejecutándose",
        "executedSuccessfully": "Ejecucción satisfactoria",
        "failed": "Fallido",
        "feedbackUpdated": "Feedback actualizado",
        "updating": "Actualizando"
    }
},
"dropScreen": {
    "dropYourFilesHere": "Deja tus archivos aquí"
},
"index": {
    "failedToUpload": "Fallo en la carga",
    "cancelledUploadOf": "Cancelada la carga de",
    "couldNotReachServer": "No se pudo llegar al servidor",
    "continuingChat": "Continuando el chat anterior"
},

```

```

    "settings": {
      "settingsPanel": "Panel de configuración",
      "reset": "Resetear",
      "cancel": "Cancelar",
      "confirm": "Confirmar"
    }
  },
  "threadHistory": {
    "sidebar": {
      "filters": {
        "FeedbackSelect": {
          "feedbackAll": "Feedback: All",
          "feedbackPositive": "Feedback: Positive",
          "feedbackNegative": "Feedback: Negative"
        },
        "SearchBar": {
          "search": "Buscar"
        }
      },
      "DeleteThreadButton": {
        "confirmMessage": "Esto eliminará el hilo, así como sus
mensajes y elementos.",
        "cancel": "Cancelar",
        "confirm": "Confirmar",
        "deletingChat": "Borrando chat",
        "chatDeleted": "Chat borrado"
      },
      "index": {
        "pastChats": "Pega Chats"
      },
      "ThreadList": {
        "empty": "Vacio..."
      },
      "TriggerButton": {
        "closeSidebar": "Cerrar sidebar",
        "openSidebar": "Abrir sidebar"
      }
    },
    "Thread": {
      "backToChat": "Volver al chat",
      "chatCreatedOn": "Este chat fue creado el"
    }
  }
}

```

```
    },
    "header": {
      "chat": "Chat",
      "readme": "Léeme"
    }
  },
  "hooks": {
    "useLLMProviders": {
      "failedToFetchProviders": "No se pudieron recuperar los
proveedores:"
    }
  },
  "pages": {
    "Design": {},
    "Env": {
      "savedSuccessfully": "Guardado con éxito",
      "requiredApiKeys": "Requiere API Keys",
      "requiredApiKeysInfo": "Para utilizar esta aplicación, se requieren
las siguientes claves API. Las claves se almacenan en el almacenamiento local
de su dispositivo."
    },
    "Page": {
      "notPartOfProject": "No eres parte de este proyecto."
    },
    "ResumeButton": {
      "resumeChat": "Reaudar Chat"
    }
  }
}
```

B.5 - Código auxiliar del Cuaderno Jupyter para el cálculo de las métricas.

```
#####
#       Máster en Ingeniería Informática - Universidad de Valladolid       #
#                                                                                   #
#               Trabajo Fin de Máster                                             #
#                                                                                   #
#   Implementación de técnicas de RAG (Retrieval Augmented Generation)         #
#   sobre LLM (Large Language Models) para la extracción y generación         #
#   de documentos en las Entidades Públicas                                     #
#                                                                                   #
#           Cuaderno Jupyter auxiliar para el cálculo de métricas             #
#                                                                                   #
#           Realizado por Miguel Ángel Collado Alonso                           #
#                                                                                   #
#####

# Instalamos rouge_score si no estuviera ya instalado
!pip install rouge_score

# Importamos las librerías necesarias
from nltk.translate.bleu_score import sentence_bleu
from rouge_score import rouge_scorer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import CountVectorizer
import re
import unicodedata
import numpy as np
import ipywidgets as widgets
from IPython.display import display
import pandas as pd

# Función para preprocesar el texto
def preprocesar_texto(texto):

    # Eliminar saltos de línea y caracteres especiales
    texto_limpio = texto.replace("\n", " ").strip() # Elimina saltos de
línea y espacios adicionales
    # Normaliza el texto para separar los caracteres base de los acentos
    texto_limpio = unicodedata.normalize('NFKD', texto_limpio)
    # Filtra solo los caracteres que no son marcas de acento
    texto_limpio = ''.join(c for c in texto_limpio if not
unicodedata.combining(c))
    texto_limpio = re.sub(r'\s+', ' ', texto_limpio) # Reemplaza múltiples
espacios por uno solo
    return texto_limpio

# Función para obtener las respuestas a través de un área de texto
def obtener_respuestas():
```

```

# Crea dos áreas de texto grandes para que el usuario introduzca el
texto del chatbot y la respuesta correcta
chatbot_area = widgets.Textarea(
    value='',
    placeholder='Introduce la respuesta generada por el chatbot
aquí...',
    description='Chatbot:',
    layout=widgets.Layout(width='100%', height='200px')
)

referencia_area = widgets.Textarea(
    value='',
    placeholder='Introduce la respuesta de referencia aquí...',
    description='Referencia:',
    layout=widgets.Layout(width='100%', height='200px')
)

# Botón para confirmar la entrada
button = widgets.Button(description="Calcular Métricas")

# Muestra las áreas de texto
display(chatbot_area, referencia_area, button)

# Función para manejar el evento del botón
def on_button_clicked(b):
    chatbot_respuesta = preprocesar_texto(chatbot_area.value)
    respuesta_correcta = preprocesar_texto(referencia_area.value)
    calcular_metricas(chatbot_respuesta, respuesta_correcta)

# Asocia el clic del botón con la función
button.on_click(on_button_clicked)

# Función para calcular ROUGE
def calcular_rouge(chatbot_respuesta, respuesta_correcta):
    scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'],
use_stemmer=True)
    scores = scorer.score(respuesta_correcta, chatbot_respuesta)
    return scores

# Función para calcular precisión, recall y F1 considerando diferencias de
longitud
def calcular_exactitud_f1(chatbot_respuesta, respuesta_correcta):
    palabras_chatbot = set(chatbot_respuesta.split())
    palabras_referencia = set(respuesta_correcta.split())

    # Intersección entre las palabras de ambas respuestas (palabras
comunes)
    palabras_comunes = palabras_chatbot.intersection(palabras_referencia)

    # Cálculo de precisión, recall y F1
    if len(palabras_chatbot) > 0 and len(palabras_referencia) > 0:
        precision = len(palabras_comunes) / len(palabras_chatbot)
        recall = len(palabras_comunes) / len(palabras_referencia)

```

```

        if precision + recall > 0:
            f1 = 2 * (precision * recall) / (precision + recall)
        else:
            f1 = 0.0
    else:
        precision, recall, f1 = 0.0, 0.0, 0.0

    return precision, recall, f1

# Función para calcular similitud coseno
def calcular_similitud_coseno(chatbot_respuesta, respuesta_correcta):

    # Usa CountVectorizer para convertir el texto en vectores
    vectorizador = CountVectorizer().fit_transform([chatbot_respuesta,
respuesta_correcta])
    vectores = vectorizador.toarray()

    # Calcula la similitud coseno entre los dos vectores
    similitud = cosine_similarity([vectores[0]], [vectores[1]])

    return similitud[0][0]

# Función principal que calcula las métricas y las muestra en formato de
tabla
def calcular_metricas(chatbot_respuesta, respuesta_correcta):

    # Calcula ROUGE
    rouge_scores = calcular_rouge(chatbot_respuesta, respuesta_correcta)

    # Calcula precisión y F1 para respuestas exactas
    precision, recall, f1 = calcular_exactitud_f1(chatbot_respuesta,
respuesta_correcta)

    # Calcula similitud coseno
    similitud_coseno = calcular_similitud_coseno(chatbot_respuesta,
respuesta_correcta)

    # Muestra las métricas en formato de tabla con pandas
    metricas = {
        "ROUGE-1": [round(rouge_scores["rouge1"].fmeasure, 2)],
        "ROUGE-2": [round(rouge_scores["rouge2"].fmeasure, 2)],
        "ROUGE-L": [round(rouge_scores["rougeL"].fmeasure, 2)],
        "Precisión": [round(precision, 2)],
        "Recall": [round(recall, 2)],
        "F1-Score": [round(f1, 2)],
        "Similitud Coseno": [round(similitud_coseno, 2)]
    }

    df_metricas = pd.DataFrame(metricas)
    display(df_metricas)

# Llama a la función para obtener respuestas
obtener_respuestas()

```

Apéndice C

**Respuestas del sistema a las
preguntas planteadas**

Se han obtenido las siguientes respuestas del sistema sin utilizar RAG.

Sin RAG - P1- ¿Cuál es el precio de los taxis en la ciudad de Segovia?

Modelo	Respuesta
Llama3	<p>Amigo, estoy aquí para ayudarte. Desafortunadamente, no encontré información precisa sobre el precio de los taxis en la ciudad de Segovia, ya que cada empresa de transporte público puede tener sus propias tarifas y políticas.</p> <p>Sin embargo, puedo proporcionarte algunas sugerencias para encontrar la respuesta que buscas:</p> <p>La página web oficial del Ayuntamiento de Segovia no tiene información específica sobre el precio de los taxis en la ciudad. El sitio web de la Asociación Española de Empresarios de Taxi (AEET) no proporciona información detallada sobre las tarifas de taxis en Segovia, pero ofrece consejos generales para elegir un taxi seguro y confiable.</p> <p>Algunos sitios web de viajes y guías turísticas como TripAdvisor o Lonely Planet pueden tener información general sobre el precio de los taxis en Segovia, pero no es información actualizada ni precisa. Si deseas obtener información más específica sobre el precio de los taxis en la ciudad de Segovia, te recomiendo contactar con empresas de transporte público locales o consultando con un agente de viajes que pueda brindarte información actualizada y precisa.</p> <p>Recuerda que mi objetivo es proporcionarte respuestas precisas y fiables, por lo que si no puedo encontrar la respuesta que buscas, te diré. ¡Espero que esta información te sea útil!</p>
Llama3:70b	<p>Después de realizar una búsqueda exhaustiva, encontré que el precio de los taxis en la ciudad de Segovia puede variar dependiendo del tipo de servicio y la zona donde se solicite. Sin embargo, según la información disponible en la página web oficial de la Asociación de Taxistas de Segovia (ATS), los precios aproximados para un viaje en taxi en Segovia son:</p> <p>Tarifa 1 (lunes a viernes, de 7:00 a 21:00): €2.40 de bandeja + €0.85/km</p> <p>Tarifa 2 (lunes a jueves, de 21:00 a 7:00; y festivos): €3.20 de bandeja + €1.05/km</p> <p>Tarifa 3 (viernes y sábados, de 21:00 a 7:00): €4.00 de bandeja + €1.25/km</p>

	Es importante tener en cuenta que estos precios pueden variar dependiendo del tráfico, la hora del día y otros factores. Además, es posible que algunos taxis cobren suplementos adicionales por servicios como el uso de tarjeta de crédito o el transporte de equipaje.
Gemma2	Desafortunadamente, no tengo acceso a información en tiempo real como precios de servicios como los taxis. Te recomiendo que consultes sitios web especializados en transporte o aplicaciones móviles de taxis para obtener tarifas actualizadas para la ciudad de Segovia.
Phi3	El costo del servicio de taxi en la ciudad de Segovia cuesta generalmente entre €1,50 y €2 por kilómetro durante el día (dependiendo de factores como el tipo de vehículo y las tarifas vigentes). Sin embargo, es importante tener en cuenta que estas cifras pueden variar según los cambios en la política municipal o la empresa especial cuya prestación se está considerando. Para obtener información más actualizada sobre las tarifas de taxi en Segovia, recomiendo consultar directamente con una compañía local de taxis o visitar el sitio web oficial del Ayuntamiento de Segovia donde pueden publicarse los precios vigentes y cualquier otra información relevante.

Tabla C.1 – Respuestas a la P1 sin RAG

Sin RAG - P2 - ¿Se puede subir al Acueducto de Segovia?

Modelo	Respuesta
Llama3	No se recomienda ni es posible subirse a la estructura del Acueducto de Segovia por razones de seguridad y conservación patrimonial. El acueducto forma parte del Patrimonio Mundial declarado por la UNESCO, y está protegido por leyes españolas que prohíben cualquier tipo de daño o alteración al monumento. En cuanto a las razones específicas para no subirse: El acueducto es una estructura antigua y frágil, construida hace más de dos mil años con técnicas constructivas que hoy en día son consideradas como ingeniosas pero también delicadas. Por lo tanto, cualquier peso adicional podría comprometer su integridad estructural. El acueducto es un símbolo importante de la historia y cultura española, por ello está protegido por leyes que prohíben cualquier tipo de daño o alteración al monumento para preservarlo para las generaciones futuras.

	<p>La seguridad personal también es una preocupación en este caso: subirse a la estructura podría ser peligroso y resultar en lesiones personales graves.</p>
Llama3:70b	<p>Desafortunadamente, no es posible subir al Acueducto de Segovia, ya que se trata de un monumento histórico y patrimonio de la humanidad declarado por la UNESCO en 1985.</p> <p>Según la página web oficial del Ayuntamiento de Segovia, el acceso al interior del acueducto no está permitido para proteger su integridad y evitar daños a este valioso patrimonio cultural. Además, la estructura del acueducto no está diseñada para soportar el peso de las personas, lo que podría generar riesgos para los visitantes.</p> <p>Sin embargo, es posible disfrutar de la belleza y majestuosidad del Acueducto de Segovia desde fuera, ya sea caminando por sus alrededores o visitando el Museo del Acueducto, que se encuentra en las cercanías. Allí, los visitantes pueden obtener información detallada sobre la historia y construcción de este impresionante monumento romano.</p>
Gemma2	<p>Sí, se puede subir al Acueducto de Segovia.</p> <p>El acueducto está abierto al público y cuenta con una plataforma para visitantes en la parte superior donde se pueden disfrutar vistas panorámicas de la ciudad.</p> <p>No hay información disponible sobre restricciones o horarios específicos para el acceso. Te recomiendo consultar el sitio web oficial del Ayuntamiento de Segovia o contactar directamente con ellos para obtener información actualizada sobre las visitas al Acueducto.</p>
Phi3	<p>No se permite el acceso público a las partes superiores del Acueducto de Segovia para proteger su estructura y conservarlo como patrimonio histórico. Sin embargo, es posible disfrutar de una visita guiada que ofrece un recorrido por la base y los alrededores del acueducto.</p>

Tabla C.2 – Respuestas a la P2 sin RAG

Sin RAG - P3 - ¿Qué oposiciones hay en el Ayuntamiento de Segovia?

Modelo	Respuesta
Llama3	<p>Según la página web oficial del Ayuntamiento de Segovia, existen varias oposiciones abiertas para diferentes áreas y departamentos. A continuación, te presento algunas de las oportunidades laborales más recientes:</p>

	<p>Oposición a Jefe de Servicio de Urbanismo: La convocatoria se publicó en el Boletín Oficial del Estado (BOE) núm. 251 de fecha 22 de octubre de 2020.</p> <p>Oposición a Técnico Superior de Gestión Administrativa: La convocatoria se publicó en el BOE núm. 234 de fecha 15 de octubre de 2020.</p> <p>Oposición a Auxiliar de Servicio Social: La convocatoria se publicó en la Gaceta Municipal núm. 144 de fecha 29 de septiembre de 2020.</p> <p>Oposición a Técnico Superior de Mantenimiento: La convocatoria se publicó en el BOE núm. 227 de fecha 8 de octubre de 2020.</p> <p>Es importante mencionar que estas oposiciones pueden estar sujetas a cambios o nuevas convocatorias, por lo que te recomiendo verificar la información en la página web oficial del Ayuntamiento de Segovia o contactar directamente con ellos para obtener la información más actualizada.</p>
<p>Llama3:70b</p>	<p>Según la página web oficial del Ayuntamiento de Segovia, actualmente hay varias oposiciones abiertas para diferentes puestos de trabajo en el ayuntamiento. A continuación, te proporciono algunas de las oposiciones que se encuentran disponibles:</p> <p>Técnico/a de Administración: El plazo de presentación de solicitudes finalizó el 24 de febrero de 2023. (Fuente: Boletín Oficial de la Provincia de Segovia, núm. 34, de 17 de febrero de 2023)</p> <p>Agente de Policía Local: El plazo de presentación de solicitudes finalizó el 10 de marzo de 2023. (Fuente: Boletín Oficial de la Provincia de Segovia, núm. 45, de 24 de febrero de 2023)</p> <p>Técnico/a de Urbanismo: El plazo de presentación de solicitudes finalizó el 17 de marzo de 2023. (Fuente: Boletín Oficial de la Provincia de Segovia, núm. 51, de 10 de marzo de 2023)</p> <p>Es importante tener en cuenta que los plazos de presentación de solicitudes pueden variar, por lo que te recomiendo visitar la página web oficial del Ayuntamiento de Segovia o contactar con ellos directamente para obtener información actualizada y precisa sobre las oposiciones disponibles.</p> <p>Si no encuentro respuesta a tu pregunta en fuentes oficiales, siempre te proporciono una respuesta clara y transparente indicando que no he podido encontrar la información solicitada. En este caso, he podido proporcionarte algunas de las oposiciones abiertas en el Ayuntamiento de Segovia.</p>
<p>Gemma2</p>	<p>Para obtener información precisa sobre las oposiciones disponibles en el Ayuntamiento de Segovia, te recomiendo que consultes directamente</p>

	<p>la página web oficial del Ayuntamiento o que te pongas en contacto con su departamento de Recursos Humanos.</p> <p>En la página web del Ayuntamiento de Segovia, generalmente encontrarás una sección dedicada a empleo o ofertas de trabajo donde se publican las últimas convocatorias para oposiciones. También puedes buscar información en portales especializados en empleo público, como el portal www.infoempleo.es.</p> <p>Recuerda que la información sobre oposiciones puede cambiar con frecuencia, por lo que es importante consultar fuentes oficiales y actualizadas.</p>
<p>Phi3</p>	<p>En el Ayuntamiento de Segovia cuentan con varias oposiciones para distintas áreas funcionales. Estas son algunas de las más comunes:</p> <p>Oposición a Concejal del Área de Urbanismo, Ordenación Territorial y Medio Ambiente (AOTMA): Esta oposición es para aquellos interesados en participar en la planificación urbanística y medioambiental de Segovia.</p> <p>Oposición a Teniente de Alcalde: Este puesto se refiere al segundo cargo más alto del Ayuntamiento, que actúa como sustituto del Alcalde.</p> <p>Oposiciones técnicas en áreas específicas: Estas oposiciones son para aquellos con experiencia y formación en campos especializados como sanidad pública, educación o servicios sociales.</p> <p>Es importante destacar que las fechas de presentación y los requisitos específicos pueden variar según la convocatoria anual del Ayuntamiento. Para obtener información actualizada sobre las oposiciones disponibles en Segovia, se recomienda visitar el sitio web oficial del Ayuntamiento o contactar directamente con su oficina de personal y recursos humanos.</p>

Tabla C.3 – Respuestas a la P3 sin RAG

Sin RAG - P4 - ¿Qué partidos políticos componen la corporación actual del Ayuntamiento de Segovia?

Modelo	Respuesta
<p>Llama3</p>	<p>La corporación actual del Ayuntamiento de Segovia se compone de los siguientes partidos políticos:</p>

	<p>Partido Popular (PP): 12 concejales Partido Socialista Obrero Español (PSOE): 8 concejales Ciudadanos (Cs): 4 concejales Vox: 3 concejales Podemos-Izquierda Unida-La Fuente (Podemos-IU-LF): 2 concejales Partido Animalista Contra el Maltrato Animal (PACMA): 1 concejal</p>
<p>Llama3:70b</p>	<p>Según la información disponible en el sitio web oficial del Ayuntamiento de Segovia y otras fuentes confiables, la composición política actual del pleno del Ayuntamiento de Segovia es la siguiente:</p> <p>Partido Popular (PP): 11 concejales Partido Socialista Obrero Español (PSOE): 6 concejales Ciudadanos-Partido de la Ciudadanía (Cs): 3 concejales Vox: 2 concejales Unidas Podemos-Izquierda Unida (UP-IU): 1 concejal Fuente: Ayuntamiento de Segovia, Boletín Oficial del Estado (BOE) y otras fuentes periodísticas.</p> <p>Es importante destacar que la composición política puede variar en función de las elecciones municipales celebradas cada cuatro años. La información proporcionada es válida a partir de las últimas elecciones municipales celebradas en 2019.</p>
<p>Gemma2</p>	<p>De acuerdo. Para obtener una lista precisa y actualizada de los partidos políticos que componen la corporación actual del Ayuntamiento de Segovia, te recomiendo consultar el sitio web oficial del Ayuntamiento de Segovia o fuentes de información política confiables como el Ministerio del Interior de España.</p> <p>En estos sitios encontrarás la información más precisa y reciente sobre la composición del Ayuntamiento.</p>
<p>Phi3</p>	<p>Los partidos políticos que forman parte de la composición actual del Ayuntamiento de Segovia cuentan con los siguientes representantes:</p> <p>Partido Popular (PP): Es el partido político más grande y cuenta con una mayoría absoluta en el ayuntamiento, lo que le permite gobernar sin necesidad de pactos con otros partidos. Ciudadanos-Partido de la Ciudadanía (C's): Este partido tiene representación en el Ayuntamiento de Segovia y suele estar en oposición al Partido Popular. PSOE - Partido Socialista Obrero Español: El PSOE es otro partido importante que cuenta con representantes en el Ayuntamiento de Segovia, aunque no tiene la misma cantidad de representantes que el Partido Popular o Ciudadanos.</p>

	<p>Vox: Este nuevo partido político ha ganado terreno rápidamente y también cuenta con representación en el Ayuntamiento de Segovia.</p> <p>Podemos - Por un Mundo Mejor: Aunque es menos influyente, este partido también tiene representantes en el Ayuntamiento de Segovia.</p> <p>Izquierda Unida (IU): Este partido político suele tener menor influencia y cuenta con pocos representantes en el Ayuntamiento de Segovia.</p> <p>Partido Animalista Contra el Maltrato Animal (PACMA): Es un pequeño partido que tiene representación en algunos ayuntamientos, pero su presencia es menos significativa en el Ayuntamiento de Segovia.</p> <p>Coalición por Segovia: Este partido político suele tener menor influencia y cuenta con pocos representantes en el Ayuntamiento de Segovia.</p> <p>Partido Regionalista Castellano-Leonés (PRCAL): Es un pequeño partido que tiene representación en algunos ayuntamientos, pero su presencia es menos significativa en el Ayuntamiento de Segovia.</p> <p>Votemos por Segovia: Este nuevo partido político ha ganado terreno rápido cuanto y también cuenta con representantes en el Ayuntamiento de Segovia. Es importante destacar que las composiciones del ayuntamiento pueden cambiar cada cuatro años, cuando se celebran elecciones municipales.</p>
--	--

Tabla C.4 – Respuestas a la P4 sin RAG

Utilizando técnicas de RAG se han obtenido las siguientes respuestas.

Con RAG - P1 - ¿Cuál es el precio de los taxis en la ciudad de Segovia?

Modelo	Respuesta
Llama3	<p>Según el acuerdo de resolución emitido por la Comisión Informativa de Urbanismo, Obras y Servicios del Ayuntamiento de Segovia, el precio de los taxis en la ciudad de Segovia para el año 2024 es lo siguiente:</p> <p>Tarifa 1: Bajada de bandera: 1,90 € Percepción mínima: 3,95 € Metros: 1.830,36 Segundos: 359,12 Kilómetro recorrido: 1,12 € Hora parada: 20,55 €</p>

	<p>Tarifa 2 (aplicable en horario nocturno, domingos y festivos, periodo de feria y fiesta local, sábados y días 24 y 31 de diciembre): Bajada de bandera: 2,65 € Percepción mínima: 5,00 € Metros: 1.598,64 Segundos: 293,24 Kilómetro recorrido: 1,47 € Hora parada: 28,85 € Además, hay un suplemento por bulto o maleta de 0,65 €.</p>
<p>Llama3:70b</p>	<p>Según el texto proporcionado, las tarifas de los taxis en la ciudad de Segovia para el año 2024 son las siguientes:</p> <p>Tarifa 1: Bajada de bandera: 1,90 € Percepción mínima: 3,95 € Metros: 1,83036 (no está claro qué significa esto) Segundos: 359,12 (no está claro qué significa esto) Kilómetro recorrido: 1,12 € Hora parada: 20,55 € Tarifa 2 (horario nocturno de 23:00 a 7:00 horas): No se proporcionan precios específicos para esta tarifa. Es importante tener en cuenta que estas tarifas pueden variar dependiendo del contexto y las condiciones de viaje.</p>
<p>Gemma2</p>	<p>El texto proporciona información sobre las tarifas del servicio de autotaxi en Segovia para el año 2024, pero no menciona el precio exacto de los taxis en la ciudad.</p> <p>Te puedo decir que hay dos tarifas:</p> <p>Tarifa 1: Se aplica de 7:00 a 23:00 horas. Tarifa 2: Se aplica de 23:00 a 7:00 horas. El texto detalla los precios por bajada de bandera, percepción mínima, metros, segundos y kilómetro recorrido para cada tarifa.</p> <p>Para saber el precio exacto de un viaje en taxi, necesitarías conocer la distancia recorrida y el tiempo que dure el trayecto.</p>
<p>Phi3</p>	<p>El cuadro tarifario del Servicio de Autotaxi en el término municipal de Segovia para el ejercicio 2024 ha sido aprobado por la Comisión Informativa de Urbanismo, Obras y Servicios. Aunque no tengo información actualizada sobre si se han producido cambios desde entonces, puedo proporcionarle los detalles del cuadro tarifario que fue propuesto en 2019:</p>

	<p>TARIFAS 2024 (propuesta)</p> <p>Tarifa 1: Se aplicará esta tarifa todos los días del año de 07:00 horas a 23:00 horas excepto cuando sea de aplicación la tarifa 2. Los precios son los siguientes: Bajada de Bandera - 1,90 €; Percepción mínima - 3,95 €; Metros - 1.830,36; Segundos - 359,12; Kilómetro recorrido - 1,12 €; Hora parada - 20,55 €</p> <p>Tarifa cuadro tarifario: Se aplicará esta tarifa en horario nocturno de 23:00 a 7:00 horas. Los precios son los siguientes: Bajada de Bandera - 2,65 €; Percepción mínima - 5,00 €; Metros - 1.598,64; Segundos - 293,24; Kilómetro recorrido - 1,47 €; Hora parada - 28,85 €</p> <p>Suplementos: Bulto o maleta - 0,65 €.</p> <p>Tenga en cuenta que estos precios son solo una propuesta y puede haber cambios antes de su implementación. Para obtener información actualizada sobre los precios de taxis en Segovia, le recomiendo contactar con la Asociación Empresarial de Taxi de Segovia o visitar el sitio web oficial del Ayuntamiento de Segovia.</p>
--	---

Tabla C.5 – Respuestas a la P1 con RAG

Con RAG P2 - ¿Se puede subir al Acueducto de Segovia?

Modelo	Respuesta
Llama3	Según el artículo 3.b) de la Ordenanza de Convivencia Ciudadana del Ayuntamiento de Segovia, no está permitido escalar, subir, trepar o descolgarse en el Acueducto de Segovia sin autorización municipal, y esta autorización debe estar basada en el interés público o por motivos de conservación o mantenimiento del monumento.
Llama3:70b	Según el texto proporcionado, no se permite subir o trepar al Acueducto de Segovia sin autorización municipal, que debe estar basada en el interés público o por motivos de conservación o mantenimiento del monumento.
Gemma2	No se permite subir al Acueducto de Segovia sin autorización municipal.
Phi3	No se permite el acceso a las partes superiores del Acueducto de Segovia sin autorización municipal. La entrada y la escalada están restringidas para proteger este importante monumento histórico, que data del siglo II d.C. Las personas interesadas en visitar o estudiar el acueducto pueden hacerlo a través de las rutas turísticas oficialmente establecidas y con los permisos adecuados otorgados por la administración municipal.

Tabla C.6 – Respuestas a la P2 con RAG

Con RAG - P3 - ¿Qué oposiciones hay en el Ayuntamiento de Segovia?

Modelo	Respuesta
Llama3	<p>No encuentro información específica sobre oposiciones en el Ayuntamiento de Segovia en el contexto proporcionado. Sin embargo, puedo decirte que se mencionan dos procesos selectivos:</p> <p>Un proceso selectivo convocado por el Ayuntamiento de Cuéllar para dos plazas de Administrativos como funcionarios de carrera.</p> <p>Un proceso selectivo para la modificación de Créditos número 1/2024 del Presupuesto General del Ayuntamiento de Riaguas de San Bartolomé.</p> <p>No hay más información sobre oposiciones en el contexto proporcionado.</p>
Llama3:70b	<p>Según el texto proporcionado, no se menciona ninguna oposición específica en el Ayuntamiento de Segovia. Sin embargo, sí se mencionan procesos selectivos en otros ayuntamientos, como el Ayuntamiento de Cuéllar, que convoca dos plazas de Administrativos por turno libre.</p>
Gemma2	<p>El texto proporcionado contiene información sobre diferentes anuncios del Boletín Oficial de la Provincia de Segovia, pero no menciona qué oposiciones hay disponibles en el Ayuntamiento de Segovia.</p>
Phi3	<p>En el Ayuntamiento de Segovia se han convocado varias oposiciones para cubrir distintas vacantes y necesidades administrativas. A continuación se detallan algunas de las más recientes:</p> <p>Oposición a la plaza de Administrativo en el Ayuntamiento de Cuéllar (Núm. 78, Pág. 2): Esta oposición fue convocada para cubrir dos plazas de funcionarios de carrera como administrativos. Se requería que los aspirantes tuvieran una justificación documental de sus méritos y se realizó a través del sistema de oposición por turno libre.</p> <p>Aprobado Reglamento del Consejo Sectorial de Medio Ambiente en el Ayuntamiento (Núm. 78, Pág. 3): Se convocó una oposición para la aprobación definitiva del reglamento del Consejo Sectorial de Medio Ambiente del Ayuntamiento.</p> <p>Aprobado precios públicos para los festejos taurinos en el Ayuntamiento (Núm. 78, Pág. 4): Se convocó una oposición para la aprobación definitiva de los precios públicos asociados a los festejos taurinos que se celebrarían en agosto de 2024.</p> <p>Es importante destacar que las oposiciones pueden variar cada año y dependen de la necesidad del Ayuntamiento, por lo que es</p>

	recomendable consultar directamente con el organismo para obtener información actualizada sobre convocatorias abiertas.
--	---

Tabla C.7 – Respuestas a la P3 con RAG

Con RAG -P4 - ¿Qué partidos políticos componen la corporación actual del Ayuntamiento de Segovia?

Modelo	Respuesta
Llama3	<p>Según el texto proporcionado, los partidos políticos que componen la corporación actual del Ayuntamiento de Segovia son:</p> <p>Partido Popular (PP): 12 concejales Partido Socialista Obrero Español (PSOE): 7 concejales Partido Vox: 2 concejales Partido Izquierda Unida (I.U.): 2 concejales Partido Podemos – Alianza Verde (Podemos AV): 1 concejal Ciudadanos (Cs): 1 concejal</p> <p>Es importante destacar que no se menciona la presencia de otros partidos políticos en la corporación, por lo que se puede considerar que estos son los únicos partidos representados.</p>
Llama3:70b	<p>Según el texto, los partidos políticos que componen la corporación actual del Ayuntamiento de Segovia son:</p> <p>Partido Popular (PP) - 12 concejales Partido Socialista Obrero Español (PSOE) - 7 concejales Vox (VOX) - 2 concejales Izquierda Unida (IU) - 2 concejales Podemos – Alianza Verde (Podemos AV) - 1 concejal Ciudadanos (Cs) - 1 concejal</p> <p>Nota: El Grupo municipal Mixto está compuesto por concejales de Podemos AV y Cs.</p>
Gemma2	<p>Los partidos políticos que componen la corporación actual del Ayuntamiento de Segovia son:</p> <p>Partido Popular (PP) Partido Socialista Obrero Español (PSOE) Vox Izquierda Unida (IU) Podemos - Alianza Verde (Podemos AV) Ciudadanos (Cs)</p>
Phi3	<p>Los partidos políticos que componen la corporación actual del Ayuntamiento de Segovia son el Partido Popular (PP), con 12</p>

	concejales; el Partido Socialista Obrero Español (PSOE), con 7 concejales; VOX, con 2 concejales; Podemos AV-Cs, con 1 concejal y Izquierda Unida (IU), también con 1 concejal.
--	---

Tabla C.8 – Respuestas a la P4 con RAG

Apéndice D

Presupuesto para implementar el sistema RAG en una Administración Pública

En este apéndice se pretende reflejar el coste estimado para la implantación de un sistema RAG en una administración pública, como puede ser el Ayuntamiento de Segovia, donde tienen un gestor documental con 5.109.727 archivos que ocupan 1,28 Tb.

Para el cálculo se valoran dos opciones, la instalación y configuración on-premise, en el propio CPD del ayuntamiento, para lo cual será necesario la adquisición de un servidor con procesamiento GPU, cabina de almacenamiento y demás elementos necesarios. La otra opción es contratar la infraestructura en la nube de Amazon Web Services (AWS) o Google Cloud Platform (GCP), pero sería necesario comprobar si se cumple con el RGPD [51] y el ENS [52], además de firmar un acuerdo de confidencialidad.

Respecto al coste en personal, será necesario destinar a un ingeniero de machine learning para el desarrollo del sistema RAG, un ingeniero de sistemas para la infraestructura on-premise o en la nube, un desarrollador web para el interfaz de usuario y un jefe de proyecto para gestionar plazos, contacto con el cliente y documentación.

El tiempo estimado en desarrollar el proyecto es de 6 meses. Se tendrá en cuenta este tiempo para el cálculo del coste de personal. Los precios de los equipos se han obtenido calculando la media de las páginas webs de los fabricantes y tiendas on-line. Los sueldos del personal se han obtenido de la web Indeed [53], referente en la búsqueda y oferta de empleo.

Opción 1: Instalación On-Premise

Concepto	Precio
Servidor Dell PowerEdge R740XD con Intel Xeon Silver 4210, 192 GB RAM, 4 TB de almacenamiento SSD	11.250 €
GPU NVIDIA A100	1.400 €
Almacenamiento adicional en red (NAS) para documentos y backup	5.840 €
Software de virtualización y gestión (VMWare, Red Hat Enterprise Linux). Licencia por un año.	3.850 €
Mantenimiento y soporte anual	4.500 €
Total hardware:.....	26.840 €
Ingeniero Machine Learning (6 meses)	21.816 €
Ingeniero de Sistemas (6 meses)	24.001 €
Desarrollador web (6 meses)	18.002 €
Jefe de Proyecto (6 meses)	35.700 €
Total personal:	99.519 €
Total proyecto:	126.359 €

Tabla D.1 – Presupuesto con implantación On-Premise

Opción 2: En la nube de AWS, actualmente es más barata que la opción de GCP

Concepto	Precio
Instancia EC2 P3 (con GPU NVIDIA Tesla V100): 3.06 €/hora. Para un uso medio de 8 horas al día durante un mes, sería alrededor de 800 € al mes. Precio total por 1 año.	9.600 €
Almacenamiento S3 (documentos): 0.023 €/GB/mes Precio total por 1 año.	353,28 €
Total suscripción en la nube por un año :	9.953,28 €
Ingeniero Machine Learning (6 meses)	21.816 €
Ingeniero de Sistemas (6 meses)	24.001 €
Desarrollador web (6 meses)	18.002 €

Jefe de Proyecto (6 meses)	35.700 €
Total personal:	99.519 €
Total proyecto:	109.472,28 €

Tabla D.2 – Presupuesto con implantación en la nube de AWS

Referencias y Bibliografía

- [1] McTear, Michael, and Marina Ashurkina. Transforming Conversational AI: Exploring the Power of Large Language Models in Interactive Conversational Agents. First edition. New York, NY: Apress Media LLC, 2024. [Transforming Conversational AI: Exploring the Power of Large Language Models in Interactive Conversational Agents | SpringerLink \(uva.es\)](#). Último acceso 22/05/2024
- [2] Gomez-Perez, Jose Manuel, Ronald Denaux, and Andres Garcia-Silva. A Practical Guide to Hybrid Natural Language Processing Combining Neural Models and Knowledge Graphs for NLP. 1st ed. 2020. Cham: Springer International Publishing, 2020. [A Practical Guide to Hybrid Natural Language Processing: Combining Neural Models and Knowledge Graphs for NLP | SpringerLink \(uva.es\)](#). Último acceso 24/05/2024
- [3] Cheonsu Jeong. Fine-tuning and Utilization Methods of Domain-specific LLMs. <https://arxiv.org/abs/2401.02981>
- [4] Jiang, Meng et al. Knowledge-Augmented Methods for Natural Language Processing. 1st ed. 2024. Singapore: Springer Nature Singapore, 2024. [Knowledge-augmented Methods for Natural Language Processing | SpringerLink \(uva.es\)](#) - 24/05/2024
- [5] ChatGPT 4o de OpenAI. <https://openai.com/index/hello-gpt-4o>. Último acceso 05/07/2024
- [6] Bing Copilot de Microsoft. <https://copilot.microsoft.com/?form=MA13FV>. Último acceso 11/07/2024
- [7] Introducing Meta Llama 3: The most capable openly available LLM to date. <https://ai.meta.com/blog/meta-llama-3>. Último acceso 05/07/2024

- [8] Gemma Open Models by Google. <https://ai.google.dev/gemma>. Último acceso 10/07/2024
- [9] Pequeños pero poderosos: los modelos de lenguaje pequeños de Phi-3 con gran potencial. <https://news.microsoft.com/source/latam/features/ia/pequenos-pero-poderosos-los-modelos-de-lenguaje-pequeno-de-phi-3-con-gran-potencial>. Último acceso 05/07/2024
- [10] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, Haofen Wang. Retrieval-Augmented Generation for Large Language Models: A Survey. 27/03/2024. <https://arxiv.org/abs/2312.10997>
- [11] Mariya Mansurova. Text Embeddings: Comprehensive Guide. Towards Data Science. Feb 2024. <https://towardsdatascience.com/text-embeddings-comprehensive-guide-afd97fce8fb5>
- [12] Sabrina Aquino. What is a Vector Database?. January 25, 2024. <https://qdrant.tech/articles/what-is-a-vector-database>
- [13] RAG - Retrieval Augmented Generation: La llave que abre la puerta de la precisión a los modelos del lenguaje. Ministerio para la Transformación Digital y de la Función Pública. <https://datos.gob.es/es/blog/rag-retrieval-augmented-generation-la-llave-que-abre-la-puerta-de-la-precision-los-modelos-del>
- [14] Documentación Docker. <https://docs.docker.com>. Último acceso 29/06/2024
- [15] Get up and running with large language models. Ollama. <https://ollama.com>. Último acceso 28/06/2024
- [16] Build the future of AI with Meta Llama 3. Meta AI. <https://llama.meta.com/llama3>. Último acceso 29/06/2024
- [17] Google AI for developers. Gemma. <https://ai.google.dev/gemma/docs?hl=es-419>. Último acceso 27/06/2024
- [18] Phi-3 family of lightweight open models by Microsoft. <https://ollama.com/library/phi3>. Último acceso 29/06/2024
- [19] Embeddings Model. Nomic-embed-text. <https://blog.nomic.ai/posts/nomic-embed-text-v1>. Último acceso 29/06/2024
- [20] Embeddings Model. Mxbai-embed-large. <https://www.mixedbread.ai/blog/mxbai-embed-large-v1>. Último acceso 29/06/2024

- [21] Embeddings Model. Snowflake-arctic-embed. <https://www.snowflake.com/blog/introducing-snowflake-arctic-embed-snowflakes-state-of-the-art-text-embedding-family-of-models>. Último acceso 29/06/2024
- [22] Documentación Python. <https://docs.python.org/3>. Último acceso 29/06/2024
- [23] Módulo VENV de Python. <https://docs.python.org/es/3/library/venv.html>. Último acceso 01/07/2024
- [24] The AI-native open-source embedding database. <https://docs.trychroma.com>. Último acceso 09/0/2024
- [25] Framework LangChain. <https://python.langchain.com/v0.2/docs/introduction>. Último acceso 03/07/2024
- [26] Paquete Chainlit. <https://docs.chainlit.io/get-started/overview>. Último acceso 03/07/2024
- [27] PyPDF2. A pure-python PDF library capable of splitting, merging, cropping, and transforming PDF files. <https://pypi.org/project/PyPDF2>. Último acceso 03/07/2024
- [28] Computación acelerada por GPU. Nvidia. <https://www.nvidia.com/es-la/drivers/what-is-gpu-computing>. Último acceso 03/07/2024
- [29] Instalación y configuración de Docker en Ubuntu Linux. <https://docs.docker.com/desktop/install/ubuntu>. Último acceso 09/07/2024.
- [30] Instalación de Ollama en un contenedor Docker. <https://ollama.com/blog/ollama-is-now-available-as-an-official-docker-image>. Último acceso 09/07/2024.
- [31] Instalación y configuración de Nvidia Container ToolKit. <https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/index.html>. Último acceso 09/07/2024.
- [32] Documentación de Ollama. <https://github.com/ollama/ollama/tree/main/docs>. Último acceso 09/07/2024.
- [33] How to Install Python3 on Ubuntu 20.04 or 22.04. <https://phoenixnap.com/kb/how-to-install-python-3-ubuntu>. Último acceso 10/07/2024
- [34] Open Models of PHI-3. Microsoft Azure. <https://azure.microsoft.com/es-es/products/phi-3>. Último acceso 17/07/2024

- [35] Mistral AI Documentation. <https://docs.mistral.ai>. Último acceso 18/07/2024
- [36] Google DeepMind. Gemini Pro. <https://deepmind.google/technologies/gemini/pro>. Último acceso 18/07/2024
- [37] Meet Claude of Anthropic. <https://www.anthropic.com/claude>. Último acceso 18/07/2024
- [38] Grok-1 model of xAI. <https://x.ai/blog/grok-os>. Último acceso 18/07/2024
- [39] GPT-3.5 Turbo fine-tuning and API updates of OpenAI. <https://openai.com/index/gpt-3-5-turbo-fine-tuning-and-api-updates>. Último acceso 18/07/2024
- [40] Leonie Monigatti. Improving Retrieval Performance in RAG Pipelines with Hybrid Search. <https://towardsdatascience.com/improving-retrieval-performance-in-rag-pipelines-with-hybrid-search-c75203c2f2f5>. Último acceso 17/07/2024
- [41] Demo of Recursive Retriever and Query Engine. LlamaIndex. https://docs.llamaindex.ai/en/stable/examples/query_engine/pdf_tables/recursive_retriever. Último acceso 17/07/2024
- [42] Cobus Greyling. 12/10/2023. A New Prompt Engineering Technique Has Been Introduced Called Step-Back Prompting. <https://cobusgreyling.medium.com/a-new-prompt-engineering-technique-has-been-introduced-called-step-back-prompting-b00e8954cacb>. Último acceso 17/07/2024
- [43] Tee Tracker. 19/02/2024. RAG with Multi-Query pattern. <https://teetracker.medium.com/rag-with-multi-query-pattern-7272deb3401a>. Último acceso 17/07/2024
- [44] Brian Petro. Revolutionizing Search: How Hypothetical Document Embeddings (HyDE) Can Save Time and Increase Productivity. <https://wfhbrian.com/revolutionizing-search-how-hypothetical-document-embeddings-hyde-can-save-time-and-increase-productivity>. Último acceso 17/07/2024
- [45] Yang et al., EMNLP 2023. PRCA: Fitting Black-Box Large Language Models for Retrieval Question Answering via Pluggable Reward-Driven Contextual Adapter. <https://aclanthology.org/2023.emnlp-main.326>
- [46] Plataforma Google Colaboratory. <https://colab.research.google.com/>. Último acceso 14/09/2024

- [47] Huggin Face documentation. <https://huggingface.co/docs>. Último acceso 15/04/2024
- [48] Jupyter. Free software, open standards, and web services for interactive computing across all programming languages. <https://jupyter.org/>. Último accso 16/09/2024
- [49] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In Text Summarization Branches Out, pages 74–81, Barcelona, Spain. Association for Computational Linguistics. <https://aclanthology.org/W04-1013/>
- [50] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008. <https://nlp.stanford.edu/IR-book/information-retrieval-book.html>
- [51] Reglamento General de Protección de Datos. Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo de 27 de abril de 2016. <https://www.boe.es/doue/2016/119/L00001-00088.pdf>. Último acceso 17/09/2024
- [52] Esquema Nacional de Seguridad. Real Decreto 311/2022, de 3 de mayo, por el que se regula el Esquema Nacional de Seguridad. <https://www.boe.es/buscar/doc.php?id=BOE-A-2022-7191>. Último acceso 17/09/2024
- [53] Web de empleo Indeed. <https://es.indeed.com>. Último acceso 17/09/2024
- [54] Free Online Gantt Chart Software. <https://www.onlinegantt.com>. Último acceso 05/09/2024