ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

Máster en Big Data: Tecnología y Analítica Avanzada

Cadena de automatización de medición de LLMs

Autor
Manuel Oliveira Piñeiro

Dirigido por
Carlos Morrás Ruiz-Falcó

Madrid
Enero 2025

Manuel Oliveira Piñeiro, declara bajo su responsabilidad, que el Proyecto con título **Cadena de automatización de medición de LLMs** presentado en la ETS de Ingeniería (ICAI) de la Universidad Pontificia Comillas en el curso académico 2024/25 es de su autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Manuel Oliveira Piñeiro. Fecha: 17 / 01 / 2025

Autoriza la entrega:

EL DIRECTOR DEL PROYECTO

Carlos Morrás Ruiz-Falcó

Fdo.: ..... Fecha: ..... / ..... / .....

V. B. DEL COORDINADOR DE PROYECTOS

Carlos Morrás Ruiz-Falcó

Fdo.: ..... Fecha: ..... / ..... / .....

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. Manuel Oliveira Piñeiro

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: Cadena de automatización de medición de LLMs, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente).*

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e

intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a …17….. de ……enero……... de …2025.

ACEPTA

Fdo…Manuel Oliveira Piñeiro……

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

Máster en Big Data: Tecnología y Analítica Avanzada

Cadena de automatización de medición de LLMs

Autor
Manuel Oliveira Piñeiro

Dirigido por
Carlos Morrás Ruiz-Falcó

Madrid
Enero 2025

Resumen

Este trabajo de fin de máster se centra en el desarrollo de una cadena de medición automática para LLMs. Con el creciente uso de los LLMs en diversas aplicaciones de procesamiento del lenguaje natural, se hace indispensable contar con mecanismos eficientes y precisos para medir su rendimiento y capacidad.

El proyecto comienza con una introducción detallada al estado del arte, abarcando los diversos problemas que rodean el ámbito de la inteligencia artificial. Se destaca la necesidad de abordar estos desafíos para mejorar la eficiencia y precisión de los modelos de lenguaje actuales. Para ello, se propone una arquitectura basada en RAG, una metodología diseñada para enriquecer la base de conocimiento del modelo.

A continuación, se describe la arquitectura propuesta para el sistema. La implementación de este sistema automatizado tiene como objetivo evaluar el rendimiento del RAG, facilitando la comparación entre distintas versiones del mismo. Estas versiones varían en función de sus parámetros de arquitectura y los hiperparámetros del LLM, lo que permite un análisis exhaustivo de su rendimiento.

La metodología desarrollada en este proyecto incluye varios componentes clave. En primer lugar, se aborda la generación automática de un dataset de evaluación. Luego, se procede a la selección de métricas adecuadas para evaluar el rendimiento del modelo de manera precisa. El proceso de evaluación se automatiza, lo que reduce significativamente el tiempo y esfuerzo requeridos para llevar a cabo estos análisis.

Finalmente, los resultados obtenidos se validan mediante estudios comparativos. Estos estudios permiten corroborar la efectividad de la metodología propuesta y ofrecen una visión clara de las fortalezas y debilidades de cada versión del modelo evaluado.

Abstract

This master's thesis focuses on the development of an automatic measurement chain for LLMs. With the increasing use of LLMs in various natural language processing applications, it is essential to have efficient and accurate mechanisms for measuring their performance and capabilities.

The project begins with a detailed introduction to the state of the art, covering the various issues surrounding the field of artificial intelligence. It highlights the need to address these challenges to improve the efficiency and accuracy of current language models. To this end, an architecture based on RAG is proposed, a methodology designed to enrich the model's knowledge base.

Next, the proposed architecture for the system is described. The implementation of this automated system aims to evaluate the performance of the RAG, facilitating the comparison between different versions of the same. These versions vary based on their architectural parameters and the hyperparameters of the LLM, allowing for a comprehensive analysis of their performance.

The methodology developed in this project includes several key components. Firstly, the automatic generation of an evaluation dataset is addressed. Subsequently, the selection of appropriate metrics to accurately evaluate the model's performance is carried out. The evaluation process is automated, significantly reducing the time and effort required to perform these analyses.

Finally, the results obtained are validated through comparative studies. These studies corroborate the effectiveness of the proposed methodology and provide a clear view of the strengths and weaknesses of each version of the evaluated model.

Agradecimientos

A mi tutor y profesor, Carlos Morrás, por haberme brindando la oportunidad de realizar este máster y estas práctcias, que me han proporcionado un aprendizaje inmenso.

A mis compañeros de clase, y en especial, al equipo de Votconnect, por hacer que cada día en la oficina haya sido productivo y ameno y por conseguir que salga de esa oficina con ganas de volver al día siguiente.

A los demás profesores del máster por dar lo mejor de sí y enseñarnos tanto a nivel académico como profesional.

A mis padres y mi hermano, por guiarme y darme la oportunidad de entrar en esta universidad y realizar el máster que quería.

Por último, a mi novia, por estar ahí cada día para todo y ser mi compañera de vida.

Índice general

Índice de figuras

Índice de cuadros

Acrónimos

IA	Inteligencia Artificial
NLP	Natural Language Processing
LLM	Large Language Model
BERT	Bidirectional Encoder Representations from Transformers
RAG	Retrieval Augmented Generation
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
AWS	Amazon Web Services
API	Application Programming Interface
PDR	Parent Document Retriever

Capítulo 1

Introducción

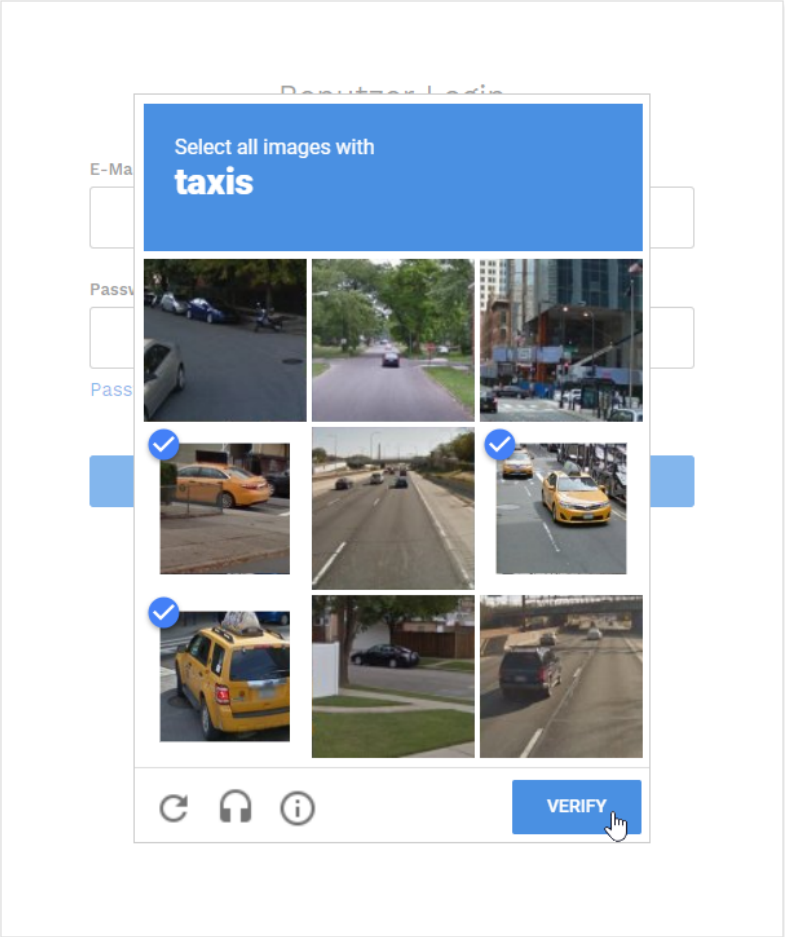
1.1. Descripción del problema y motivación

En el contexto de la IA y el NLP, los LLMs han demostrado grandes capacidades a la hora de generar texto coherente respondiendo preguntas. Sin embargo, a medida que estos modelos se vuelven más complejos, surge un problema vital: la autoevaluación precisa de su rendimiento.

La realidad es que además de los propios LLMs, las arquitecturas de las que forman parte estos modelos también cuentan con parámetros de máxima importancia para conseguir el mejor rendimiento posible. Por ello, al igual que se hace en el machine learning tradicional, la forma de afrontar esta abundancia de parámetros e hiperparámetros, es intentar encontrar la combinación de estos que consiguen la mayor productividad de la arquitectura y del modelo.

Automatizar este proceso supone una reducción de costes, ya que tradicionalmente eran personas las que se dedicaban a evaluar y etiquetar para el entrenamiento estos modelos, con sistemas como puede ser el de CAPTCHA de Google [9](Figura 1.1). La automatización no solo aligera la carga de trabajo humano, sino que también permite evaluaciones más rápidas y menos subjetivas, optimizando así el desarrollo y la implementación de los LLMs.

Figura 1.1: CAPTCHA de Google.



La motivación de este proyecto es doble: por un lado queremos reducir las evaluaciones humanas, que a menudo se vuelven subjetivas y costosas. Por otro lado, buscamos mayor robustez y mejorar la confiabilidad de estos LLMs en entornos de producción, donde estos factores se vuelven diferenciales.

A través de esta cadena de medición automatizada, no solo se logra una evaluación más eficiente y objetiva de los modelos de lenguaje, sino que también proporciona una manera de comparar el rendimiento de tu producto con el de tus competidores. Esto no solo es crucial para optimizar el rendimiento interno del producto, sino que también permite posicionar tu producto de manera más competitiva en el mercado, asegurando que esté a la par o incluso supere a las soluciones ofrecidas por otros competidores.

Por último, conviene tener en cuenta que en un campo como el de la inteligencia artificial, que está en constante desarrollo y donde la tecnología cambia rápidamente, tener un sistema escalable es imprescindible. Evaluar el rendimiento de los LLMs de forma automática nos acerca a este objetivo, además de lograr una velocidad en la medición acorde con el ritmo de avance de la propia IA.

1.2. Objetivos

1. Aplicar las técnicas del estado de arte referentes a la evaluación de LLMs.
2. Automatizar la creación de datasets de preguntas de evaluación.
3. Añadir métricas de valor para poder decidir entre qué versión de modelo/arquitectura es más beneficioso para nuestro caso.
4. Integrar todo de forma que sea muy fácil de utilizar y de sacar los insights correspondientes.

1.3. Estructura del documento

- **Capítulo** 1: Introducción al problema y motivación, incluyendo la situación de la empresa en la que se desarrolló el proyecto.
- **Capítulo** 2: Estado del arte del campo, explicando la historia brevemente y los problemas actuales de la IA.
- **Capítulo** 3: Arquitectura de Votconnect y funcionamiento del RAG.
- **Capítulo** 4: Parámetros e hiperparámetros que intervienen en el versionado del sistema de Votconnect.
- **Capítulo** 5: Pasos en la cadena de medición y métricas utilizadas.
- **Capítulo** 6: Resultados de las pruebas de la cadena de medición.
- **Capítulo** 7: Conclusión de las pruebas y líneas futuras que abordar.

1.4. Contexto de la empresa

Votconnect es una start-up tecnológica cuyo principal producto consiste en ofrecer chatbots personalizados a nivel de cliente. Se distingue en el mercado de la IA por su capacidad de integración fácil y su enfoque en la personalización, lo que permite a sus clientes obtener soluciones adaptadas a sus necesidades específicas.

Para mantenerse competitiva frente a las grandes empresas de IA, Votconnect aplica las técnicas más avanzadas y actuales del estado del arte en IA. Esto asegura que su producto mejore de manera correlacionada con el avance de la tecnología, proporcionando a los clientes una herramienta siempre a la vanguardia de la innovación en inteligencia artificial.

Además de la integración de chatbots, Votconnect ofrece otros productos, aunque no entraremos en profundidad en ellos, ya que el proyecto surgió y se utiliza principalmente para el desarrollo del chatbot.

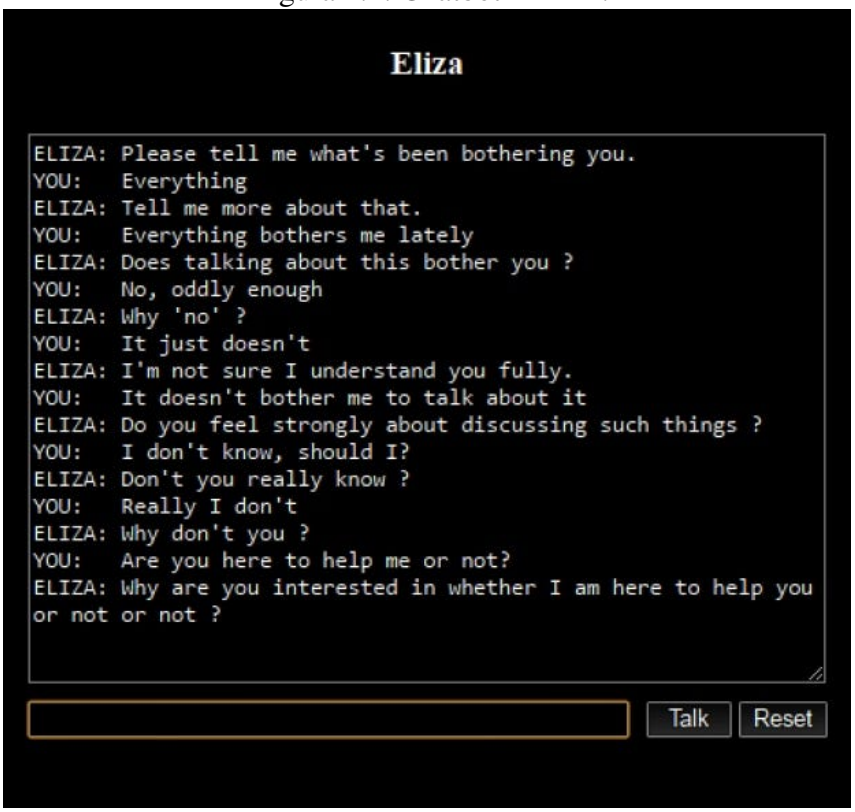
Capítulo 2

Estado del arte

El campo de la inteligencia artificial y el procesamiento de lenguaje natural ha experimentado avances muy significativos en los últimos años, gracias en gran parte al desarrollo de los LLMs. Estos modelos, como GPT-4 de OpenAI, Gemini de Google o Claude 3 de Anthropic, han superado las expectativas y continúan evolucionando, logrando capacidades inimaginables como la generación de audio e imágenes de muy alta calidad.

En los inicios, el campo del NLP se basaba en reglas de programación que no comprendían el lenguaje, lo que limitaba las capacidades de las herramientas disponibles para tareas sofisticadas. Un claro ejemplo de estas limitaciones es el primer chatbot de la historia, ELIZA [26].

Figura 2.1: Chatbot ELIZA.



Esta situación comenzó a cambiar con el auge de Internet y el aumento exponencial de datos disponibles, lo que resultó en el desarrollo de modelos basados en machine learning. Además

del significativo incremento de datos, hubo una notable mejora en la capacidad computacional de los ordenadores, permitiendo la implementación de modelos que utilizaban estos grandes volúmenes de datos.

Gracias a estas mejoras, el uso de redes neuronales volvió a ser relevante, y surgieron varios tipos de redes neuronales muy útiles en el campo del NLP. Entre estas redes destacan las Redes Neuronales Recurrentes (RNN), las Long Short-Term Memory (LSTM) y las Gated Recurrent Units (GRU). Estas arquitecturas fueron diseñadas para manejar secuencias de datos con contextos más largos, haciéndolas especialmente efectivas para tareas como la traducción y la generación de texto.

Sin embargo, estas arquitecturas seguían siendo algo limitadas para procesar texto de manera refinada, ya que los modelos no lograban comprender plenamente el contexto y las relaciones entre las palabras. Fue entonces cuando surgieron los Transformers, introduciendo un concepto revolucionario llamado *atención* [25]. Este mecanismo de atención permite que el modelo entienda cómo se relacionan las palabras en una secuencia, capturando mejor el contexto y las dependencias a largo plazo, resultando en un mayor potencial y precisión en comparación con los modelos anteriores.

De hecho, los LLMs están basados en la arquitectura de Transformers. Esta arquitectura ha demostrado ser muy eficaz para una amplia gama de tareas de procesamiento del lenguaje natural. Modelos como BERT de Google han sido pioneros en el uso de Transformers para comprender mejor el contexto bidireccional de las palabras en una oración, lo que ha llevado a avances significativos en tareas como la respuesta a preguntas y la clasificación de texto.

La llegada de los LLMs ha transformado diversas industrias, desde el servicio al cliente hasta la generación de contenido, pasando por la investigación médica y el análisis de datos financieros. Empresas de todo el mundo están adoptando estos modelos para automatizar y mejorar procesos, reducir costes y ofrecer experiencias personalizadas a sus usuarios.

A pesar de las capacidades avanzadas de los LLMs, llevar estos modelos a aplicaciones prácticas presenta varios desafíos:

- **Alucinaciones**: Los usuarios de estos modelos han observado casos en los que el LLM responde con información irrelevante o no sigue las instrucciones. Estas alucinaciones pueden comprometer la confiabilidad del modelo.
- **Información obsoleta**: La mayoría de los LLMs disponibles, como ChatGPT de OpenAI, fueron entrenados hace tiempo y no tienen conocimiento de eventos ocurridos después de su último entrenamiento, hasta que se libera una nueva versión. Esto significa que cuando un usuario intenta consultar información reciente, el modelo puede no saber qué responder o, peor aún, generar una respuesta inventada.
- **Falta de explicabilidad**: Estos modelos, basados en transformers, ya tienen de por sí una baja explicabilidad. Si además consideramos que muchos LLMs tienen miles de millones de parámetros, entender por qué generan ciertos outputs se vuelve una tarea casi imposible. Esto dificulta la comprensión de su funcionamiento y la identificación de fallos o inconsistencias cuando se integran en entornos de producción.

Independientemente de estos obstáculos, se han conseguido implementaciones exitosas de LLMs

y se anticipa que, con el avance acelerado de estos modelos, cada vez más LLMs serán utilizados en entornos de producción.

El objetivo de este proyecto es implementar una cadena automática de medición para optimizar los LLMs, minimizar los dos primeros problemas mencionados y mejorar el rendimiento general de una arquitectura en la que estos modelos juegan un papel fundamental.

Capítulo 3

Arquitectura

Para entender bien a qué vamos a aplicar nuestra cadena automática de medición, conviene primeramente tener una idea de la arquitectura de Votconnect y cómo se consigue la personalización a nivel de cliente.

3.1. Arquitectura de Votconnect

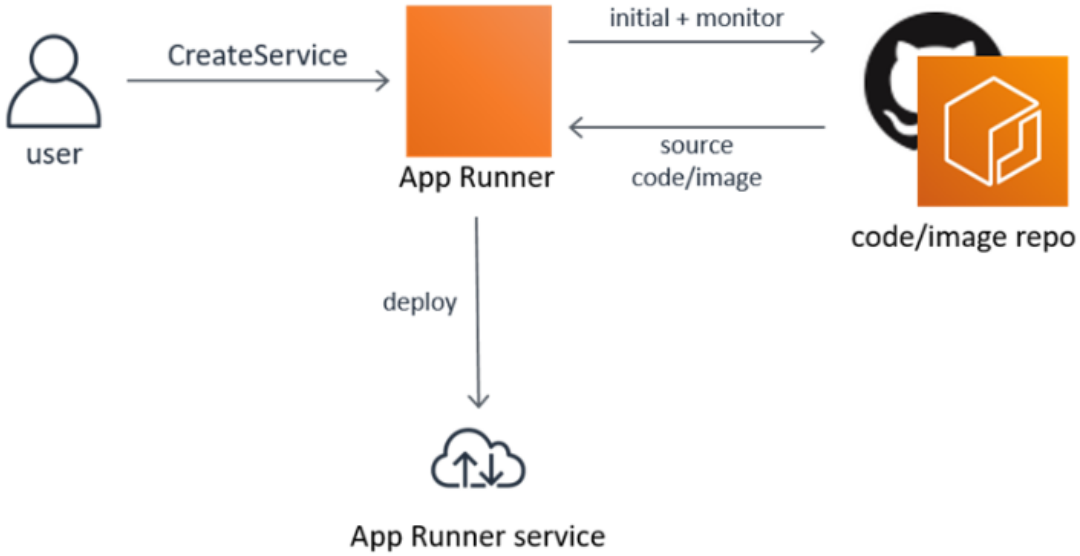
Para el contenido de este proyecto hablaremos de la arquitectura de Votconnect a alto nivel.

El sistema de Votconnect está dividido en dos grandes componentes: el scraper y la parte en la que se desenvuelve el RAG, del cual hablaremos en detalle más adelante.

Ambos componentes se encuentran en la nube, concretamente en AWS. Dentro de AWS, las partes del código están divididas en dos App Runners distintos.

AWS App Runner [20] es un servicio de AWS que permite desarrollar, implementar y ejecutar aplicaciones web en contenedores y sus APIs de forma sencilla, sin necesidad de gestionar la infraestructura subyacente.

Figura 3.1: AWS App Runner.



Nuestros códigos están dockerizados y almacenados en Bitbucket [2], un repositorio de código. Cada vez que hacemos una actualización del código, se actualiza la imagen correspondiente y se realiza el despliegue en el App Runner, obteniendo así la versión actualizada de código en AWS.

Por otro lado, también utilizamos AWS S3 [19] para almacenar archivos en la nube. AWS S3 es un sistema de almacenamiento en la nube que ofrece escalabilidad, disponibilidad de datos, seguridad y rendimiento.

El scraper se encarga de recopilar datos de diversas fuentes web, procesarlos y almacenarlos en AWS S3. Estos datos son esenciales para alimentar los modelos de lenguaje y asegurar que las respuestas generadas sean precisas y relevantes.

Para coordinar y gestionar las operaciones que se hacen, Votconnect utiliza varios servicios adicionales de AWS, como AWS Lambda [21] para la ejecución de funciones serverless, AWS CloudWatch [18] para el monitoreo y registro de actividades, y AWS IAM [22] para la gestión de identidades y accesos, asegurando así la seguridad y el control de los recursos en la nube.

La arquitectura de Votconnect está diseñada para ser escalable y eficiente, permitiendo integrar nuevas funcionalidades y actualizaciones de manera rápida y sin interrupciones en el servicio.

3.2. Personalización del chatbot

Para que el usuario pueda personalizar el chatbot, es necesario que proporcione información, ya sea en formato de documento (PDF, Word, etc.) o a través de una URL. La carga de información se realiza mediante endpoints A.1. El usuario interactúa desde el frontend haciendo peticiones a los endpoints correspondientes del App Runner del scraper.

Una vez se realiza una llamada a un endpoint, tanto en el código como en el propio endpoint al que se hace la petición, se hace una distinción entre si se trata de una URL, PDF, DOC, DOCX, etc. Sin embargo, se sigue una estructura general:

1. **Extracción de la información**: Se aplica web scraping o se utilizan librerías de Python directamente para extraer la información en bruto de la URL o documento en cuestión.

2. **Parseo**: Se parsea y se limpia la información lo máximo posible para acercar lo máximo nuestro texto en bruto a texto legible para un LLM.

3. **Chunking**: Se divide el texto parseado en trozos de texto aplicando un chunking heurístico A.2. Esta técnica se basa en trocear el texto inteligentemente de forma que la información no quede cortada y sea lo más modular posible. Esto se consigue, por ejemplo, dividiendo el texto por puntos, signos de exclamación e interrogación, etc.

   Además, este chunking tiene dos parámetros: *min chunk size* y *max chunk size*, que son los tamaños mínimo y máximo, respectivamente, que puede tener el chunk.

4. **Embedding y metadatos**: Una vez aplicado el chunking, se realiza el embedding A.3 de cada chunk y se añaden metadatos a cada vector. Dependiendo del caso de uso, se añaden unos metadatos u otros, pero siempre se incluyen el texto y la fuente de donde se sacó la

información, para poder mantener trazabilidad.

Por ejemplo, si el usuario carga un PDF llamado *PDFconInfo* con 30 páginas y tenemos un chunk en la página 5 que dice: "*Este es un chunk con información*", el embedding de este chunk sería un número *x*, la fuente sería el nombre del PDF y añadiríamos otro metadato que sea el número de página.

Así, se guardaría:

Embedding: *x*.

Metadatos:

- Texto: *Este es un chunk con información*.
- PDF: *PDFconInfo*.
- Página: *5*.

5. **Guardar en base vectorial**: Finalmente, se añaden los chunks con los metadatos a una base de datos vectorial, en este caso, Pinecone [12].

   Pinecone permite segmentar vectores en distintos namespaces. Por ello, identificaremos cada usuario con un namespace de Pinecone.

En esta fase, ya se nombran los primeros parámetros que entrarán en juego a la hora de aplicar nuestra cadena de medición, referentes al tamaño de cada chunk. Esto es importante, ya que la forma de recuperar contextos se basa en similitud semántica y cuanto más grande sea un chunk, menos "preciso"será el embedding. Sobre esto, volveremos más adelante.

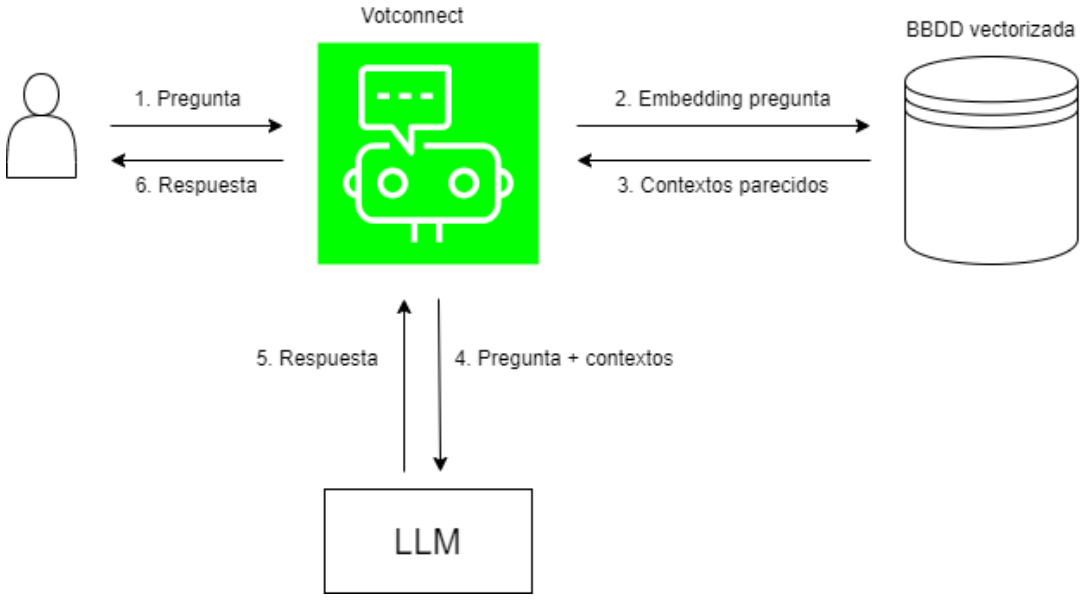
Este proceso de personalización permite que el chatbot pueda adaptarse a los intereses y necesidades específicas de cada usuario, mejorando así la experiencia de usuario y la relevancia de las respuestas proporcionadas.

3.3. RAG

El RAG [17] es una técnica que tiene como objetivo mejorar la precisión y confianza de los LLMs. Recordemos que una de las grandes desventajas de estos modelos radica en la falta de una base de conocimiento actualizada. Esta técnica trata de abordar este problema aportando la información necesaria como contexto sin introducir directamente esta información en la base de conocimiento del modelo.

Después de que el usuario haya añadido información a la base de conocimiento del chatbot, está en posición de hacer preguntas:

Figura 3.2: RAG de Votconnect.



El proceso de RAG en Votconnect se desarrolla de la siguiente manera:

1. El usuario hace la pregunta por llamada a un endpoint y llega al sistema de Votconnect.
2. Se guarda el texto de la pregunta, se realiza el embedding y llega a Pinecone donde tenemos todos los contextos almacenados.
3. Se devuelven los k contextos más parecidos a la pregunta inicial del cliente. La similitud semántica se basa en la distancia de coseno de sus embeddings A.4.
4. Una vez que los contextos llegan al sistema, se completa una plantilla de prompt previamente construida y se pasa este prompt a un LLM.
5. El LLM genera una respuesta basada en el prompt proporcionado.
6. Votconnect devuelve la respuesta final al usuario.

Con el RAG, conseguimos aportar información relevante y actualizada para el usuario. Otra técnica que intenta abordar este problema de manera diferente es el fine-tuning, del cual no entraremos en detalle (ver [11] para más información). Dependiendo del caso de uso, puede ser más conveniente utilizar una técnica u otra. En el caso de Votconnect, RAG superaba al fine-tunning claramente.

En el paso 3, entra en juego otro parámetro que es el número de contextos k. Este parámetro tiene un efecto similar al que anticipamos con el tamaño del chunk. Muchos contextos aportan más información pero también pueden confundir al LLM a la hora de construir la respuesta, por lo que conviene elegir un número que consiga un equilibrio adecuado.

3.3.1. Ventaja del RAG

Veamos un ejemplo de uso muy sencillo y visual donde se saca provecho del RAG. Para ello, supongamos que en 2023, el gobierno francés decidió que París debía dejar de ser la capital francesa y que Toulouse debería pasar a ser la capital.

1. El usuario hace la pregunta: *¿Cuál es la capital de Francia?*.

2. Se guarda la pregunta en el sistema de Votconnect, se hace el embedding y se compara con los vectores que hay en el namespace del usuario de Pinecone.

3. Supongamos que el número de contextos $k = 1$. Entonces se devuelve un vector que tiene el metadato de texto asoaciado: *La capital de Francia es Toulousse*.

4. Así, se construye un ejemplo de prompt:
   Trata de contestar las preguntas del usuario de forma amigable y concisa, siempre basándote en los contextos que se te facilitan. Si no sabes la respuesta, por favor, no te inventes información.

   Pregunta: ¿Cuál es la capital de Francia?

   Contextos: La capital de Francia es Toulousse.

5. Este prompt llega al LLM que contesta: *La capital de Francia es Toulousse.*

6. La respuesta llega al usuario.

En este ejemplo se ve reflejado como el RAG ayuda a nutrir de información por prompt a un LLM base que sin esa información, habría contestado que la capital de Francia es París.

Además, el valor de k está perfectamente seleccionado, pero podría haber pasado que en el primer contexto, se hablase mucho de la capital de Francia sin decir explícitamente cuál es, por ejemplo:

La capital de Francia es un lugar fascinante donde puedes conocer a gente increíble y hay mucha diversidad de culturas.

El embedding de este texto podría ser similar al de la pregunta, por lo que podría ser seleccionado y que $k = 1$ se nos quedase corto. Por ello, muchas veces es más conveniente selecciona más de un contexto para asegurarse de que al LLM se le pasa el contexto en el que se incluye la información para responder a la pregunta.

Capítulo 4

Configuración de versiones

En esta sección definiremos los parámetros e hiperparámetros que determinan las diferentes versiones de nuestra arquitectura. Son estas versiones las que vamos a comparar con nuestra cadena de medición para decidir cómo podemos obtener el mejor rendimiento con la integración del RAG en nuestro sistema.

Se dividen en:

- **Parámetros de la arquitectura.**
- **Hiperparámetros del LLM.**
- **Componentes de la IA.**
- **Estrategias de retrieval.**

4.1. Parámetros de la arquitectura

En este grupo se incluyen los parámetros que intervienen en la recuperación de contextos del RAG. En el capítulo anterior 3 los hemos mencionado y ahora profundizaremos en cómo afectan al funcionamiento del sistema. Dos de los tres parámetros corresponden al chunking, por lo que estarán relacionados.

- **Min chunk size y max chunk size**: Estos parámetros controlan una ventana de posibles tamaños de los chunks. Cuanto más alejados estén los valores de estos parámetros, más volatilidad presentarán los tamaños de los distintos chunks, mientras que si son valores muy cercanos, los tamaños de los chunks no se alejarán mucho entre sí.

  Además de cómo estos parámetros se relacionan para crear una ventana de tamaños de chunks, ambos están directamente correlacionados con el tamaño medio del chunk, siendo esta elección un problema trascendental para el RAG.

  Para entender la magnitud del problema, consideremos el siguiente ejemplo: supongamos que me he registrado como usuario en Votconnect y he cargado un documento que describe el procedimiento en caso de incendio en un hospital. Este documento contiene una descripción del problema, un historial de incendios del hospital y los pasos a seguir en caso de incendio.

El historial de incendios se divide en secciones muy pequeñas, incluyendo el número de heridos y muertos y la duración del incendio, mientras que el procedimiento en caso de incendio, consiste en 10 pasos descritos a lo largo de 2 páginas.

Como usuario, se me ocurren dos tipos de preguntas. Por un lado, quiero saber si hubo algún muerto en el último incendio del hospital. Por otro lado, también puedo querer consultar cuál es el procedimiento en caso de incendio.

Si elegimos valores pequeños para los parámetros, a priori, obtendremos el contexto correcto para responder a la pregunta sobre el número de muertos en el último incendio. Sin embargo, responder adecuadamente a la pregunta sobre el procedimiento puede resultar difícil, ya que es probable que en un solo contexto se incluya solo uno o dos pasos, o incluso que alguno esté dividido. Incluso seleccionando un número alto de contextos, puede ser difícil obtener la información completa y ordenada.

A la vista de este problema, parece que la solución es aumentar el tamaño de chunk. Sin embargo, esto plantea dos problemas:

1. Cuanto más grande es un contexto, más ruido recogerá el embedding, lo que puede llevar a devolver contextos semánticamente no relacionados.
2. En el caso de la consulta sobre el historial de incendios, es probable que en un mismo contexto se incluyan varios registros de incendios, lo que aumenta la probabilidad de confundir al modelo.

Ante esta situación, parece razonable buscar un equilibrio en la elección de estos parámetros. Sin embargo, suele priorizarse el caso de las preguntas concretas, ya que las preguntas que requieren un contexto grande son más difíciles de abordar.

A día de hoy, este es uno de los principales desafíos para el RAG, y la comunidad de IA está trabajando en soluciones como la inclusión de resúmenes como contextos o el uso de ventanas de contexto dinámicas. Sin embargo, sigue siendo un problema sin una solución óptima [14].

- **Número de contextos**: Este parámetro controla el número de contextos que el LLM utiliza para responder la pregunta. Al igual que los parámetros anteriores, según el caso de uso puede ser mejor un número de contextos u otro.

  Siguiendo con el ejemplo anterior, para la pregunta concreta, a priori lo mejor es seleccionar un solo contexto. Sin embargo, esto puede ser arriesgado, ya que la similitud semántica no siempre es una métrica fiable. Por ello, aunque por lo general funciona bien, es prudente ampliar el número de contextos a dos o tres para asegurar que el LLM disponga del contexto correcto.

  Por otro lado, proporcionar demasiados contextos puede confundir al modelo si se le proporciona más información de la necesaria.

  Para preguntas generales, suele ser también mejor optar por un número pequeño de contextos, entre 1 y 4, ya que agregar más contextos no suele resolver el problema de comprensión del contexto.

Hemos visto la importancia de la elección de estos parámetros para el funcionamiento del RAG. En la siguiente sección, veremos cuáles son los hiperparámetros del propio LLM.

4.2. Hiperparámetros del LLM

En esta subsección hablaremos de los hiperparámetros del LLM, que son variables externas al modelo en sí y que afectan a su configuración. A diferencia de los parámetros, que son variables del modelo que surgen durante el entrenamiento, los hiperparámetros son configuraciones que se establecen antes del entrenamiento y afectan al comportamiento del modelo durante la generación de texto. En el caso de los LLM, estos modelos pueden llegar a tener billones de parámetros.

El principal hiperparámetro del LLM es la temperatura. La temperatura controla la aleatoriedad de la respuesta del modelo. Se expresa como un número entre 0 y 1, donde 0 indica una menor aleatoriedad y 1 una mayor.

Como hemos mencionado en secciones anteriores, en entornos de producción es crucial minimizar las alucinaciones del modelo. Por ello, generalmente se seleccionan valores bajos de temperatura, ya que cuanto más alta sea, más propenso será el LLM a generar respuestas irrelevantes o incoherentes. Aunque aumentar la temperatura también puede aumentar la creatividad del modelo, se tiende a priorizar la robustez sobre la creatividad en aplicaciones críticas.

Además de la temperatura, existen otros hiperparámetros de configuración del LLM, pero solo se realizaron pruebas con la temperatura porque es el hiperparámetro más influyente, o directamente no tiene sentido realizar pruebas con el resto.

Otros hiperparámetros del LLM son el *top p*, *top k*, *max tokens*... (ver [15], [3] para más información).

Considerando la importancia de la configuración de estos hiperparámetros, es fundamental comprender su impacto en el comportamiento del modelo durante la generación de texto.

4.3. Componentes de la IA

A los componentes de la IA pertenecen los modelos utilizados en toda la arquitectura del RAG. Estos componentes, a diferencia de todos los definidos anteriormente, van mejorando y actualizándose frecuentemente por el avance de la tecnología.

El enfoque a la hora de medir estos componentes es diferente, ya que se plantea de forma que si el día de mañana OpenAI, Google, Meta o cualquiera de las empresas que lideran la carrera de la inteligencia artificial sacan un modelo, se compara el rendimiento del sistema de Votconnect con ese modelo y con el que actualmente se tiene en Votconnect, y se decide cuál integrar.

El problema es que a la hora de decidir cuál se va a integrar, no sólo se tiene en cuenta el rendimiento, si no que también entran factores como costes o facilidad de integración. Por tanto, la decisión ya no reside en la cadena de medición totalmente.

Votconnect ofrece flexibilidad a la hora de decidir todos los parámetros, ya que todo está extraído por base de datos de forma que cambiar la versión es extremadamente fácil.

En el caso del resto de parámetros que no pertenecen a este grupo, Votconnect ofrece flexibilidad interna. Esto quiere decir que estos parámetros son modificados por los empleados de Votconnect dependiendo del caso de uso. Rara vez el usuario decide modificar los valores de estos parámetros. Esto se debe a que los empleados tienen un conocimiento a bajo nivel de como estos parámetros pueden afectar al rendimiento, mientras que el usuario generalmente no.

En el caso de los componentes de la IA, la flexibilidad es más externa. Esto es debido a que muchos clientes tienen preferencia por algún LLM en particular, bien porque su experiencia es mejor con uno que con otro, o por alguna desconfianza.

Por ejemplo, OpenAI en el pasado filtró datos personales y de vital importancia para algunas empresas al utilizar datos de conversaciones para entrenamiento. Por tanto, puede darse el caso de que un cliente no se fie de OpenAI a raíz de ello y prefiera utilizar otros modelos, por miedo a alguna fuga de sus datos o un uso indebido [1].

A estos componentes de la IA pertenecen:

- **El motor de embeddings utilizado**: Por ejemplo text-embedding-3-small o text-embedding-3-large de OpenAI. De ello depende la calidad de los contextos recuperados.
- **El LLM utilizado**: Por ejemplo ChatGPT-4 o Claude 3. Afecta a la calidad de la respuesta.

Es importante tener en cuenta que ambos pueden afectar notablemente a la latencia del sistema. Por ejemplo, ChatGPT-3.5 tiene bastante menos latencia que ChatGPT-4, por lo que, aunque ChatGPT-4 sea mejor en términos de calidad, a veces merece más la pena tener integrado ChatGPT-3.5 debido a su latencia.

4.4. Estrategias de retrieval

Hasta ahora, hemos hablado exclusivamente del RAG como estrategia de recuperación de contextos. Sin embargo, en un campo tan dinámico como la IA, surgen modificaciones y posibles mejoras del RAG constantemente.

Cuando se comienza a integrar un RAG, uno se da cuenta de todas las debilidades que tiene el sistema, algunas de las cuales ya hemos comentado en este documento, como el problema de los contextos grandes y las preguntas abiertas.

Es por ello que surge la necesidad de comparar también entre distintas estrategias de recuperación de contextos que intenten abordar estos casos.

Durante el proyecto, se realizaron pruebas con solo dos estrategias adicionales. Esto se debe a que se detuvo el trabajo en la mejora del sistema de RAG como tal y se invirtió tiempo en el desarrollo de otros productos. Además, no parecía que ningún avance tecnológico en este campo fuera de especial relevancia para nuestro caso de uso en ese momento.

Es importante destacar que estas estrategias no son necesariamente excluyentes, y muchas veces pueden combinarse de manera efectiva. De hecho, más adelante, cuando se presenten las pruebas combinaremos estas dos buscando optimizar el rendimiento de nuestra arquitectura.

Las dos estrategias a mayores del RAG con las que se hicieron pruebas fueron: el RAG Rerank y el Parent Document Retriever.

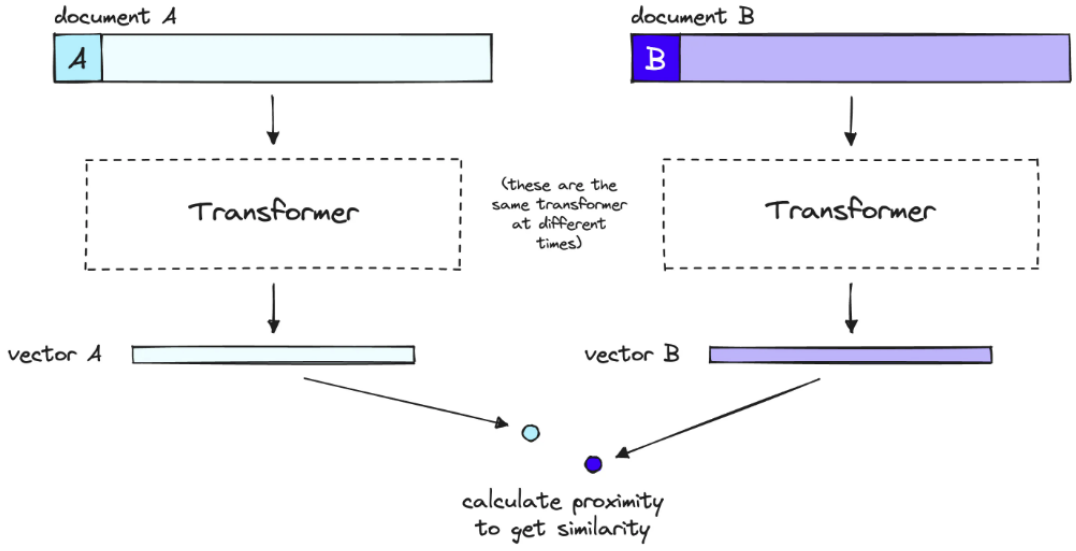
Explorar estas alternativas permite entender mejor las posibilidades y limitaciones de cada enfoque y encontrar la combinación más adecuada para el caso de uso específico.

4.4.1. RAG Rerank

El RAG Rerank [13] es una estrategia de recuperación de contextos que intenta mejorar la calidad de los embeddings añadiendo complejidad al proceso del RAG.

Cuando un usuario hace una pregunta, el RAG tradicional busca la similitud semántica con los contextos de la base de datos a los que previamente se les ha aplicado el embedding. Por lo tanto, el único cálculo que se necesita hacer es el embedding de la pregunta y ya se puede calcular la similitud.

Figura 4.1: RAG tradicional.

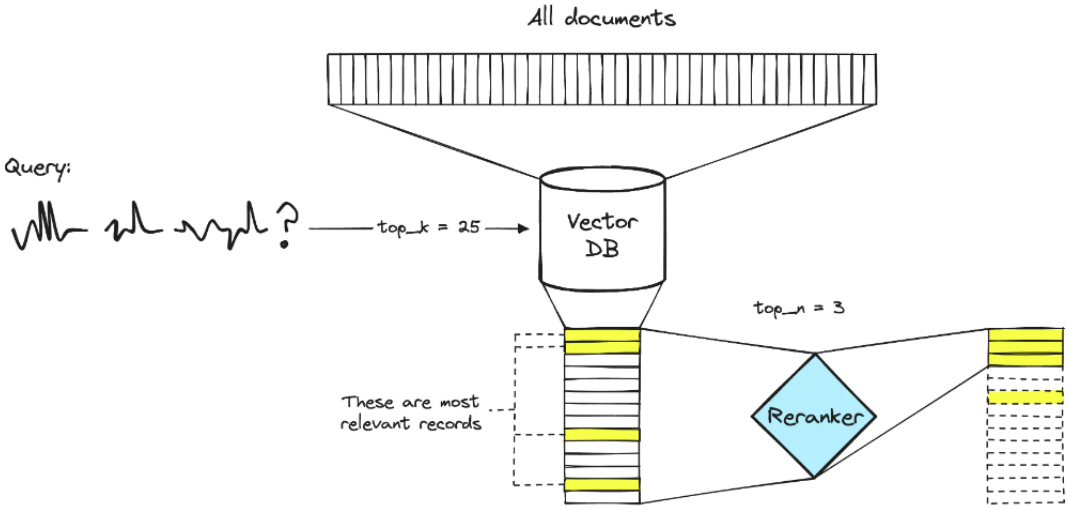


En la Figura 4.1, *documento A* representa la pregunta y *documento B* es un contexto.

El problema del RAG, es que como el embedding de los contextos se está haciendo antes de la pregunta del usuario, los embedding de los contextos tienen que ser lo más generales posibles, sin especificar el caso de uso de la pregunta que hace el usuario.

El RAG Rerank tiene como objetivo solucionar este problema calculando la similitud semántica por tuplas de la consulta más los contextos.

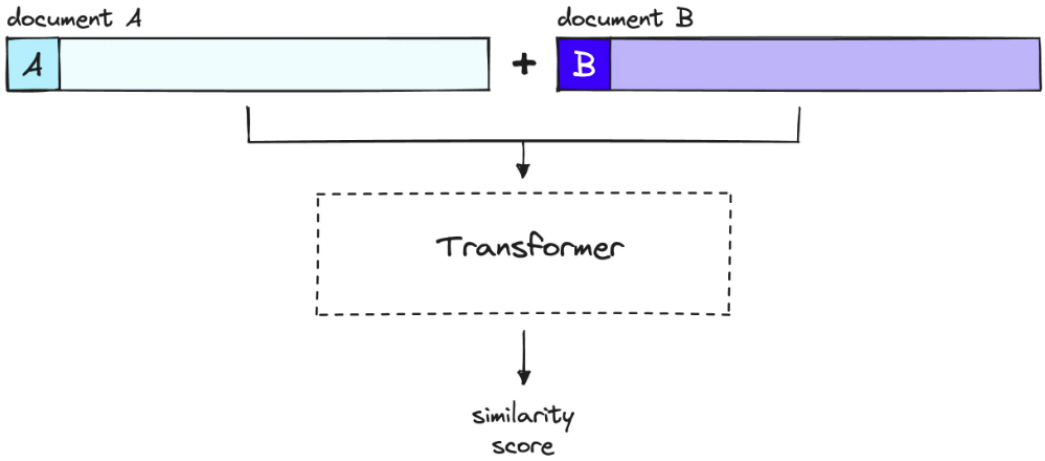
Figura 4.2: RAG Rerank.



Inicialmente, se realiza un retrieval de los documentos más relevantes para la consulta, como se haría en el RAG tradicional.

En la Figura 4.2, en la primera fase se devuelven 25 contextos. Es aquí donde entra en juego el Reranker. De esos 25 documentos, queremos crear 25 tuplas con la pregunta y cada uno de los contextos.

Figura 4.3: Reranker.



El Reranker calcula la similitud con embeddings más específicos, ya que tienen en cuenta el caso particular de la consulta.

Finalmente, se devuelven los n contextos más relevantes para esa consulta. Esto asegura una mayor precisión en los contextos seleccionados, asegurándonos de elegir los mejores posibles para el caso particular de esa pregunta.

Por el propio funcionamiento, el RAG Rerank es considerablemente más lento que el RAG. Su velocidad disminuye cuanto mayor sea k, ya que tendrán que pasar k nuevas tuplas por el Reranker cada vez que se haga una pregunta.

Así, el RAG Rerank es más preciso pero más lento que el RAG. Por tanto, de cara a que el RAG Rerank no tenga latencias excesivas y poder sacarle partido, combiene elegir un k no demasiado alto.

4.4.2. Parent Document Retriever

El Parent Document Retriever [16] intenta solventar el problema de los contextos largos asociando chunks hijos a los chunks originales.

Los chunks hijos son aquellos con los que se realiza la comparación semántica con la pregunta, mientras que el contexto devuelto es el chunk original. De esta manera, los chunks hijos pueden tener embeddings precisos y el chunk original puede tener un tamaño mayor.

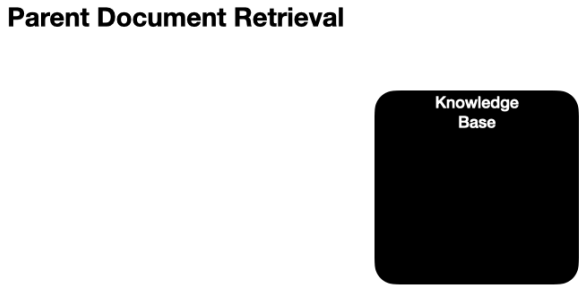


Figura 4.4: Parent Document Retriever

1. Obtención de los contextos del scraper.
2. Generación de resúmenes de los chunks originales que servirán como chunks hijos.
3. Inserción de los embeddings de los chunks hijos en la base de datos. Se les añade un metadato que los relaciona con el chunk padre.
4. Llega la pregunta y se seleccionan los chunks hijos correspondientes por similitud semántica.
5. Se devuelven los chunks originales correspondientes a los chunks hijos seleccionados.

Esto proporciona dos ventajas significativas:

- **Mayor precisión en la similitud semántica**: Al asociar chunks hijos específicos con los chunks originales, se mejora la precisión en la recuperación de contextos relevantes.
- **Añade contexto a los chunks**: Es posible utilizar tanto los chunks hijos como los chunks padres para proporcionar contexto adicional a los hijos, lo que puede mejorar la comprensión del contexto completo.

Este enfoque permite una mejor adaptación a contextos extensos y complejos, mejorando así la capacidad del sistema.

Capítulo 5

Cadena de medición

En este capítulo, comentaremos los diferentes pasos de la cadena de medición para lograr automatizar y simular el proceso de un cliente.

El flujo completo consiste en una serie de procesos, donde cada proceso tiene como resultado un estado específico.

1. **Generación del dataset** → Estado 1:

   En este paso, se genera el conjunto de datos que se utilizará para evaluar el modelo. Es crucial para simular casos reales y variados.

2. **Scraping y respuesta** → Estado 2:

   Aquí se realiza el scraping de los documentos y se proporciona una respuesta del sistema basada en la fuente de conocimiento.

3. **Herramientas de medición** → Estado 3:

   Se aplican diversas herramientas de medición para evaluar el rendimiento del modelo en base a la respuesta proporcionada. Esto incluye métricas de evaluación predefinidas.

4. **Insights** → Estado 4:

   En este último estado, se extraen conclusiones y se generan insights a partir de los resultados de la evaluación. Estos insights son fundamentales para entender el rendimiento del sistema y tomar decisiones para su mejora.

Este flujo de trabajo permite simular y automatizar el proceso que experimentaría un cliente, desde la generación de datos hasta la obtención de insights útiles para la mejora continua del sistema.

5.1. Generación del dataset

La primera parte de la cadena consiste en generar el dataset con el que se va a evaluar el modelo. Para evaluar es necesario tener un dataset de preguntas y respuestas que se tomarán como *ground truth*. La ground truth es la respuesta que se va a tomar como ideal para el modelo, con la que posteriormente compararemos nuestra respuesta.

Por ejemplo, la ground truth para la pregunta *"¿Cuál es la capital de España?"* podría ser:

- *La capital de España es Madrid.*
- *Madrid.*
- *Madrid es una ciudad preciosa que es la capital de España.*

¿Cuál elegir? Pues cada persona o sistema tiene sus requerimientos. Podría elegir la primera por ser una frase bien escrita y concisa, la segunda por ser extremadamente concisa, o la tercera por contener más información explicativa.

Por lo general, se suele intentar tener ground truths lo más concisas posibles para que las métricas funcionen mejor. Las métricas operan por similitud semántica, por lo que, cuanto menos ruido contengan, mejor. Por tanto, la primera y la segunda parecen mejores opciones que la tercera.

Esta parte de la cadena incluye dos opciones:

- **Datasets personalizados**: Se construyen manualmente para casos específicos. Es útil cuando se desean probar el sistema para preguntas concretas en lugar de casos generales. En este caso, es conveniente dedicar tiempo a la generación de preguntas para obtener resultados más fiables. Por ejemplo, para evaluar cómo responde el chatbot a preguntas de recomendación de comidas a partir de un documento que contiene un menú.

  Cuando se trata de este caso, se construye un Estado 0 previo al Estado 1. En este Estado 0 se escriben en un csv seguidamente las preguntas con sus ground truths.

- **Datasets automáticos**: Estos datasets son generados mediante los assistant de GPT.

  El código crea un assistant con el prompt:

  "*Encuentra afirmaciones en el texto que puedan ser contestadas a partir de preguntas. No fuerces a encontrar muchas, sólo extrae las que creas que tienen interés y de las que se pueda reformular preguntas con sentido.*

  Tienes que devolver una string en formato json de las preguntas y respuestas como el que sigue:

  {'questions': [question1, question2, ...], 'answers': [answer1, answer2, ...]}"

  A mayores, se le pasa una mensaje de usuario indicando que intente extraer 30 preguntas como máximo.

Con esto, se invoca el asistente y se devuelve una string. Esta string se valida que tiene el formato json con el método *literal_eval* del módulo *ast* de python [7]. Esto se podría haber intentado llevar a cabo a priori con el modo json de OpenAI [10], pero actualmente existen bastantes restricciones para utilizar este modo con la API de los assistant, por lo que se descartó.

Si se valida, ya tenemos un json que se puede transformar a un dataframe donde tendremos nuestras preguntas y ground truths. Si no, se hace otra llamada al assistant incluyendo un mensaje más haciendo hincapié en la estructura de json requerida.

Es importante tener en cuenta que este método presenta cierto sesgo. El hecho de que un LLM busque afirmaciones que se encuentran en el documento hace que la información para contestar a las preguntas estén completamente contenidas en el texto, lo cual puede no reflejar completamente cómo haría preguntas un usuario real.

En el primer caso, el Estado 1 se construye a través de un código de python que transforma las preguntas y ground truths del csv en un dataframe.

En el caso de la generación automática, el Estado 1 es justamente el dataframe que se obtiene a partir del json.

Figura 5.1: Ejemplo de Estado 1.

Question	Ground_truths
¿Se acoge siempre a nuevos beneficiarios?	Debido a las limitaciones de espacio y recursos, no siempre podemos acoger a nueva
¿Qué tipo de voluntariado hay en Refood España?	Podríamos distinguir 2 tipos diferentes de voluntariado. 1. Rutas: Nuestras voluntaria
¿Qué impacto tiene Refood?	Actualmente, funcionamos 4 días a la semana, rescatando unas 160 raciones de comi
¿Qué hace Refood?	- Rescatar el excedente de comida no servida de los establecimientos del barrio - Lle
¿Qué actividades organiza Refood?	- Eventos: A menudo organizamos pequeños eventos para conocernos mejor, compa
¿Cuáles son los requisitos para ser voluntario?	1. Vivir en Tetuán o Pan Bendito (o cerca) 2. Tener una tarde a la semana (al menos) d
¿Cuáles son los requisitos para ser beneficiario?	- Vivir en el barrio de Tetuán o Pan Bendito (o cerca). - Poder acudir de lunes a jueves
¿Cuales son los principales donantes de alimentos preparados?	Una de nuestras principales fuentes de alimentos preparados son colegios mayores c
¿Cuál es la misión de Refood?	Acabar con el desperdicio alimentario barrio a barrio, redistribuyendo el excedente d
¿Cómo se puede contactar con la organización?	A través del correo informacion@refoodesp.org o bien visitando alguna de sus sedes

5.2. Scraping y respuesta

En esta parte se añade la información necesaria para contestar la preguntas y además, se contestan.

Para ello, simularemos el proceso que haría un usuario a la hora de configurar su chatbot.

1. **Alta de usuario y creación de base de conocimiento**:
   - Llamada al endpoint de creación de usuario.
   - Llamada al endpoint de creación de base de conocimiento y asociación de usuario con ella.
2. **Configuración de la base de conocimiento**:
   - Llamada al endpoint del scraper para scrapear los documentos o URLs con los que queramos configurar nuestra base de conocimiento.

- Los contextos se guardan en AWS S3 y Pinecone. Además, en el código del scraper, se hace un conteo de los contextos que se deben ingresar en Pinecone. Con esto, tenemos control de cuándo ha terminado el proceso de scraping y ya están todos los contextos almacenados en Pinecone.
- La cadena de medición en este paso deja ejecutando una función que comprueba el número de contextos asociados al namespace del usuario en Pinecone.
- Esto se hace debido a que el scrapeo es un proceso que puede tardar algunos minutos, por lo que, como la cadena de medición es un script ejecutado secuencialmente, debemos tener control de cuándo la base de conocimiento está totalmente configurada para poder realizar las preguntas.

3. **Contestación de las preguntas**:

   - Llamada al endpoint de consultas del sistema para cada una de las preguntas del dataframe del Estado 1.
   - Este endpoint devuelve la respuesta del sistema y los contextos utilizados de Pinecone para responder a la pregunta.
   - En la llamada al endpoint se recoge la latencia del sistema para devolver la respuesta.

   El Estado 2 se trata de una ampliación del dataframe del Estado 1, con las preguntas, ground truths, respuestas, contextos y latencias.

Figura 5.2: Ejemplo de Estado 2.

Question	Ground_truths	Answer	Contexts	Latency
¿Qué tipo de voluntariado hay en Refood Españ	Podríamos distinguir 2 tipos diferente	En Refood España, hay dos tipos diferentes de 1. Rutas: Los voluntarios se reúnen en los local Es importante destacar que para ser voluntario	['¿Vives cerca de Tetuán o Pan Bendito? ¿Tienes unas dos horas a	5.26
¿Cómo se puede contactar con la organización?	A través del correo informacion@refo	Puedes contactar con la organización escribien	['en la cual poder hacer llamamientos de apoyo a la actividad de l	1.85
¿Cuáles son los requisitos para ser voluntario?	1. Vivir en Tetuán o Pan Bendito (o cer	Para ser voluntario en Refood España, es nece	['¿Vives cerca de Tetuán o Pan Bendito? ¿Tienes unas dos horas a	3.93
¿Qué impacto tiene Refood?	Actualmente, funcionamos 4 días a la	El impacto de Refood se puede ver en varios a 1. Reducción del desperdicio de alimentos: Re 2. Ayuda a personas en situación de vulnerabil 3. Fomento de la solidaridad y la participación 4. Creación de empleo y oportunidades: A me En resumen, el impacto de Refood se puede v	['Refood llega a España de la mano de su presidente, Alfonso Pura	8.85
¿Qué hace Refood?	- Rescatar el excedente de comida no	Refood es una organización que se dedica a la	['Refood llega a España de la mano de su presidente, Alfonso Pura	4.55
¿Cuales son los principales donantes de alimento	Una de nuestras principales fuentes d	No tengo información específica sobre los prin	['Haz click en los botones para conocernos\nVOLUNTARIAS\nDON	1.87
¿Se acoge siempre a nuevos beneficiarios?	Debido a las limitaciones de espacio y	Basándonos en la información proporcionada,	['Solicitudes para ser beneficiario del proyecto:\nPara poder ser be	3.62
¿Cuáles son los requisitos para ser beneficiario?	- Vivir en el barrio de Tetuán o Pan Be	Los requisitos para ser beneficiario del proyec 1. Gestionar y proporcionar datos personales p 2. Compromiso con el proyecto y sus valores. 3. Tener el carnet de manipulador de alimento 4. Cumplir con los requisitos establecidos por Es importante destacar que estos requisitos pu	['Solicitudes para ser beneficiario del proyecto:\nPara poder ser be	5.1
¿Cuál es la misión de Refood?	Acabar con el desperdicio alimentario	La misión de Refood es reducir el desperdicio	['Refood llega a España de la mano de su presidente, Alfonso Pura	2.04
¿Qué actividades organiza Refood?	- Eventos: A menudo organizamos pe	Refood organiza actividades relacionadas con	['Refood llega a España de la mano de su presidente, Alfonso Pura	4.14

5.3. Herramientas de medición

En esta sección primero hablaremos de las librerías utilizadas para extraer nuestras métricas, incluyendo una definición de las mismas, y finalmente, hablaremos de en qué consiste el Estado 3.

5.3.1. Librerías utilizadas

Por un lado, hablaremos de Ragas, que fue la primera librería utilizada y que finalmente descartamos. En segundo lugar, se hablará de LlamaIndex más en profundidad, ya que fue finalmente la librería utilizada.

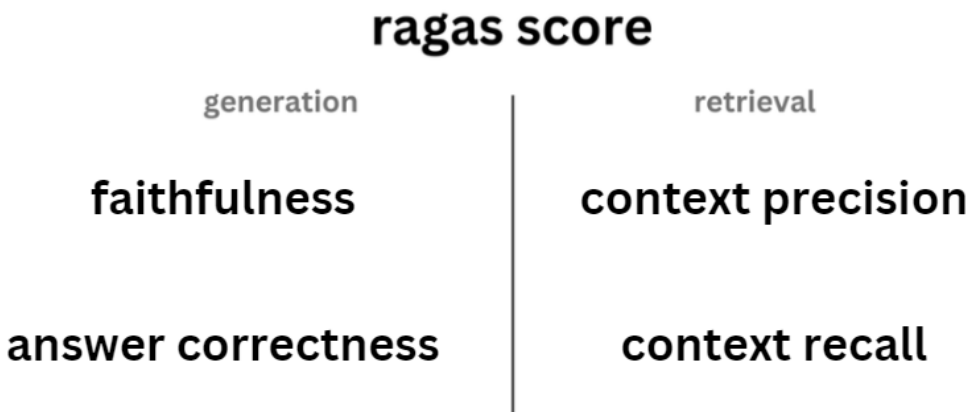
Ragas

Al principio, Ragas [6] fue el marco para evaluar nuestro LLM. Ragas es una herramienta destinada a evaluar y analizar modelos de lenguaje.

Ragas incluye una gran variedad de métricas, incluso ofreciendo la posibilidad de definir métricas personalizadas. En nuestro caso, en los primeros estudios se consideraron todas las métricas que ofrecía la librería, ya que aparentemente no había ningún motivo para descartarlas.

Finalmente, se optó por seleccionar parte de las métricas. El criterio de selección de unas sobre otras fue el valor que aportaba cada una de ellas para obtener insights. Así, nos quedamos con 4 métricas. El resto de métricas solo complicaban los resultados y dificultaban la extracción de conclusiones.

Figura 5.3: Métricas escogidas de Ragas.



En la Figura 5.3 aparecen las métricas que utilizamos para Ragas. No entraremos en las definiciones ya que finalmente no se utilizó Ragas. Para más información, consultar [5].

Más adelante hablaremos de las métricas finalmente elegidas por LlamaIndex, que son similares a las de la Figura 5.3.

La razón que motivó la elección de Ragas fue que, al inicio del proyecto, era uno de los pocos frameworks disponibles que ofrecía la posibilidad de medir el rendimiento de los LLMs. Sin embargo, las latencias de medición eran demasiado altas como para considerarlo una solución definitiva.

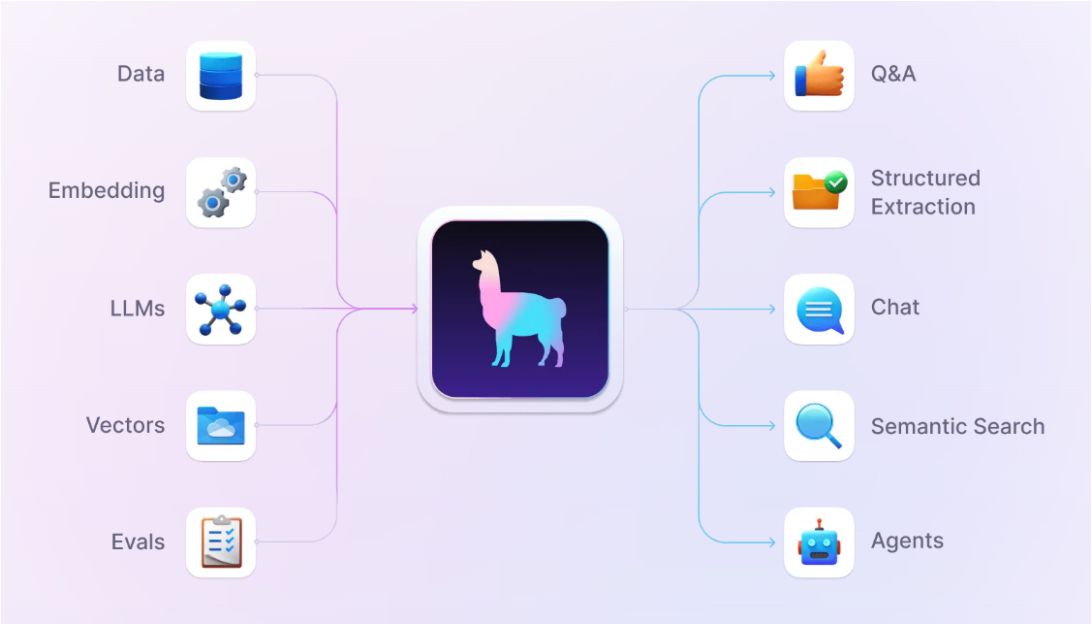
Más adelante, cuando encontramos LlamaIndex, optamos por descartar Ragas porque mejoraba notablemente la latencia ofreciendo un rendimiento muy similar, incluso algo superior.

LlamaIndex

LlamaIndex [23] es un framework que facilita la construcción de RAGs. Proporciona todas las herramientas necesarias para integrar este tipo de sistemas.

A diferencia de Ragas, LlamaIndex incluye muchas más herramientas además que las referentes a la evaluación.

Figura 5.4: Herramientas de LlamaIndex.



LlamaIndex basa su evaluación en llamadas a un LLM. En nuestra cadena de medición, incluimos 3 métricas: correctness, relevancy y faithfulness, dando especial importancia al correctness, al que también nos referiremos como answer correctness.

1. **Correctness**: mide lo correcta que es una respuesta a una pregunta comparando con la ground truth. Esta es la medida más importante. Nos da una idea de calidad general de nuestro RAG. Toma valores de 0 a 10.

   Por ejemplo:
   Pregunta: *¿Cuánto es* $2+2$*?*
   Respuesta con correctness bajo: 3.
   Respuesta con correctness alto: 4.

   Se calcula construyendo un prompt a partir de dos mensajes, uno con instrucciones y otro con los elementos necesarios:

   - **Prompt de instrucciones**:
     You are an expert evaluation system for a question answering chatbot.

     You are given the following information:
     - a user query,
     - a reference answer, and
     - a generated answer.

Your job is to judge the relevance and correctness of the generated answer.
Output a single score that represents a holistic evaluation.
You must return your response in a line with only the score.
Do not return answers in any other format.
On a separate line provide your reasoning for the score as well.

Follow these guidelines for scoring:
- Your score has to be between 0 and 10, where 0 is the worst and 10 is the best.
- If the generated answer is not relevant to the user query, you should give a score between 0 and 1.
- If the generated answer is relevant but contains many mistakes, you should give a score between 2 and 3.
- If the generated answer is relevant but contains some mistakes, you should give a score between 4 and 5.
- If the generated answer is relevant and almost correct,
you should give a score between 6 and 7.
- If the generated answer is relevant and correct, you should give a score between 8 and 9.
- If the generated answer matches perfect with the reference answer, you should give a score of 10.

- **Prompt con los elementos para la llamada al LLM**:
  ## User Query
  {query}

  ## Reference Answer
  {reference_answer}

  ## Generated Answer
  {generated_answer}

2. **Faithfulness**: mide si la información de la respuesta está presente en los contextos. Nos da información de si el LLM se ha basado en la información de los contextos o simplemente ha contestado con información propia. Esto se traduce en saber cuánto está aportando la base de conocimiento a nuestro RAG.

   Esta métrica solo toma valores 0 o 1, dependiendo de si la respuesta del LLM es *"no"* o *"si"* respectivamente.

   Por ejemplo:
   Pregunta: *¿Cuál es la capital de España?*

   Tupla de respuesta y contexto con alta faithfulness:

   - Respuesta: *Madrid.*
     Contexto: *La capital de España es Madrid.*

   Tupla de respuesta y contexto con baja faithfulness:

- Respuesta: *Madrid.*
  Contexto: *La capital de Italia es Roma.*

Se calcula construyendo un prompt con instrucciones y ejemplos:

- **Prompt:**
  Please tell if a given piece of information
  is supported by the context.
  You need to answer with either YES or NO.
  Answer YES if any of the context supports the information, eve
  if most of the context is unrelated.
  Some examples are provided below.

  Information: Apple pie is generally double-crusted.
  Context: An apple pie is a fruit pie in which the principal filling
  ingredient is apples.
  Apple pie is often served with whipped cream, ice cream
  ('apple pie à la mode'), custard or cheddar cheese.
  It is generally double-crusted, with pastry both above
  and below the filling; the upper crust may be solid or
  latticed (woven of crosswise strips).
  Answer: YES
  Information: Apple pies tastes bad.

  Context: An apple pie is a fruit pie in which the principal filling
  ingredient is apples.
  Apple pie is often served with whipped cream, ice cream
  ('apple pie à la mode'), custard or cheddar cheese.
  It is generally double-crusted, with pastry both above
  and below the filling; the upper crust may be solid or
  latticed (woven of crosswise strips).
  Answer: NO

  Information: {query_str}
  Context: {context_str}
  Answer:

3. **Relevancy**: mide la relevancia de los contextos y de la respuesta con respecto a la pregunta.

   Esta medida nos da una idea de como está funcionando la recuperación de contextos. Por un lado, evalúa si tenemos contextos disponibles para proporcionar al LLM el conocimiento para responder, y por otro lado, si los tenemos, si somos capaces de recuperarlos con la similitud semántica.

   Esta métrica solo toma valores 0 o 1, dependiendo de si la respuesta del LLM es *"no"* o *"si"* respectivamente.

   Por ejemplo:

Pregunta: *¿Cuál es la capital de España?*

Tupla de respuesta y contexto con alta relevancy:

- Respuesta: *España es un país donde se come genial.*
  Contexto: *La capital de España es Madir.*

Tupla de respuesta y contexto con baja relevancy:

- Respuesta: *España es un país donde se come genial*
  Contexto: *La comida en España es de las mejores de Europa.*

Se calcula construyendo un prompt con las instrucciones:

- **Prompt:**
  Your task is to evaluate if the response for the query
  is in line with the context information provided.
  You have two options to answer. Either YES / NO.
  Answer - YES, if the response for the query
  is in line with context information otherwise NO.
  Query and Response: {query_str}
  Context: {context_str}
  Answer:

Con estas métricas obtenemos toda la información necesaria para ofrecer insights. A lo largo del desarrollo del proyecto, los prompts de LlamaIndex se fueron modificando para que nos ofreciesen la precisión necesaria y que las evaluaciones correlasen lo máximo posible con el juicio humano, llegando a los prompts que incluimos arriba.

5.3.2. Construcción del Estado 3

En este Estado añadiremos las métricas que hemos comentado en el apartado anterior de LlamaIndex.

Además, añadimos una métrica de monitorización que cuenta las palabras tanto de cada pregunta, como de los contextos utilizados para contestarla: *query_words*. Esta métrica fue incluida para facilitar la extracción de insights, más que para sacar insights en sí, es decir, nos facilitaba la lectura del Estado.

También hicimos una modificación con el correctness. Por un lado, lo normalizamos de forma que tomase valores entre 0 y 1 para que tuviera concordancia con el relevancy y el faithfulness. Por otro lado, asignamos un -1 al correctness de las respuestas que realmente no estuviesen contestando una pregunta.

Por ejemplo:
Pregunta: *¿Cuál es la capital de España?*
Respuesta: *Lo siento, no tengo información para responder a la pregunta.*

La detección de cuando la respuesta realmente no estaba contestando a la pregunta, se hacía

con una llamada a ChatGPT, donde se especificaba que devolviese un "sí.º un "no", dependiendo de si se había contestado a la pregunta o no, respectivametne. Así, el correctness puede tomar valores en el conjunto $\{-1\} \cup [0, 1]$.

Figura 5.5: Ejemplo de Estado 3.

Question	Ground_truths	Answer	Contexts	Latency	Query_words	Faithfulness	Relevancy	Answer_correct
¿Cuál es la visió	Nuestra visión e	La visión de Ebr	['Misión, visión,	2.18	66	1	1	0.9
¿Cuál es la histo	Ebrofood es el r	Ebro Foods es u Durante este pr La empresa cuer Ebro Foods tiene En cuanto a su p En resumen, la h	['Más informació	11.54	133	1	1	0.9
¿Cómo puedo c	Puedes contacta	Para consultas o - Departamento - Oficina de Prer - Departamento - Oficina de Ater - Departamento - Servicios de At - Canal de Denu Recuerda que es	['Contacto\n\nP	8.63	144	1	1	0.9
¿Cuál es la misió	Nuestra misión e	La misión de Ebr	['Conoce Ebro »	2.53	66	1	1	1
¿Cuáles son las	1. Crecimiento o	Las palancas de 1. Crecimiento o 2. Apertura a pa 3. Diferenciación 4. Baja exposició	['Raw text (addit	7.9	82	1	1	1
¿Cómo se define	Son empleados,	Según el código	['[votconnect_im	3.8	136	1	1	0.9
¿Dónde se encu	Ebrofoods cuent	El Grupo Ebro es	['¿Qué producto	2.55	31	0	0	0.8
¿Cuáles son las	Algunas de las n	No tengo inform	['Conoce Ebro -	1.88	82	0	0	-1
¿Cuál es la estra	Nuestra estrateg	- Ser líderes en l - Ofrecer produc - Innovar consta - Expandir nuest - Mejorar la efici - Fomentar la so	['Conoce Ebro »	3.31	74	0	1	0.9

Es importante señalar que en realidad cada pregunta se evalúa 5 veces en la cadena de medición. De esta manera, los valores de faithfulness, relevancy y answer correctness son el promedio de calcular estas métricas cinco veces para la misma pregunta.

Esta repetición se debe a que LlamaIndex realiza evaluaciones haciendo llamadas a un LLM, que no es determinista. Por lo tanto, es posible que al proporcionar el mismo input, el LLM genere respuestas diferentes.

Sin embargo, se llevó a cabo un estudio que reveló que la variabilidad en la evaluación con el mismo input era mínima, por lo que este problema se considera de menor importancia.

5.4. Insights

En el último paso, crearemos un dataframe resumiendo el Estado 3. Este nuevo dataframe, al que llamaremos Estado 4, contendrá toda la información necesaria para realizar comparativas

y evaluar la calidad de una versión o documento.

El dataframe del Estado 4 solo contiene una fila. En él se incluyen los promedios de todas las métricas mencionadas anteriormente a lo largo de todas las preguntas, además de incorporar dos nuevas métricas personalizadas y modificar ligeramente una de ellas.

- ***Answered***: se calcula como uno menos la proporción de preguntas que tienen un valor de answer correctness igual a -1, es decir, las preguntas que no se han respondido:

$$\frac{1 - \sum(\text{preguntas no respondidas})}{\text{total preguntas}}$$

- ***Answer correctness***: esta métrica se modifica para que los valores de -1 no se incluyan en el cálculo del promedio. Así, solo se calcula el promedio con las preguntas que han sido respondidas:

$$\frac{\sum_{i \in Q} \text{answer\_correctness}_i}{|Q|}$$

  donde Q es el conjunto de preguntas tales que $\text{answer\_correctness}_i \neq -1$.

- ***Total***: se calcula a partir de la multiplicación de las dos métricas anteriores. Esta métrica se utiliza como la medida de calidad definitiva y se empleará para comparar diferentes versiones:

$$answer\_correctness * answered$$

Figura 5.6: Ejemplo de Estado 4.

Faithfulness_mean	Relevancy_mean	Answer_correctness_mean	Answered	Query_words_mean	Latency_mean	Total
0.8	0.6	0.81	0.75	119	4.45	0.61

5.5. Comparativas

Para realizar comparativas, nuestra cadena de medición nos ofrece diversas posibilidades dependiendo del caso de uso.

Por ejemplo, si quisiéramos comparar 2 versiones e incluir 5 documentos para la comparativa, podemos combinar los Estados 4 siguiendo los siguientes pasos:

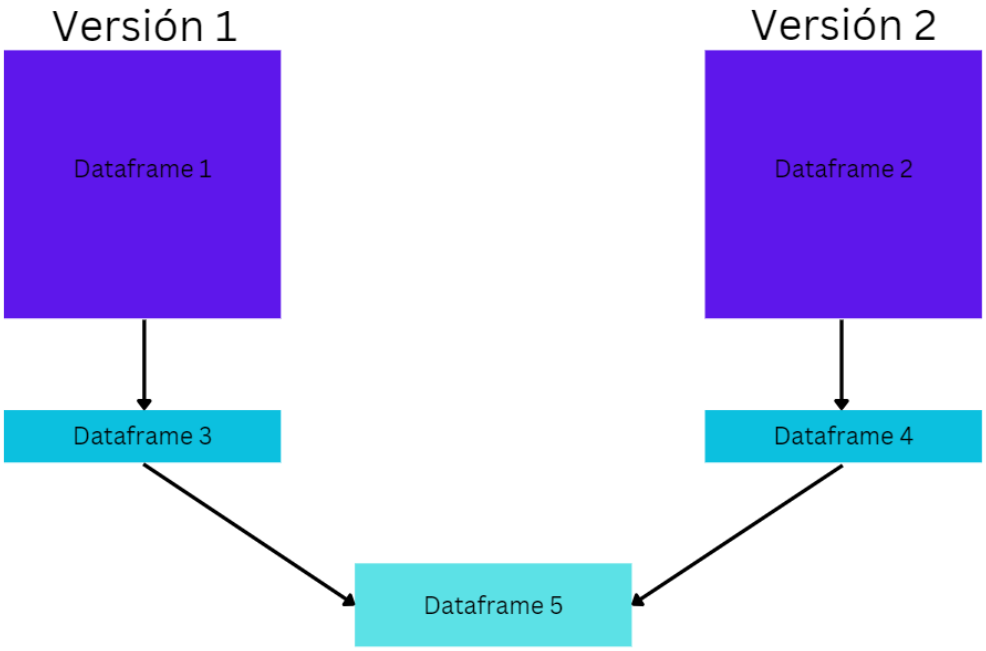
1. Crear el Estado 4 de cada documento de la primera versión, construyendo un dataframe 1 con 5 filas.

2. Realizar el procedimiento análogo para la segunda versión, obteniendo otro dataframe 2 con 5 filas.

3. Construir un Estado 4 promediando las métricas del dataframe 1 y otro promediando las métricas del dataframe 2, resultando en los dataframes 3 y 4, respectivamente.

4. Finalmente, combinar estos dataframes, resultando en un dataframe 5 con 2 filas que contienen las comparativas de las versiones.

Figura 5.7: Diagrama de dataframes



Este diagrama nos da la posibilidad de comparar entre documentos (dataframe 1 y 2) o entre versiones (dataframe 5).

Un ejemplo de comparativa se puede ver en la Figura 5.8 donde se comparan hasta 16 documentos.

Figura 5.8: Ejemplo comparativa entre documentos.

Documento	Faithfulness_mean	Relevancy_mean	Answer_correctness_mean	Answered	Query_words_mean	Latency_mean	Total
Doc1	0.8	0.6	0.81	0.75	119	4.45	0.61
Doc2	0.83	0.83	0.88	0.83	103.17	3.46	0.73
Doc3	0.36	0.4	0.77	0.52	125.68	3.04	0.4
Doc4	0.91	0.83	0.9	0.91	111.22	3.27	0.82
Doc5	0.83	0.83	0.84	0.83	71.17	3.8	0.7
Doc6	0.88	0.92	0.9	0.92	113.48	3.47	0.83
Doc7	0.58	0.67	0.72	0.92	102.42	4.63	0.66
Doc8	0.7	0.9	0.88	0.8	102.9	4.65	0.7
Doc9	0.67	0.78	0.91	0.89	90.44	4.92	0.81
Doc10	0.7	0.6	0.93	0.6	107.4	2.55	0.56
Doc11	0.7	0.8	0.8	0.9	135.3	4.12	0.72
Doc12	0.86	0.71	0.84	1	104.43	5.27	0.84
Doc13	0.78	0.89	0.89	0.78	139.22	3.98	0.69
Doc14	0.8	1	0.8	1	203.56	5.23	0.8
Doc15	0.96	0.96	0.89	0.96	251.28	4.37	0.86
Doc16	0.8	1	0.88	1	163.92	4.35	0.88

A la vista de los resultados, el mejor rendimiento se obtiene para el documento 16 y el peor para el documento 4.

Capítulo 6

Resultados

En este capítulo, incluiremos todos los resultados obtenidos a través de la cadena de medición. Las mediciones se basarán en las versiones de nuestro sistema, construidas a partir de los parámetros mencionados en el capítulo 4.

Para conseguir la combinación óptima de parámetros, no realizaremos un grid search como en el machine learning tradicional. En su lugar, fijaremos todos los parámetros excepto uno y cambiaremos el valor del parámetro a comparar. Aunque lo ideal sería hacer un grid search, se decidió proceder de esta manera para simplificar el problema.

Las razones para esta decisión incluyen la reducción de costes, la minimización del tiempo de las pruebas y el hecho de que hay combinaciones de parámetros que carecen de sentido.

Las pruebas compararán:

- Tamaño mínimo y máximo de chunk.
- Número de contextos.
- Temperatura.
- Estrategias de recuperación de contextos: RAG, RAG Rerank y Parent Document Retriever.
- Componentes de IA.

Como mencionamos en el capítulo 4, las pruebas de los componentes de IA se realizan de forma más espontánea y con un enfoque distinto. Por ello, las pruebas incluidas en este documento serán un ejemplo de los resultados obtenidos al comparar ChatGPT y Mistral, pero no se tratarán de pruebas masivas.

Para todas las pruebas, se utilizarán 5 datasets generados automáticamente a partir de 5 PDFs distintos, lo que nos proporciona un total de 83 preguntas.

6.1. Tamaño de chunk

Cuando introducimos los dos parámetros que intervienen en el tamaño del chunk, mencionamos que la diferencia entre el mínimo y el máximo podía ser grande o pequeña.

Dependiendo de esta diferencia, el tamaño del chunk podía tener más o menos varianza. En nuestro caso, optamos por minimizar esta varianza siempre, por lo que seleccionamos valores de manera que la diferencia siempre fuera igual a 128.

Se utilizaron los tamaños: 128, 256, 384, 512, 640, 768, 896, 1024, 1152 y 1280.

De esta forma, las tuplas de valores eran: $(0, 128), (128, 256), \ldots, (1152, 1280)$, siendo el primer número de la tupla el mínimo y el segundo, el máximo.

Para estas pruebas se utilizaron:

- Número de contextos: 2.
- Temperatura: 0,2.
- Estrategia de recuperación de contextos: RAG.
- Componentes IA: ChatGPT 3,5 y el motor de embeddings text-embedding-ada-002.

Cuadro 6.1: Comparación tamaños de chunk.

	Answer Correctness	**Answered**	**Total**
(0, 128)	0.87	0.87	0.76
(128, 256)	0.89	0.95	0.85
(256, 384)	0.90	0.93	0.84
(384, 512)	0.90	0.95	0.86
(512, 640)	0.90	0.90	0.81
(640, 768)	0.90	0.89	0.80
(768, 896)	0.88	0.89	0.78
(896, 1024)	0.88	0.94	0.83
(1024, 1152)	0.89	0.91	0.81
(1152, 1280)	0.90	0.87	0.78

A la vista de los resultados, la mejor combinación fue $(384, 512)$, que corresponde a un tamaño de chunk mediano, y la peor fue $(0, 128)$, que corresponde a un tamaño de chunk muy pequeño.

Parece que los chunks pequeños/medianos tienen un mejor desempeño, siempre y cuando no sean demasiado pequeños, como en el caso de $(0, 128)$. Esto se debe a que los embeddings de chunks más pequeños concentran mejor la información y son más precisos. Sin embargo, cuando se tratan de chunks excesivamente pequeños, no concentran la suficiente información como para llegar a ser relevantes.

6.2. Número de contextos

Para obtener el número óptimo de contextos, se consideraron los siguientes valores: 1, 2, 3, 4, 5 y 6.

Para estas pruebas se utilizaron:

- Tamaño mínimo de chunk: 384.
- Tamaño máximo de chunk: 512
- Temperatura: 0,2.
- Estrategia de recuperación de contextos: RAG.
- Componentes IA: ChatGPT-3.5 y de motor de embeddings text-embedding-ada-002.

Cuadro 6.2: Comparación número de contextos.

	Answer Correctness	Answered	Total
1	0.89	0.94	0.84
2	0.90	0.95	0.86
3	0.90	0.92	0.83
4	0.88	0.91	0.80
5	0.90	0.89	0.80
6	0.86	0.88	0.76

El mejor rendimiento se consiguió con 2 contextos, mientras que el peor fue con 6 contextos. Los resultados confirman nuestras conclusiones previas: incluir demasiados contextos puede confundir al LLM, ya que se añade información excesiva en muchos casos.

Así, un número pequeño de contextos aporta mejor rendimiento al RAG. A partir de 3 contextos, añadir más contextos aporta más ruido que información.

6.3. Temperatura

Compararemos los siguientes valores de temperatura: 0, 0,1, 0,2, 0,3, 0,4, 0,5, 0,6, 0,7, 0,8, 0,9 y 1.

Para estas pruebas se utilizaron los siguientes parámetros:

- Tamaño mínimo de chunk: 384.
- Tamaño máximo de chunk: 512
- Número de contextos: 2.
- Estrategia de recuperación de contextos: RAG.
- Componentes IA: ChatGPT 3-5 y de motor de embeddings text-embedding-ada-002.

Cuadro 6.3: Comparación temperatura.

	Answer Correctness	Answered	Total
0	0.89	0.92	0.82
0.1	0.89	0.94	0.84
0.2	0.90	0.95	0.86
0.3	0.88	0.95	0.84
0.4	0.87	0.96	0.84
0.5	0.88	0.96	0.85
0.6	0.85	0.96	0.82
0.7	0.86	0.98	0.84
0.8	0.83	0.96	0.80
0.9	0.84	0.98	0.82
1	0.82	0.98	0.80

A la vista de la Tabla 6.3, el valor óptimo para la temperatura es 0,2, mientras que los valores con los que se obtiene peor rendimiento son 0,8 y 1.

Si observamos las columnas, se puede ver claramente cómo los valores de la columna *Answered* van aumentando conforme sube la temperatura, mientras que la columna *Answer Correctness* va disminuyendo.

Esto tiene sentido, ya que a medida que la temperatura asciende, el LLM tiende a alucinar más, respondiendo preguntas para las que realmente no tiene información y, consecuentemente, muchas veces lo hace de forma incorrecta.

Por otro lado, para temperaturas bajas, el LLM sólo responde cuando está seguro y dispone de la información necesaria para ello.

6.4. Estrategias de recuperación de contextos

En esta sección compararemos las tres estrategias de recuperación de contextos que hemos implementado: RAG, RAG Rerank y Parent Document Retriever.

Además, probaremos a combinar el Parent Document Retriever y el RAG Rerank.

Recordemos que cuando introducíamos el RAG Rerank, había dos parámetros relacionados con el recuperación de contextos de contextos: top_k y top_n (ver Figura 4.2). Puesto que para el desarrollo de Votconnect era primordial mantener una latencia baja, el top_k máximo que se probó fue 5.

Para estas pruebas se utilizaron los siguientes parámetros:

- Tamaño mínimo de chunk: 384.
- Tamaño máximo de chunk: 512
- Número de contextos (top_n): 2.
- Top_k: 5

- Componentes IA: ChatGPT 3-5 y de motor de embeddings text-embedding-ada-002.

Cuadro 6.4: Comparación estrategias de recuperación de contextos.

	Latencias	Answer Correctness	Answered	Total
RAG	3.44	0.90	0.95	0.86
RAG Rerank	4.12	0.91	0.94	0.86
PDR	3.61	0.89	0.94	0.84
RAG Rerank + PDR	4.18	0.89	0.95	0.85

Como se podía prever, el RAG Rerank tiene más latencia que el RAG normal. Además, se observa que el RAG y el RAG Rerank están bastante empatados en rendimiento, por lo que lo ideal es quedarse con RAG debido a sus menores latencias.

El Parent Document Retriever no resultó ser del todo efectivo, aunque dependiendo del caso de uso, podría llegar a serlo.

6.5. Componentes IA

Aquí hablaremos de un caso concreto, que es de la comparación de Mistral y ChatGPT.

Este caso surgió cuando Mistral ganó popularidad y se empezó a comparar con el rendimiento de ChatGPT. Mistral contaba con 3 modelos distintos que se diferenciaban en la cantidad de parámetros que tenían cada uno. Compararemos los 3 con ChatGPT-3.5-turbo y ChatGPT-4.

Para este caso, se elaboró un dataset personalizado de 11 preguntas de una web. En ese momento, era un dataset que se utilizaba bastante para las pruebas, porque era muy representativo del rendimiento y tenía preguntas de todo tipo: razonamiento, concretas, etc.

Para estas pruebas se utilizaron los siguientes parámetros:

- Tamaño mínimo de chunk: 384.
- Tamaño máximo de chunk: 512
- Número de contextos: 2.
- Estrategia de recuperación de contextos: RAG.

Cuadro 6.5: Comparación modelos.

	Latencias	Answer Correctness	Answered	Total
GPT 3.5-turbo	1.96	0.81	1	0.81
GPT 4	9.96	0.87	1	0.87
Mistral Tiny	1.69	0.68	1	0.68
Mistral Small	8.1	0.79	1	0.79
Mistral Medium	7.94	0.83	1	0.83

Si nos fijamos en la Tabla 6.5, los únicos modelos con latencias lo suficientemente bajas son el GPT-3.5-turbo y el Mistral Tiny. Al comparar estos dos modelos, está claro que el de OpenAI supera con creces al de Mistral.

Por otro lado, al comparar los modelos con mayor latencia, también queda en evidencia que OpenAI supera a Mistral. Por lo tanto, para el sistema de Votconnect, ChatGPT sigue siendo mejor opción que Mistral.

Capítulo 7

Conclusión y líneas futuras

7.1. Conclusión

En el capítulo anterior 6, hemos comparado las distintas versiones de nuestro sistema y hemos llegado a la conclusión de que la mejor opción para nuestro caso de uso es la siguiente:

- Tamaño mínimo de chunk: 384.
- Tamaño máximo de chunk: 512
- Número de contextos: 2.
- Estrategia de recuperación de contextos: RAG
- Componentes IA: ChatGPT 3-5 y de motor de embeddings text-embedding-ada-002.

A lo largo del documento, hemos observado cómo, dependiendo del caso de uso, la cadena de medición se utilizaba total o parcialmente. Esto depende de nuestros requerimientos y de cuán robusto deseamos que sea el proceso de medición. Por ejemplo, generar el dataset automáticamente añade un sesgo significativo y reduce la robustez de los resultados obtenidos.

A medida que presentamos los resultados, también nos dimos cuenta de que, aparte de la combinación que finalmente resultó ser óptima, hay muchas otras configuraciones que se asemejan mucho en rendimiento a la óptima. Conviene mencionar que los documentos a partir de los cuales se generaron las preguntas automáticas fueron creados a partir de documentos de potenciales clientes. Esto se hizo porque queríamos encontrar la combinación que mejor funcionase en el caso particular de nuestros clientes y no de forma general.

Además, hemos observado que la medición no termina de ser completamente fiable ya que estamos utilizando un LLM para medir, que no es determinista y puede alucinar. Sin embargo, esto también tiene un aspecto positivo; implica que nuestra cadena de medición ganará en precisión y fiabilidad conforme avanza la tecnología y mejoran los modelos.

Cuando se inició el proyecto, prácticamente no existían frameworks que facilitaran la medición, por lo que se trata de un tema aún muy poco maduro y que requiere estar pendiente del estado del arte de forma constante.

En resumen, la combinación de técnicas y herramientas mencionadas proporciona una base

sólida para la medición automática en diversos contextos. No obstante, es crucial seguir explorando y perfeccionando estos componentes para alcanzar un nivel de madurez adecuado para aplicaciones de producción. La evolución constante de la tecnología y la mejora continua de los modelos serán fundamentales para lograr una mayor precisión y fiabilidad en nuestras mediciones.

7.2. Líneas futuras

Desde la finalización del proyecto, han surgido nuevas herramientas de medición que facilitan y mejoran la cadena de medición actual.

Estos avances tecnológicos no sustituyen nuestra cadena de medición. Más bien, la complementan y mejoran. La cadena de medición va más allá de evaluar preguntas y respuestas con métricas; se trata de simular todo el proceso que realiza un usuario, integrarlo con nuestra arquitectura y ofrecer la funcionalidad de generar preguntas y respuestas automáticas.

Por tanto, conforme aparecen mejores frameworks que LlamaIndex, lo ideal sería implementarlos en nuestra cadena de medición automática para mejorarla.

Actualmente, Langsmith [8] ofrece una evaluación automática mediante la inclusión de su API key en el código de ejecución. En su página web, es posible ver un registro de las preguntas realizadas y de sus valoraciones.

De manera similar, Trulens [24] es otro framework que ofrece funcionalidades similares y permite monitorizar las preguntas.

Para mejorar la cadena de medición, una buena opción sería explorar a fondo estos dos frameworks y evaluar cómo se pueden integrar en la cadena de medición actual.

Además, es importante mantenerse al día con los desarrollos más recientes en el campo de la inteligencia artificial y la medición automatizada. La incorporación de nuevas herramientas y tecnologías emergentes podría ofrecer mejoras significativas en la precisión y eficiencia de nuestra cadena de medición.

Otro aspecto a considerar es la colaboración con otros equipos de investigación y desarrollo que trabajan en áreas similares. Esta colaboración podría abrir nuevas oportunidades para mejorar nuestra metodología y adoptar mejores prácticas de la industria.

Finalmente, invertir en la formación continua del equipo sobre las nuevas tecnologías y métodos emergentes en el ámbito de la medición automática puede ser crucial para mantener la competitividad y la innovación en nuestro trabajo. Explorando estas líneas futuras, podemos asegurarnos de que nuestra cadena de medición se mantenga a la vanguardia y continúe proporcionando resultados fiables y precisos.

Apéndice A

Explicación adicional

A.1. Endpoint

Un endpoint es una URL que permite acceder a un recurso específico en una red o sistema. Los endpoints son fundamentales en las arquitecturas de sistemas distribuidos, especialmente en las aplicaciones web y los servicios de microservicios, ya que facilitan la comunicación y el intercambio de datos entre diferentes componentes del sistema.

A.1.1. Tipos de endpoint

Dependiendo del tipo de operaciones que se necesiten realizar, los endpoints pueden ser clasificados en varias categorías, como:

1. **Endpoint de recuperación**: Utilizados para obtener información o datos de un sistema. Generalmente, estos endpoints responden a solicitudes HTTP GET.

2. **Endpoint de creación**: Permiten agregar nuevos recursos al sistema. Responden a solicitudes HTTP POST.

3. **Endpoint de actualización**: Utilizados para modificar o actualizar recursos existentes. Normalmente responden a solicitudes HTTP PUT o PATCH.

4. **Endpoint de eliminación**: Permiten eliminar recursos del sistema. Responden a solicitudes HTTP DELETE.

A.1.2. Endpoints utilizados

Algunos de los endpoints de los que se hizo uso para interactuar con nuestro código de AWS fueron:

- /api/query/
- /api/service/create/
- /api/knowledge_base/create/

A.2. Chunking

El chunking de texto es una técnica utilizada para dividir un texto en trozos más pequeños, conocidos como chunks. Esta técnica es esencial en el procesamiento del lenguaje natural y en la manipulación de grandes volúmenes de texto por modelos de lenguaje, ya que permite que los modelos procesen y analicen textos extensos de manera más eficiente y efectiva.

A.2.1. Ventajas

Algunas de las ventajas que ofrece el chunking son:

1. **Limitaciones de los modelos de lenguaje**: Los modelos de lenguaje tienen limitaciones para procesar grandes cantidades de texto debido a restricciones de memoria y capacidad de cómputo. El chunking ayuda a manejar textos más largos dividiéndolos en partes más pequeñas.

2. **Mejora de rendimiento**: Los chunks reducen la complejidad del texto, permitiendo un procesamiento más focalizado y eficiente.

3. **Facilita la recuperación de la información**: Dividir el texto en chunks hace que los embeddings sean más precisos, lo que mejora la recuperación de información, un elemento vital para el RAG.

A.2.2. Métodos de chunking

Existen diversos métodos de chunking, que pueden basarse en la longitud del texto, en frases o párrafos, entre otros. Estos métodos son ofrecidos por librerías como Langchain, entre otras.

En nuestro caso, optamos por dividir el texto en función de una longitud mínima y máxima. Dentro de estos parámetros, se aplica un chunking heurístico para evitar cortar frases.

Hay casos en los que los chunks superan el tamaño máximo establecido. Esto se debe a que, una vez formado el chunk, se comprueba si la siguiente frase tiene cierta similitud semántica con la última frase del chunk. Si la similitud supera un umbral determinado, la frase siguiente se añade al chunk original, incluso si esto hace que el chunk exceda el tamaño máximo.

Por ejemplo:

Chunk: *Madrid es una ciudad muy bonita. Tiene muchos sitios que visitar. Siempre hay mucho ambiente por las calles y hay una gran variedad de restaurantes y bares.*

Frase_1: *Hay restaurantes italianos, japoneses, chinos, etc.*

Frase_2: *Además, hay dos equipos de fútbol grandes en la ciudad.*

En este caso, la primera frase se añadiría al chunk original, mientras que la segunda no.

Esta estrategia asegura que las unidades de texto sean coherentes y significativas, mejorando la calidad del análisis y la interpretación del modelo de lenguaje.

A.3. Embedding

En el campo del procesamiento del lenguaje natural, un embedding es una representación densa y de baja dimensión de datos en un espacio vectorial. Los embeddings se utilizan para convertir palabras, frases o documentos en vectores numéricos que capturan sus significados semánticos, permitiendo a los modelos de lenguaje manejar datos textuales de manera más eficiente y realizar tareas como la clasificación, la búsqueda y la traducción de textos.

A.3.1. Ventajas

- **Reducción de dimensionalidad**: Los embeddings transforman datos de alta dimensión (como palabras en un vocabulario extenso) en vectores de baja dimensión, lo que reduce la complejidad computacional.

- **Captura de relaciones semánticas**: Los vectores de embedding preservan las relaciones semánticas entre las palabras. Palabras con significados similares tienen vectores cercanos en el espacio de embedding.

A.3.2. Uso

En nuestro caso lo utilizamos para la recuperación de contextos de Pinecone. Como ya se menciona a lo largo del documento, cuanto más ruido tenga el texto, es decir, cuanta más información irrelevante tenga el texto, peor es la recuperación del contexto, porque es más impreciso el embedding.

A.4. Similitud de coseno

La similitud de coseno [4] es una medida utilizada para evaluar la similitud entre dos vectores en un espacio vectorial. En el contexto del procesamiento del lenguaje natural, esta técnica es esencial para comparar embeddings y determinar la similitud semántica entre palabras, frases o documentos.

A.4.1. Definición matemática

La similitud de coseno entre dos vectores $\vec{A}$ y $\vec{B}$ se define como el coseno del ángulo entre ellos:

$$\text{Similitud de Coseno}(\vec{A},\vec{B}) = \frac{\vec{A}\cdot\vec{B}}{\|\vec{A}\|\|\vec{B}\|} \tag{A.1}$$

donde:

- $\vec{A}\cdot\vec{B}$ es el producto punto de los vectores $\vec{A}$ y $\vec{B}$.

- $\|\vec{A}\|$ y $\|\vec{B}\|$ son las normas de los vectores $\vec{A}$ y $\vec{B}$, respectivamente.

A.4.2. Uso

En nuestro caso, lo utilizamos para la comparación de embeddings entre la pregunta hecha por el usuario y los contextos presentes en Pinecone. Así, si el número de contextos es k, se devuelven los k contextos más parecidos en similitud de coseno con la pregunta.

Bibliografía

[1] 20minutos. Chatgpt filtraba por error datos personales reales con solo usar este pequeño truco, 2024. Accessed: 2024-06-15.

[2] Bitbucket. Getting started with bitbucket, 2024. Accessed: 2024-06-18.

[3] Jack Dickens and Evan Tunador. Configuration hyperparameters, 2024. Accessed: 2024-06-15.

[4] Graph Everywhere. Algoritmo de similitud de coseno, 2024. Accedido: 19 de junio de 2024.

[5] ExplodingGradients. Metrics, 2023. Accessed: 2024-06-17.

[6] ExplodingGradients. Ragas documentation, 2023. Accessed: 2024-06-17.

[7] Python Software Foundation. ast — abstract syntax trees, 2024. Accessed: 2024-06-17.

[8] LangSmith. Evaluation overview. `https://docs.smith.langchain.com/old/evaluation`, 2023. Accessed: 2024-06-17.

[9] Yen Nhi Nguyen. How google trains ai with your help through captcha, 2023. Accessed: 2024-06-14.

[10] OpenAI. Json mode for text generation, 2024. Accessed: 2024-06-17.

[11] Pangeanic. ¿qué es fine-tuning? Blog post, Fecha de publicación no especificada. Accessed: 2024-06-14.

[12] Pinecone. Pinecone - an api for building vector similarity applications, 2024. Accedido: 2024-06-18.

[13] Pinecone. Rerankers and two-stage retrieval, 2024. Accessed: 2024-06-15.

[14] Ming Qian, Yuying Xie, Chongxuan Li, Nan Duan, Lei Zhang, and Changyou Chen. Longiclbench: Long-context llms struggle with long in-context learning. 2024. Accessed: 2024-06-15.

[15] Yasaman Razeghi, Roberto Dessì, Leonardo Ruiz, Sebastian Ruder, and Aliaksei Severyn. Llm can achieve self-regulation via hyperparameter aware generation. *arXiv preprint arXiv:2402.11251*, 2024. Accessed: 2024-06-15.

[16] Full Stack Retrieval. Parent document retriever, 2024. Accessed: 2024-06-15.

[17] Amazon Web Services. Retrieval-augmented generation, 2023. Accedido: 2024-06-18.

[18] Amazon Web Services. ¿qué es amazon cloudwatch?, 2024. Accedido: 2024-06-18.

[19] Amazon Web Services. ¿qué es amazon s3?, 2024. Accessed: 2024-06-14.

[20] Amazon Web Services. ¿qué es aws app runner?, 2024. Accessed: 2024-06-14.

[21] Amazon Web Services. ¿qué es aws lambda?, 2024. Accedido: 2024-06-18.

[22] Amazon Web Services. ¿qué es iam?, 2024. Accedido: 2024-06-18.

[23] LlamaIndex Team. *LlamaIndex Documentation*, 2024. Accessed: 2024-06-17.

[24] TruEra. Trulens for llms. `https://www.trulens.org/`, 2023. Accessed: 2024-06-17.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017. Accessed: 2024-06-14.

[26] Tomas Zemcik. A brief history of chatbots. 2019. Accessed: 2024-06-14.