Integración de arquitectura de generación aumentada por recuperación (RAG) en la plataforma Wattwin

Serena Sánchez García

2 de julio de 2024

Resumen– La inteligencia artificial está cada día más presente en nuestra vida, pero los modelos de lenguaje largo que utiliza, presentan algunas limitaciones. ¿Las conoces? Este proyecto aborda dos retos principales: la implementación de una arquitectura RAG para superar las limitaciones de los LLM y la integración completa de esta solución desde cero en Wattwin, una organización sin experiencia previa en este ámbito. El resultado es una solución de IA que no solo mejora la interacción con los usuarios, sino que también es flexible y escalable según las necesidades de la empresa. Este proyecto confirma el potencial transformador de una tecnología bien implementada y demuestra que aprovechar las ventajas que ofrece la IA está en nuestras manos.

Palabras clave– Inteligencia Artificial (IA), Modelo Largo de Lenguaje (LLM), modelo de lenguaje, agente, procesamiento de lenguaje natural (NLP), generación de lenguaje natural (NLG), documentos estructurados, documentos no estructurados, microservicios, búsqueda semántica, CRUD, fragmentos

Abstract– Artificial intelligence is increasingly present in our lives, but the large language models it uses, have some limitations. Are you familiar with them? This project addresses two main challenges: the implementation of a RAG architecture to overcome the limitations of LLMs, and the complete integration of this solution from scratch into Wattwin, an organization with no prior experience in this field. The result is an AI solution that not only improves user interaction but is also flexible and scalable to the company's needs. This project confirms the transformative potential of a well-implemented technology and demonstrates that leveraging the advantages offered by AI is within our reach.

Keywords– Artificial Intelligence (IA), Large Language Model (LLM), foundation model, agent, Natural Language Processing (NLP), Natural Language Generation (NLG), structured documents, unstructured documents, microservices, semantic search, CRUD, chunks

1 Introducción

Este proyecto se centra en la integración de una arquitectura de generación aumentada por recuperación (RAG) en la plataforma Wattwin para adaptar un modelo de lenguaje grande (LLM) a las necesidades del negocio. La arquitectura RAG, implica la integración de fuentes de información adicional a los modelos de lenguaje para mejorar la precisión y fiabilidad de sus respuestas. Esta mejora se logra alimentando los modelos con datos privados de la empresa, como bases de datos internas o documentos, ampliando así el conocimiento y contexto de los LLM sin necesidad de un nuevo entrenamiento.

Wattwin es una plataforma SaaS (Software as a Service) que ofrece servicios de gestión en los procesos de ingeniería relacionados con energías renovables. Ofrece funcionalidades comunes de un CRM y gestiona la creación de ofertas para instalaciones fotovoltaicas. La aplicación de inteligencia artificial a esta plataforma permitirá la creación de un agente especializado en energía. Mediante una arquitectura RAG, un LLM tendrá acceso a los datos internos de la empresa Wattwin, lo que lo capacitará para responder preguntas específicas relacionadas con su ámbito de negocio.

El objetivo principal de este trabajo es establecer una base para desarrollar herramientas de software en Wattwin basadas en inteligencia artificial, capaces de procesar lenguaje natural para la obtención y tratamiento de datos. Esto incluye investigar la arquitectura RAG y los modelos de lenguaje largo asociados, y monitorizar los resultados para evaluar su rendimiento y uso en la plataforma.

El proyecto se divide en dos partes. Primero, se implementará un asistente virtual inteligente con arquitectura RAG, capaz de responder preguntas técnicas y administrativas en ingeniería eléctrica, proporcionando información específica de la plataforma. Segundo, se creará un sistema de control de calidad de las respuestas del agente, permitiendo a los clientes valorar su calidad y a los administradores monitorizar el uso de datos, el coste del chat por usuario y la calidad de las respuestas, facilitando así un plan de mejora continua del proyecto.

2 ESTADO DEL ARTE

En los últimos años, la inteligencia artificial ha adquirido una importancia significativa en nuestra vida cotidiana. En el ámbito empresarial, empresas de todos los sectores reconocen la importancia de la inteligencia artificial y están integrándola en sus plataformas para mejorar sus servicios. A continuación, se presentan algunos conceptos clave para entender mejor su funcionamiento.

Un *Large Language Model* (LLM) es un modelo basado en redes neuronales de tipo *Transformer* y que ha sido entrenado con un corpus muy extenso de datos. Su objetivo principal es modelar patrones del lenguaje para abordar tareas específicas como el procesamiento del lenguaje natural (NLP), que implica la comprensión, procesamiento y manipulación del lenguaje humano, así como la generación de lenguaje natural (NLG). Anteriormente, las redes recurrentes eran las encargadas de esta tarea, ya que el texto era considerado como una secuencia de caracteres [1]. Sin embargo, estaban limitadas a poder recordar solo unas pocas palabras anteriores en la secuencia, por lo que perdían el contexto original. En este contexto, surgieron las redes *Transformer*, diseñadas específicamente para abordar esta limitación.

Las redes *Transformer* son una arquitectura de red neuronal desarrollada por Google en 2017 que, a diferencia de las redes recurrentes, poseen una memoria a más largo plazo gracias a los mecanismos de autoatención y de atención multi-head que procesan información de manera paralela. Su arquitectura se presenta en el artículo "Attention is All You Need" de Vaswani et al. (2017) [2]. Muchos de los modelos de lenguaje existentes hacen uso de este tipo de arquitectura de red neuronal para abordar las tareas de procesamiento y generación de lenguaje natural.

En resumen, un modelo de lenguaje es un modelo computacional capaz de entender y generar lenguaje natural y, un *Large Language Model* (LLM) es un modelo de lenguaje entrenado con una cantidad de datos muy superior. Un ejemplo destacado es ChatGPT, un sistema de chat basado en el modelo de lenguaje GPT-3.5 desarrollado por OpenAI [3]. A pesar de haber sido entrenados con una cantidad masiva de datos, los LLM presentan algunas limitaciones [4]. Entre ellas se encuentran la falta de información en tiempo real y el entrenamiento basado en fuentes genéricas. Estas limitaciones pueden resultar en respuestas poco técnicas y desactualizadas, ya que no reflejan eventos recientes y solo responden a preguntas presentes en su corpus de entrenamiento. Esto puede ser un problema al utilizar un LLM en aplicaciones empresariales, como es el caso de Wattwin. Los LLM especializados en dominios específicos deben recibir datos propietarios y específicos del negocio para proporcionar respuestas precisas y contextualizadas sobre los procesos empresariales particulares, incluso si estos datos no se han utilizado para entrenar el modelo.

En respuesta a estas limitaciones, surge la arquitectura RAG, o *Retrieval Augmented Generation* [5]. Esta técnica se fundamenta en la idea de que el conocimiento de un LLM puede mejorarse proporcionándole contexto adicional, sin la necesidad de reentrenar el modelo o crear un LLM especializado en el negocio en concreto. RAG combina dos conceptos clave: en primer lugar, el uso de representaciones vectoriales sobre los documentos para indexarlos y extraer la información más relevante para cada consulta; y en segundo lugar, la utilización de esta información como contexto para un modelo de lenguaje largo entrenado con fuentes de datos genéricas. La integración de documentación adicional en los *Large Language Models* busca mejorar la precisión y confiabilidad de las respuestas, así como mitigar las alucinaciones.

En el ámbito empresarial, la integración de la inteligencia artificial en plataformas CRM ha experimentado un crecimiento significativo en los últimos años, marcando un cambio fundamental en la manera en que las empresas gestionan las relaciones con sus clientes y mejoran sus operaciones comerciales. Centrándonos en algunas plataformas específicas, como Salesforce [6] y HubSpot[7], encontramos funcionalidades relevantes que pueden servir de inspiración para el uso que se le quiere dar en la empresa Wattwin.

Salesforce destaca como líder en el espacio CRM y ha incorporado la IA a través de Einstein, su plataforma de inteligencia artificial. Einstein ofrece funcionalidades en los ámbitos de Ventas, como la redacción de correos electrónicos enriquecidos con datos de clientes y la generación de resúmenes sobre las llamadas de ventas; en Customer Service, mediante chat de soporte para resolver incidencias de manera más rápida y eficiente; en Marketing, con la personalización individual de las comunicaciones a cada cliente mediante un análisis de sentimiento; y en Comercio, generando descripciones de productos automáticamente y ofreciendo recomendaciones de productos relevantes.

En relación con HubSpot, la integración de la IA en su plataforma CRM también se divide en tres ámbitos: Marketing, generando imágenes mediante un contexto, asuntos de correo para correos ya redactados y contenido para redes sociales; Ventas, para las previsiones de ventas y la calificación predictiva de la reputación de un negocio; y Servicios, mediante un chatbot con IA, asistentes de contenido para el correo electrónico y resúmenes de conversaciones. De esta manera, HubSpot utiliza la IA principalmente para potenciar la automatización del marketing y las ventas.

En general, se observa una gran predisposición de las empresas hacia el uso de la IA para mejorar la satisfacción de los clientes personalizando las comunicaciones y brindando soporte para la resolución rápida de problemas. Asimismo, la IA se emplea para el análisis predictivo y la automatización de procesos, proporcionando herramientas para impulsar las estrategias de gestión de clientes. En el caso específico de Wattwin, este proyecto representa el primer paso hacia la integración de la IA en su plataforma.

3 OBJETIVOS PRINCIPALES

La meta principal del proyecto es el desarrollo de un agente especializado en energía y en los datos de la empresa Wattwin, capaz de responder preguntas técnicas y administrativas, brindando soporte a las consultas de los clientes en la plataforma. A continuación, se detallan los objetivos definidos en orden de prioridad:

- Objetivo 1. Desarrollar una interfaz de chat con un modelo de lenguaje grande de propósito general.
- Objetivo 2. Enriquecer el contexto del agente con documentación no estructurada propia de Wattwin.

- Objetivo 3. Integrar documentación estructurada (bases de datos) de Wattwin en el agente de consultas.
- Objetivo 4. Desarrollar una interfaz destinada a los administradores de Wattwin para la subida de documentación técnica y administrativa para alimentar al LLM.
- Objetivo 5. Implementar un sistema de control de calidad para los administradores, con el propósito de evaluar la eficacia del agente inteligente y analizar los patrones más comunes de interacción de los usuarios con este.

4 METODOLOGÍA

Para la ejecución de este proyecto, se ha seleccionado la metodología Scrumban, que combina estratégicamente dos enfoques ágiles: Scrum y Kanban. Esta metodología es la que se utiliza en Wattwin, por lo tanto, ha facilitado la integración de los plazos y las tareas planificadas en el ritmo habitual del equipo de desarrollo de la empresa. La fusión de estas dos metodologías ágiles aprovecha las fortalezas de cada una: Scrum permite dividir el proyecto en fases específicas, conocidas como Sprints, con una duración definida. En cada Sprint se identifican las tareas concretas que contribuyen a los objetivos del proyecto. En el caso concreto de este proyecto, los Sprints son de 3 semanas. Por otro lado, Kanban se utiliza para visualizar el estado de estas tareas mediante un tablero con columnas que representan diferentes estados, como ”en proceso” o ”hecha”. Esta estrategia facilita la supervisión del progreso de manera clara y sencilla. A continuación, se presentan las herramientas utilizadas para el control de la metodología y el conjunto de herramientas y lenguajes de programación utilizados para el desarrollo del proyecto:

- JIRA: Plataforma integral que permite la creación y seguimiento de tareas en cada Sprint, garantizando un control efectivo de la metodología Scrum y Kanban.
- GIT: Herramienta para el control de versiones asegurando la trazabilidad del código a lo largo del desarrollo del proyecto.
- Visual Studio Code: IDE como entorno de desarrollo, con herramientas útiles para facilitar el proceso de codificación.
- MongoDB Atlas [8]: Sistema de base de datos NoSQL en la nube que permite el almacenamiento de datos vectoriales y creación de índices de búsqueda.
- Figma: Editor de gráficos vectorial utilizado para el diseño de las interfaces.
- Angular2: Framework para aplicaciones web desarrollado en TypeScript, utilizado para la implementación frontend y backend.
- Nest.js [9]: Framework de desarrollo web basado en Node.js que utiliza TypeScript.
- Amazon Bedrock [10]: Servicio de Amazon Web Services que ofrece una selección de modelos fundacionales de las principales empresas de IA.
- Amazon S3 [11]: Servicio de almacenamiento en la nube proporcionado por Amazon Web Services.
- Langchain [12]: Librería para la comunicación con el LLM, la comunicación con MongoDB y el procesamiento, indexación y recuperación de datos.
- Python: Lenguaje de programación utilizado para analizar los resultados.
- RAGAS [13]: Framework para evaluar sistemas RAG.

5 PLANIFICACIÓN

En este apartado se presenta la planificación del proyecto que se ha dividido en seis partes principales. En primer lugar, en una fase de investigación sobre el tema del proyecto, seguida por la definición de los objetivos y requisitos del mismo. En la fase de implementación se han incluido tanto el desarrollo del backend como del frontend, así como la comunicación interna entre ellos. Las últimas semanas del proyecto se han dedicado a evaluar el rendimiento y la usabilidad del chat a nivel funcional, realizando ajustes en el diseño según los resultados obtenidos. Durante todo el desarrollo del proyecto, también se ha establecido una fase para la documentación que debe entregarse.

En la Figura 1, se presenta la planificación del proyecto dividida en fases y subfases. En la columna de la derecha se indica a qué Sprint del equipo de Wattwin correspondió cada fase. Hay algunas fases que no fueron consideradas como tareas de la empresa porque ocurrieron antes o después de las fechas de desarrollo del proyecto, por lo tanto, quedan fuera de los Sprints.

FASES Y SUBFASES	INICIO	FINAL	SPRINT
1. Investigación			
1.1. Orígen del RAG y antecedentes	12-feb.-23	15-feb.-23	--
1.2. Amazon Web Service	15-feb.-23	22-feb.-23	--
1.3. Arquitecturas RAG	15-feb.-23	22-feb.-23	--
2. Definición			
2.1. Objetivos	26-feb.-23	4-mar.-23	1
2.2. Requisitos funcionales y no funcionales	26-feb.-23	4-mar.-23	1
3. Diseño			
3.1. Interfaz Chat y Admin	4-mar.-23	14-mar.-23	1
3.2. Arquitectura back (microservicios)	28-feb.-23	15-mar.-23	1
3.3. Documentos Wattwin	4-mar.-23	13-mar.-23	1
3.4. Comunicación (front-back)	8-mar.-23	17-mar.-23	1
4. Implementación			
4.1. Microservicio model communication (MS1)	18-mar.-23	5-abr.-23	2
4.2. Microservicio data processing (MS2)	8-abr.-23	26-abr.-23	3
4.3. Implementación del front para chat	15-abr.-23	26-abr.-23	3
4.4. Integración MS2 en MS1	29-abr.-23	24-may.-23	4-5
4.5. Implementación del front para admin	29-abr.-23	24-may.-23	4-5
4.6. Comunicación front-back para MS1	8-abr.-23	19-abr.-23	3
4.7. Comunicación front-back para MS2	13-may.-23	24-may.-23	4-5
5. Validación del producto			
5.1. Pruebas de usuario	27-may.-23	21-jun.-23	5-6
5.2. Cambios según estadísticas	27-may.-23	21-jun.-23	5-6
6.Documentación			
6.1. Informe Inicial	26-feb.-23	8-mar.-23	1
6.2. Informe Progreso I	25-mar.-23	19-abr.-23	2-3
6.3. Informe Progreso II	29-abr.-23	24-may.-23	4-5
6.4. Proposta Informe Final	27-may.-23	14-jun.-23	5-6
6.5. Proposta presentació	17-jun.-23	28-jun.-23	--
6.6. Dossier del TFG	17-jun.-23	28-jun.-23	--
6.7. Defensa TFG	28-jun.-23	5-jul.-23	--

Fig. 1: Planificación TFG

La subfase 4.5, que implica la implementación de la interfaz de administración para monitorear el chat, y está incluida en las tareas del Objetivo 5, aún no ha sido completada. Aunque no se ha priorizado la implementación de la interfaz, sí se ha realizado la implementación del almacenamiento de los datos relevantes para poder hacer el monitoreo de forma interna.

El diagrama de Gantt que corresponde a esta planificación se muestra en la Figura 2. Las tareas completadas se muestran en verde, las que están en proceso en azul, y la línea vertical morada marca la fecha de la captura de la imagen (01.07.2024).

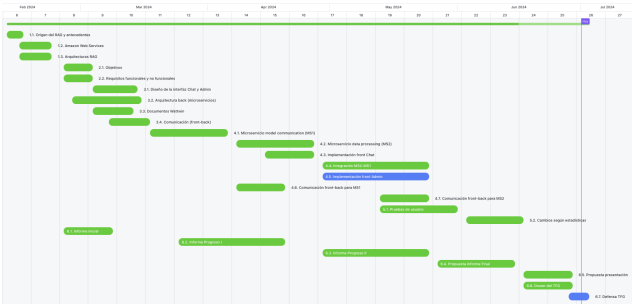

Fig. 2: Diagrama de Gantt de la planificación

6 Diseño e implementación

Para introducir este apartado, se presenta una descripción del esquema general de una arquitectura RAG (Figura 3). Posteriormente, se detalla el desarrollo del proyecto por fases: diseño, implementación y evaluación de los resultados.

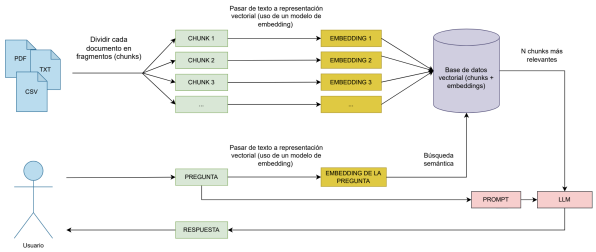

Fig. 3: Arquitectura RAG básica

La arquitectura RAG mejora los LLM mediante la integración de documentos que aportan conocimiento adicional.

Primeramente, los documentos provenientes de diferentes fuentes de datos se fragmentan para facilitar su procesamiento, luego se emplea un modelo de *embedding* para crear representaciones vectoriales que se almacenan en una base de datos. Cuando un usuario formula una pregunta, se genera una representación vectorial de esta usando el mismo modelo de *embedding*. Utilizando algoritmos de cálculo de distancia como similitud coseno o KNN, se puntúa entre 0 y 1 cada fragmento de texto para determinar su relevancia en relación con la pregunta. Los fragmentos más relevantes, es decir, aquellos que se asemejan más semánticamente a la pregunta, reciben una puntuación más alta.

Estos fragmentos seleccionados se incorporan al prompt, un conjunto de instrucciones que se le proporciona al LLM para guiar la generación de texto. En este se incluyen tanto los textos relevantes como la pregunta inicial del usuario, para que el LLM pueda generar finalmente una respuesta con lenguaje natural para el usuario.

6.1. Diseño de la arquitectura en Wattwin

Como se ha mencionado en los objetivos, este proyecto requiere poder utilizado documentos no estructurados (PDF) y estructurados (bases de datos). Además, se debe mantener el contexto de la conversación y la capacidad de responder preguntas fuera del contexto empresarial. Frente a estos requisitos, la arquitectura RAG general explicada en el apartado anterior enfrenta dos problemas principales: la gestión de información en bases de datos y la ejecución de búsquedas semánticas en cada interacción con el LLM. En primer lugar, aplicar la técnica de chunking a bases de datos, presenta múltiples limitaciones:

1. **Número de fragmentos recuperados**: A diferencia de los documentos de texto en PDF, donde los fragmentos pueden tratarse de forma más independiente, las bases de datos requieren recuperar todas las filas correspondientes a una consulta específica. Dividir una base de datos en fragmentos plantea el desafío de no saber cuántos fragmentos se necesitan para obtener toda la información de una consulta.

2. **Actualizaciones en tiempo real**: Los datos en una base de datos pueden cambiar con frecuencia. Para que el sistema diese respuestas correctas sería necesario mantener los fragmentos y sus representaciones vectoriales actualizadas cada vez que se modifique, inserte o elimine un registro en la base de datos, asegurando que los cambios en los datos se reflejen inmediatamente en los fragmentos.

Además de estos inconvenientes, realizar una búsqueda semántica en cada interacción del chat implica recuperar fragmentos de texto cada vez, incluso cuando no es necesario. Por ejemplo, si un cliente simplemente saluda antes de hacer una pregunta, la IA podría responder directamente sin necesidad de realizar una búsqueda adicional. Esta forma directa de respuesta ahorra tiempo y costes de procesamiento, evitando la recuperación innecesaria de documentos y la adición de información extra al prompt del modelo. Asimismo, se ha observado que la recuperación constante de información adicional puede llevar a que la IA incluya datos irrelevantes, lo que afecta la precisión de las respuestas. En la Figura 4 se muestra un ejemplo de interacción real con el usuario donde este problema queda reflejado.

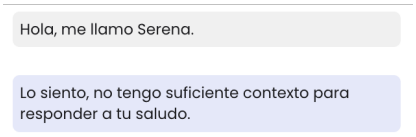

Fig. 4: Ejemplo de conversación iniciada con saludo.

Debido a los nuevos retos enfrentados en la implementación, se a ha realizado un rediseño de la arquitectura RAG general (Figura 5), de forma que pueda decidir si es necesario recurrir a documentación adicional o no y, también, poder añadir la información de las bases de datos de forma consistente. Tras una breve investigación, se encontró el concepto de agente [14]: un nuevo enfoque que permite determinar mediante un LLM los pasos a seguir para responder a una tarea concreta.

Introducción a los agentes

La idea central detrás de un agente es emplear un modelo de lenguaje para razonar sobre la secuencia de pasos necesarios para responder preguntas o realizar acciones. A diferencia de una cadena de acciones lineal, donde se siguen todos los pasos incluso si no son necesarios, un agente decide qué tareas realizar y en qué orden. En nuestro caso, el agente evalúa si debe realizar una búsqueda semántica, consultar una base de datos o responder directamente utilizando su conocimiento base.

Para funcionar correctamente, un agente requiere herramientas (*tools*) que representen las acciones disponibles entre las cuales puede elegir para resolver la tarea o consulta planteada. Cada herramienta tiene un nombre, una descripción y una función asociada con parámetros de entrada, cuerpo de la función y especificación de salida. Dependiendo de la entrada recibida, el agente decide si necesita utilizar ninguna, una o varias herramientas. Una vez que el agente obtiene la salida de la herramienta, esta información junto con la tarea o consulta del usuario se añade al prompt del agente, generando una respuesta en lenguaje natural.

La diferencia principal en esta nueva arquitectura es que ya no se cuenta únicamente con un LLM para interactuar, sino con un agente que utiliza dicho modelo para tomar decisiones. Este enfoque permite utilizar la búsqueda semántica solo cuando sea necesario y separar las consultas sobre documentos de las consultas sobre bases de datos. Esto resulta en una división clara de responsabilidades que hace de esta alternativa una solución flexible y escalable para futuros desarrollos.

Diseño final de la arquitectura backend

Esta arquitectura se divide en dos microservicios basados en el framework de JavaScript Nest.js, y utiliza la base de datos vectorial: MongoDB Atlas. En la Figura 5 se muestra el diseño de la arquitectura final del sistema donde integramos el agente al proceso, la base de datos utilizada, Elastic Search y Amazon S3.

A continuación, se proporciona una explicación más detallada sobre la arquitectura diseñada.

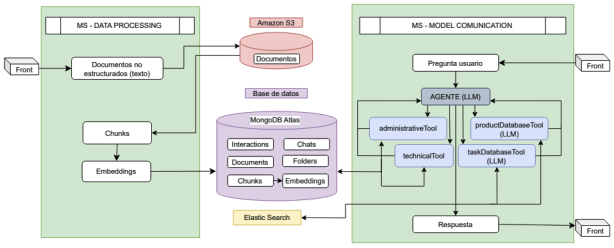

Fig. 5: Arquitectura backend final.

El microservicio denominado ”ms-data-processing”, a la izquierda en la imagen, se dedica al procesamiento de documentos PDF sin imágenes. Estos documentos se cargan en el servicio de Amazon S3 y simultáneamente, divide los datos en bloques, genera sus representaciones vectoriales correspondientes y las almacena en MongoDB Atlas. Además, este microservicio facilita operaciones CRUD (Create, Read, Update and Delete) para gestionar los datos de los documentos de manera eficiente.

Por otro lado, el microservicio de la derecha, ”ms-model-communication” se encarga de comunicarse con el modelo fundacional de Amazon Bedrock: Claude, para dar respuesta a las consultas de los usuarios. El agente recibe las preguntas y gracias a la información de las herramientas implementadas devuelve una respuesta informada. También se encarga de gestionar el CRUD de las conversaciones de los usuarios, gestionar la memoria para proporcionar contexto al LLM y guardar información relevante para monitorizar el uso del chat.

6.2. Diseño de las interfaces de usuario

En esta sección se muestra el diseño de las interfaces de usuario creadas con Figma para cumplir con los requisitos establecidos. Estas interfaces han sido diseñadas para ser intuitivas y coherentes con el resto de la plataforma. El proyecto incluye tres interfaces: una para las conversaciones, otra para la subida de documentos y una de monitoreo para los administradores.

6.2.1. Interfaz de conversaciones

La interfaz de conversaciones comienza con la aparición de una barra lateral derecha al presionar un botón en el encabezado de la página, la cual permanece visible independientemente al resto de pantallas. Esto permite al usuario navegar por la plataforma mientras mantiene el chat abierto para realizar consultas en cualquier momento.

Al abrir la pantalla, se muestra un chat vacío con propuestas de temas para facilitar al usuario iniciar la conversación. Se puede seleccionar una opción para enviar una pregunta por defecto relacionada con el tema escogido o iniciar la conversación de forma libre. Para acceder al historial de chats, se debe presionar una flecha de retroceso en la cabecera del chat. Cada chat en el historial puede ser eliminado o se puede editar su título manualmente.

Dentro de un chat con mensajes anteriores, se muestran todas las respuestas previas del agente. Se ofrecen opciones para copiar la respuesta al portapapeles, solicitar una nueva respuesta y marcar la respuesta como incorrecta. Las interfaces mencionadas se muestran en la Figura 6.

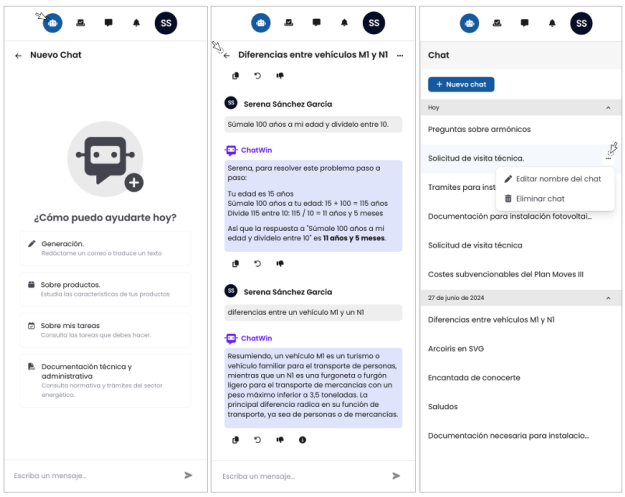

Fig. 6: Pantallas del chat.

6.2.2. Interfaz de subida de documentos

Esta página sirve para subir documentos y está ubicada en el espacio de la plataforma destinado a los administradores. Esta interfaz (Figura 7) permite a personas autorizadas subir documentos de texto en formato PDF a dos carpetas proporcionadas por el sistema: Documentación Administrativa y Documentación Técnica. Estos documentos serán utilizados para proporcionar información adicional al modelo fundacional. Para cada documento se permite descargar, editar el nombre del archivo, ver versiones previas y archivar el documento. En este último caso dejaría de utilizarse como información adicional para el LLM.

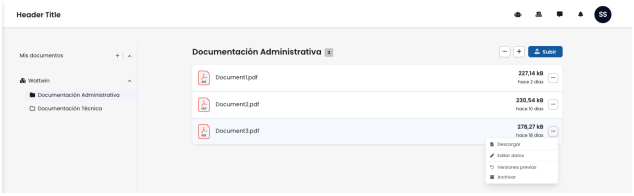

Fig. 7: Subida de documentos.

6.2.3. Interfaz de monitoreo

Se ha diseñado otra pantalla dentro del espacio de administradores para visualizar información relevante para hacer seguimiento del uso que los usuarios finales le dan al chat. Se ha creado un diseño sencillo que permite la visualización de estadísticas básicas sobre el rendimiento del agente y las interacciones de los usuarios. Dentro de esta información se incluye qué documentos se están usando para responder a las preguntas, lo que permite identificar temas relevantes para los usuarios y corregir posibles sesgos del modelo. Además, se proporciona información sobre el número de respuestas consideradas incorrectas por los usuarios, el número de tokens procesados y el coste de las interacciones con la IA. Esta pantalla de estadísticas se muestra en la Figura 8.

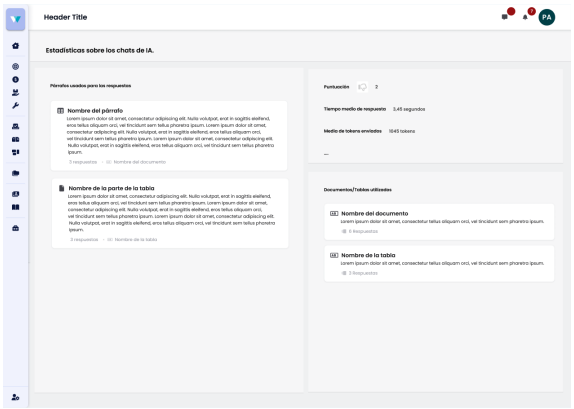

Fig. 8: Visualización de estadísticas.

6.3. Desarrollo

En esta sección se detallarán las diferentes etapas del desarrollo del proyecto desde una perspectiva más técnica.

6.3.1. Microservicio ”ms-data-processing”

El microservicio ”ms-data-processing” ha sido desarrollado para gestionar únicamente el procesado de documentos no estructurados, cumpliendo con el Objetivo 4. Además, maneja el CRUD relacionado con archivos, las carpetas de documentos y los chunks. A continuación, se porporciona una explicación detallada del flujo de trabajo del microservicio y se muestra el modelo de datos (Figura 9) creado para gestionar esta lógica.

Este microservicio se activa al recibir una solicitud de subida de documento por parte de los administradores de la empresa, quienes gestionan la información utilizada para ampliar la base de conocimiento del modelo.

Las funcionalidades principales de este microservicio incluyen crear una instancia en *Documents* cada vez que un administrador carga un archivo desde la plataforma, y una instancia en *DocumentFolder* si es la primera vez que se añade un documento a esa carpeta. En esta instancia se especifica el nombre del archivo y la URL donde se almacena en Amazon S3, además de un indicador de su estado como activo, lo que permite filtrar posteriormente si el archivo se debe utilizar o no como fuente de información para el modelo.

Con cada carga de archivo, este se divide en fragmentos y se crea una instancia en la colección *Chunks* para cada uno. Cada *chunk* está vinculado al archivo y a la carpeta correspondiente mediante referencias a sus IDs respectivos en MongoDB. Cada instancia contiene el texto del fragmento y su representación vectorial, generada con el modelo de *embedding* de Amazon Titan. También se registra el modelo utilizado para generar la representación vectorial del contenido, lo que facilita futuras actualizaciones del modelo si fuera necesario.

Además, se incluye un atributo llamado *timesUsed*, que indica cuántas veces se ha empleado ese fragmento para generar una respuesta para el usuario. Este atributo es útil para evaluar la eficacia del algoritmo de recuperación, detectar posibles sesgos e identificar los archivos que generan una mayor demanda de información, permitiendo ampliar su contenido o aplicar estrategias para incrementar el conocimiento en esos temas para los clientes.

Los modelos de datos utilizados para este microservicio son los siguientes:

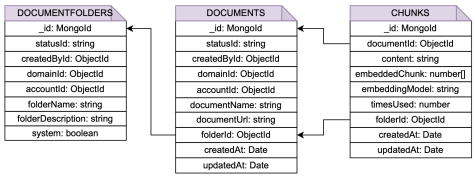

Fig. 9: Modelo de datos: DocumentFolders, Documents y Chunks.

Las tres colecciones se han creado en un clúster de MongoDB Atlas. Para poder realizar la búsqueda semántica de los fragmentos más relevantes, se ha tenido que crear y configurar un índice de búsqueda para la colección *Chunks*. Un índice de búsqueda es una estructura de datos que clasifica los datos para facilitar y acelerar su búsqueda mediante un mapeo entre términos y los documentos que los contienen.

Este índice mapea la representación vectorial de la consulta del usuario con la de los fragmentos, asignándoles una puntuación y devolviendo el texto correspondiente de los k *chunks* más relevantes para la búsqueda. La configuración del índice, tanto en MongoDB Atlas como la referencia a este en el código, se muestra en la Figura 10. Esta configuración incluye el nombre asignado al índice, el nombre del campo que contiene las representaciones vectoriales (*embeddedChunk*), el campo que almacena el texto (*content*) y el tipo de búsqueda utilizado, que en este caso es de similitud por coseno. De esta manera, el sistema realiza una búsqueda semántica utilizando las representaciones vectoriales y devuelve el contenido de los k *chunks* más relevantes, es decir, el texto de los fragmentos seleccionados.

Fig. 10: Configuración del índice de búsqueda.

6.3.2. Microservicio ”ms-model-communication”

En este apartado se explica se ha desarrollado el microservicio ”ms-model-communication” para cumplir con varios objetivos. Para el objetivo 1, se implementó una integración básica para hacer llamadas con preguntas de propósito general al modelo de Amazon: Claude, entrenado por Anthropic. Para los objetivos 2 y 3, se ha ampliado el microservicio con el agente XML de Langchain [15], adecuado para contextos conversacionales y compatible con XML *(Extensible Markup Language)*, que es el lenguaje utilizado para los modelos de Anthropic. Este microservicio gestiona la comunicación con el LLM para responder a las consultas de los usuarios y maneja el CRUD de los chats y conversaciones de cada usuario. A continuación, se presenta el modelo de datos (Figura 11) desarrollado para estas funcionalidades, una explicación detallada de su flujo de trabajo y las funcionalidades implementadas.

En términos generales, funciona de la siguiente manera: recibe una pregunta, determina a qué conversación pertenece, gestiona la memoria para proporcionar contexto al modelo de lenguaje y construye un prompt apropiado según las necesidades del usuario y del agente. Este prompt se envía al agente, que decide qué herramienta utilizar, gestiona internamente el proceso, entrega la respuesta al usuario, y guarda la información necesaria para el monitoreo y evaluación de la interacción.

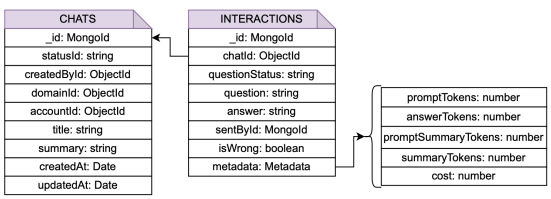

Fig. 11: Modelo de datos: Chats y Interactions.

Para este propósito, las funcionalidades más destacadas incluyen la creación de una instancia en *Chats* y otra en *Interactions* cada vez que un usuario realiza una pregunta y se recibe la respuesta del modelo. Si es una nueva conversación, se crea tanto en *Chats* como en *Interactions*; si es una conversación existente, solo se crea en *Interactions*. Además, con la primera pregunta y respuesta de cada conversación, se genera automáticamente el título del chat. Cada interacción también genera metadatos como el coste, el tiempo de respuesta, y el número de tokens de entrada y salida, que se incluyen en la información de la interacción para el análisis posterior en la interfaz de administración. Además, se gestionan los endpoints necesarios para el atributo *isWrong*, que determina si el usuario considera que la respuesta no es correcta. Tanto el historial de chats como la carga de los mensajes anteriores dentro del detalle de una conversación se realiza mediante paginación, para mejorar la experiencia del usuario.

Para dotar al agente de funcionalidades concretas, se ha decidio dividir las responsabilidades del agente en diferentes herramientas. En Langchain, existen varias herramientas predefinidas que solo requieren ser instanciadas y pasadas al agente. Sin embargo, ninguna de estas herramientas cumple con los objetivos específicos que se quieren alcanzar, se ha decidido implementar herramientas personalizadas. Estas herramientas constan de tres elementos principales: un nombre, una descripción, en inglés, y una función asociada. A continuación se hace una descripción breve de cada una de ellas:

- **Documentación Técnica:** Filtra los archivos sobre documentación técnica en el sector energético para la búsqueda semántica sobre estos. Tiene información sobre el diseño, el mantenimiento, la seguridad de sistemas y otros aspectos técnicos.
- **Documentación Administrativa:** Filtra los archivos sobre documentación administrativa en el sector energético para la búsqueda semántica sobre estos. Tiene información sobre los cumplimientos necesarios legales, regulatorios y obligaciones organizacionales en el sector energético.
- **Colección Productos:** Realiza una consulta generada por un LLM sobre la base de datos de Productos de Wattwin en Elastic Search. Esta contiene información sobre las propiedades de los productos, marca, catalogo y categoría, entre otros.
- **Colección Tareas:** Realiza una consulta generada por un LLM sobre la base de datos de Tareas de los usuarios de Wattwin en Elastic Search. Esta contiene información sobre las el rango de tiempos de las tareas, su prioridad y los usuarios asignados, entre otros.

En todos los casos, la entrada de las herramientas es la pregunta del usuario reformulada utilizando el LLM de Anthropic: Claude para simplificarla y eliminar palabras adicionales que podrían afectar la precisión del algoritmo de búsqueda. Además, se incorpora información contextual de la conversación en esta, para que la herramienta escogida tenga toda la información relevante para realizar su tarea de forma precisa.

En referencia a las dos herramientas de documentación y asumiendo que el procesamiento de documentos no estructurados ya ha sido completado y que las representaciones

vectoriales de los fragmentos asociados se han creado correctamente en la base de datos:

En ambas herramientas, cuando el usuario introduce una pregunta, esta se convierte en una representación vectorial utilizando el mismo modelo de *embedding* que se ha utilizado para los fragmentos de documentos. Luego, se utiliza la función *similaritySearchVectorWithScore* de la librería Langchain para recuperar los k fragmentos más relevantes, donde k es el número de fragmentos que se recuperan y es configurable. Esta función ejecuta una búsqueda semántica por coseno, tal y como se ha indicado en el índice de búsqueda. El texto en caracteres de los k *chunks* más relevantes es la salida de la herramienta. El agente añade en su prompt, junto con la pregunta del usuario y la conversación anterior, esta información, proporcionando una respuesta en base a esta.

La diferencia entre la herramienta técnica y la administrativa es que, si se hace una pregunta relacionada con aspectos técnicos, la función de búsqueda filtra los *chunks* de los documentos que están dentro de esa carpeta específica. Por otro lado, si la pregunta se refiere a aspectos administrativos, se filtran los *chunks* de los documentos en la carpeta correspondiente. Este enfoque permite que el chat amplíe su conocimiento incorporando datos relevantes de forma dinámica y mejore la calidad de las respuestas ofrecidas a los usuarios.

Para las herramientas que acceden a bases de datos, se ha creado un prompt específico para invocar al LLM Claude para generar una consulta para Elastic Search basada en la pregunta del usuario. Esta consulta se ejecuta directamente en la colección correspondiente: *Products* o *Tasks* y recupera la información necesaria sobre estas. La salida de la herramienta es la respuesta de esa consulta. Se ha elegido Elastic Search para realizar las consultas debido a su diseño optimizado para búsquedas rápidas y eficientes. Además, en esta tecnología las bases de datos se almacenan desnormalizadas, evitando la necesidad de realizar consultas complejas, lo cual podría aumentar la probabilidad de errores al predecir la consulta adecuada.

Dejando a un lado las herramientas, uno de los retos de este microservicio es la gestión de memoria, ya que un modelo de lenguaje utilizado dentro de un chat necesita contexto para participar en una conversación. Existen diversas estrategias para proporcionar información de contexto al modelo, que varian en tiempo de respuesta y consumo de tokens que es la unidad que utilizan los LLM para procesar el texto y, por lo tanto, también para calcular el coste del servicio. Para los modelos de Anthropic un token equivale aproximadamente a 3,5 caracteres. Cuanto más extenso sea el prompt, mayor será el coste económico y temporal asociado. Para poder realizar pruebas para decidir qué tipo de gestión de memoria se quiere utilizar, se ha implementado una clase abstracta con una función que maneja cuatro implementaciones diferentes. En la sección *7.1. Gestión de memoria.* se explican los 4 tipos de memoria implementados, las pruebas realizadas con las diferentes opciones y la decisión tomada sobre el tipo de memoria a utilizar en base al rendimiento obtenido.

El microservicio ”ms-model-communication” y la interfaz de chat se comunican en dos puntos distintos: cuando el usuario envía una pregunta por el chat y cuándo se debe mostrar por pantalla la respuesta generada por el modelo. Para conseguir un efecto streaming de esta información, al usuario enviar una pregunta, se inicia con una solicitud por deepstream, que abre una conexión con el microservicio. La respuesta del agente está configurada para que se vaya mandando en pequeños trozos de texto y mediante la conexión establecida, se puede hacer que el frontend reciba las respuestas también en pequeños fragmentos de texto. Esto permitirá enviar las respuestas de manera gradual, simulando una respuesta en tiempo real y evitando largos tiempos de espera para el usuario, mejorando así su experiencia.

Monitoreo del rendimiento

Dentro de ambos microservicios, se han integrado características en los modelos de datos para permitir a los administradores observar aspectos clave a través de la interfaz de control.

Por un lado, se ha guardado un atributo para registrar cuántas respuestas incorrectas han considerado los clientes en una conversación, junto con las respuestas y las preguntas asociadas. Esto permite identificar temas recurrentes que generan respuestas insatisfactorias y ajustar el agente para mejorar la calidad de las respuestas. Asimismo, se registra el número de tokens para las preguntas de entrada y las respuestas generadas por el modelo, proporcionando información clave sobre el uso del agente por parte de cada cliente. Esta información será especialmente útil si en el futuro se implementa un modelo de tarificación basado en el uso del usuario. También se consideran el coste y el tiempo de respuesta del modelo como métricas clave para evaluar su rendimiento.

Se han implementado funciones para mostrar un *ranking* de los fragmentos más utilizados para responder a las consultas de los usuarios, así como los documentos de los que provienen estos fragmentos. Esta funcionalidad ayuda en la mejora continua del producto al proporcionar información sobre el comportamiento del usuario.

Una vez que el producto se haya utilizado ampliamente, se evaluará la necesidad de realizar cambios significativos en la arquitectura, así como la posible creación de nuevas herramientas. Esto permitirá optimizar la herramienta según las necesidades identificadas de los usuarios finales.

7 EVALUACIÓN Y OPTIMIZACIÓN

En este apartado se expondrán las diferentes pruebas realizadas para validar los resultados de la arquitectura implementada. Esta evaluación comprenderá el ajuste de los hiperparámetros de la arquitectura, abordará la decisión sobre el tipo de memoria más adecuado para el sistema y finalizará respondiendo a la pregunta de si un sistema RAG resulta útil para mejorar el rendimiento de un LLM y proporcionar un mejor servicio al usuario.

7.1. Gestión de memoria

Uno de los retos enfrentados en el proyecto ha sido cómo gestionar la información de la conversación con el usuario para que el modelo no pierda el contexto de esta. Se han probado las siguientes formas de manejar esta información:

- Conversación entera: Se proporciona al modelo todos los mensajes anteriores ofreciendo así el contexto completo de la conversación.

- Últimos n mensajes: Se pasan solo los últimos n mensajes de la conversación. En esta prueba, n está establecido en 3 mensajes.
- Resumen de cada interacción: Después de cada interacción (pregunta-respuesta), se genera un resumen con IA.
- Resumen de interacciones anteriores y últimos n mensajes: Esta opción combina la generación de resúmenes con la inclusión de los últimos n mensajes literales en el prompt. El valor de n se ha establecido en 3.

En primer lugar, se ha comprobado si el modelo es capaz de mantener la coherencia de la conversación en todos los casos. Para esto, se ha diseñado un conjunto de 10 preguntas donde la primera mencionaba el tema central de la conversación: el café, y las siguientes 9 hacían referencia a aspectos relacionados con el café pero sin mencionarlo explícitamente en la pregunta, por ejemplo: "¿Qué beneficios para la salud tiene?". En todos los casos, la conversación continuó de forma coherente, sin perder el contexto del tema principal.

También se ha analizado el aumento del número de tokens en el prompt correspondientes a la memoria a medida que avanzaba la conversación (Figura 20). Este incremento en los tokens afecta directamente al tiempo de procesamiento de la consulta y, en consecuencia, al tiempo de respuesta del agente, así como al coste asociado.

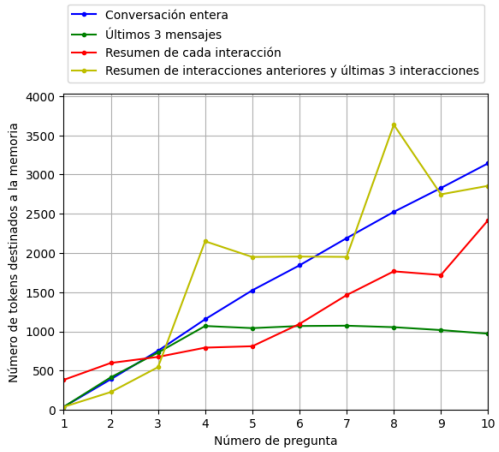

Fig. 12: Variabilidad del número de tokens.

De esta prueba se pueden extraer las siguientes conclusiones:

- La conversación se ha seguido de forma coherente en todos los casos.
- Entre los tipos de memoria evaluados, la opción de "Últimos n mensajes" (línea verde) parece ser la mejor opción, debido a que se mantiene estable a partir de la pregunta 4. Esto variará según el valor de n, pero igualmente podremos asegurar un valor prácticamente constante del número de tokens pasados al modelo, asegurando así un control preciso sobre este.
- Para las otras opciones de memoria, podemos observar que a medida que la conversación se extiende, el número de tokens tiene tendencia a aumentar, lo que supondría que creciera indefinidamente si el usuario hace uso de un solo chat para todas sus consultas. Esto afectaría negativamente al rendimiento del modelo y al coste asociado a cada llamada.

7.2. Recuperación y Generación

Dentro del proceso que sigue el agente para responder a una consulta, se distinguen dos fases: la recuperación y la generación. En la fase de recuperación se evalúa la adecuación de los fragmentos recuperados (contexto) en la búsqueda semántica y en la fase de generación se verifica si el contexto proporcionado ha sido útil para generar una respuesta correcta. Estos aspectos se evaluarán utilizando diferentes métricas de la librería RAGAS, un framework diseñado para evaluar sistemas RAG mediante el uso de un modelo de OpenAI.

Para evaluar estos aspectos, se ha creado un conjunto de datos con: 7 preguntas (Apéndice A), los fragmentos de texto recuperados para contestar cada pregunta, la respuesta generada por el modelo y la respuesta considerada correcta (*ground truth*) escrita por una persona del equipo de Learning de la empresa. Todas las métricas varían de 0 a 1, donde 1 indica los mejores resultados y 0 los menos satisfactorios. Las preguntas se han generado mediante los documentos técnicos: BOE-A-2023-23120.pdf y BOE-A-2023-12786-PRVE.pdf.

Tanto en la fase de recuperación como en la de generación hay dos hiperparámetros clave: *k*, que determina cuántos fragmentos se quieren devolver en la búsqueda semántica, y *chunk_size*, que se refiere al tamaño de estos fragmentos. En otros contextos, estos dos valores no tienen por qué estar vinculados, pero en el caso concreto de este proyecto sí lo están, debido a que se ha establecido en 3000 el número de tokens que se quiere destinar a la documentación adicional proporcionada en el prompt del modelo.

Con este límite establecido, se han realizado diferentes pruebas variando *k* y el *chunk_size* siguiendo la tabla mostrada en el Apéndice A.1). La teoría indica que fragmentos de texto más pequeños dan información semántica más concreta y, en consecuencia, se encontrarán mejor los fragmentos semánticamente similares a la pregunta del usuario. El objetivo de estas pruebas es explorar las tendencias de las diferentes métricas al variar el número de fragmentos recuperados y su tamaño, y ver si esto afecta al rendimiento del modelo.

7.2.1. Recuperación: Fine-tuning de los hiperparámetros

Las métricas utilizadas para evaluar la fase de recuperación son las siguientes:

- ***Context precision***: Evalúa si los ítems más relevantes dentro del *ground truth* están presentes en los fragmentos de contexto y en qué posición. Idealmente, los fragmentos más relevantes deberían estar al principio del contexto con la mejor puntuación.
- ***Context recall***: Determina si la información de los fragmentos recuperados existe dentro del *ground truth*, es decir, si se recupera toda la información relevante contenida en el *ground truth*.

7.2.2. Generación: Fine-tuning de los hiperparámetros

Las métricas utilizadas para evaluar la fase de generación de respuestas son:

- ***Faithfullness***: Evalúa en qué medida la información de contexto proporcionada se ha utilizado para la generación de la respuesta por parte del LLM.
- ***Answer Relevancy***: Calcula en qué medida la respuesta generada por el modelo responde adecuadamente a la pregunta original, considerando si la respuesta es coherente con la pregunta planteada y el contexto recuperado.

En la Figura 13 se presenta un resumen de las cuatro métricas mencionadas anteriormente para las fases de recuperación y generación.

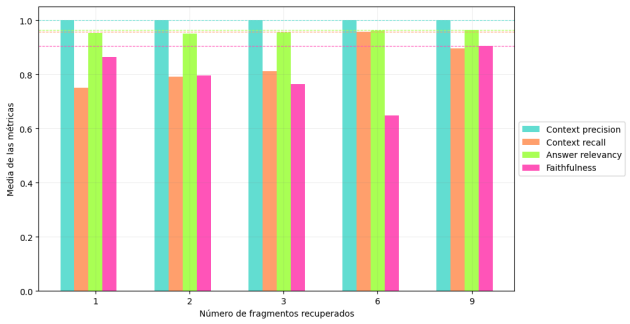

Fig. 13: Métricas de recuperación y de generación.

De esta gráfica se deducen las siguientes conclusiones:

- La métrica *Context Precision* no nos aporta información.
- En promedio, vemos mejoras cuándo aumentamos la k y disminuimos el $chunk\_size$, lo que confirma la hipótesis formulada teóricamente.
- Si comparamos los valores de las métricas cuándo $k = 6$ y $k = 9$, podemos ver que el *Context Recall* es mejor para $k = 6$, lo que significa que la información de los fragmentos recuperados existe dentro del *ground truth* en mayor medida. Sin embargo, la métrica *Faithfulness*, que indica como de fiel es la respuesta del modelo al contexto recuperado, es superior en $k = 9$. Dado que la diferencia en el *Context Recall* entre ambos valores es bastante pequeña y se prioriza que el modelo dé respuestas correctas y basadas principalmente en el contexto recuperado, se ha escogido el valor óptimo de $k = 9$ y $chunk\_size = 333tokens$.

En el Apéndice A.3 y A.4 se detallan las gráficas por pregunta y métrica, mostrando las tendencias de la relación entre k y el tamaño de los *chunks* para una visualización más detallada.

En cuanto a la generación, otro hiperparámetro relevante es la temperatura del modelo, que afecta el grado de aleatoriedad en las respuestas generadas. En este proyecto, se prefiere un valor cercano a 0 para que el modelo sea más determinista, lo cual promueve respuestas menos creativas. Por esta razón, se ha elegido un valor de 0.2. No se ha optado por el valor 0 debido a que se permite al usuario generar una segunda respuesta ante una misma pregunta, lo que permite un margen de ajuste en la respuesta.

7.2.3. Evaluación End-to-End

En este apartado se comparará el rendimiento del LLM con su entrenamiento base en relación con el rendimiento utilizando la arquitectura RAG mediante el agente. Con esto obtendremos datos concretos sobre el valor que aporta la integración de documentación adicional en el LLM.

Para esta evaluación, el equipo de *Learning* de la empresa ha formulado 12 preguntas (A.5) que consideran que el modelo debe ser capaz de responder, junto con las respuestas correctas que se utilizarán como *ground truth*. Se compararán las respuestas generadas por el modelo con y sin RAG con las respuestas de una persona humana. En este caso se utilizarán las siguientes métricas:

- ***Answer Semantic Similarity***: Se realizará una comparación de la similitud semántica entre el *ground truth* y la respuesta del modelo.
- ***Answer Correctness***: Además de la métrica de similitud semántica, se incluirá el F1 score, que mide la precisión y la corrección factual de la respuesta.

Estas métricas permitirán evaluar si la integración de la arquitectura RAG mejora significativamente la calidad de las respuestas generadas. A continuación, se presenta una gráfica resumen con ambas métricas y su promedio, separadas en preguntas relacionadas con documentación y preguntas relacionadas con bases de datos. Además, en el Apéndice A.6 se proporciona información más detallada sobre las diferentes métricas.

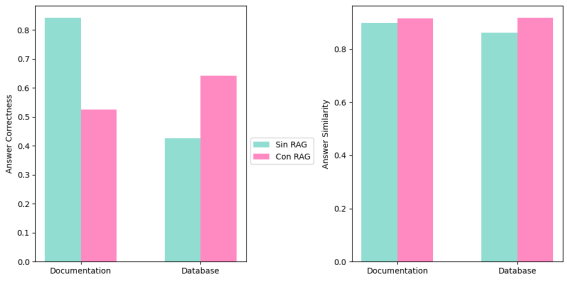

Fig. 14: LLM base vs RAG.

De esta imagen se extraen las siguientes conclusiones:

- Para las preguntas de documentación, se observa una mejora en la métrica de *Answer Correctness* con el sistema sin RAG en comparación con el sistema con RAG. Esto podría explicarse debido a que estas preguntas sobre documentación son públicas y el modelo pudo haber sido entrenado con estos datos, resultando en respuestas más precisas y correctas. En cuanto a la métrica *Answer Semantic Similarity*, el sistema RAG ha obtenido puntuaciones ligeramente mejores, aunque no se ha observado una mejora significativa.
- En el caso de las preguntas relacionadas con bases de datos, que no son información pública, observamos que la arquitectura RAG para la métrica *Answer Correctness* supone una mejora considerable en comparación con el modelo base, debido a que el modelo base

no tiene acceso a esa información. Además, haciendo una evaluación sin la librería RAGAS, las respuestas del LLM base carecen totalmente de la información relevante sobre bases de datos, mientras que con RAG se proporcionan respuestas correctas.

8 Conclusiones y futuras mejoras

El proyecto tenía dos metas principales: implementar un agente inteligente capaz de manejar diversos tipos de consultas y establecer las bases de la IA en Wattwin.

Primeramente, es importante destacar que los objetivos del proyecto se han cumplido satisfactoriamente, con la excepción de la interfaz del Objetivo 5 para la visualización de estadísticas sobre el uso del chat para los administradores.

Los resultados obtenidos han sido positivos, demostrando la utilidad de la arquitectura RAG dentro de una empresa que trabaja con datos privados. Aunque en algunos casos se ha observado que las métricas favorecían más al modelo base en las preguntas relacionadas con documentación, para las preguntas sobre datos actuales y propios de la empresa, con los que el modelo no ha sido entrenado, la incorporación de RAG ha mostrado mejoras significativas. Por este motivo, se seguirán utilizando las herramientas actuales para gestionar la documentación nueva que el modelo no conozca y se se seguirá trabajando en el sistema para conseguir hacer más exacta la recuperación del contexto y uso de este en las respuestas del agente.

Uno de los desafíos principales de este proyecto ha sido la constante y rápida evolución en el uso de los LLM en diferentes sistemas y arquitecturas, lo que ha dificultado adherirse a una única forma de implementación. La técnica con agentes definida ha resultado ser una buena solución actualmente, pero podría ser reemplazada por una nueva arquitectura en un futuro cercano.

Un aspecto a mejorar es la personalización del agente para cada cliente. Actualmente, los administradores son los responsables de cargar los documentos correspondientes en las dos carpetas ofrecidas por la plataforma, lo que limita la capacidad de los usuarios para decidir qué documentación utilizar en las conversaciones con el agente. Como mejora, se plantea permitir a los usuarios introducir nuevas carpetas y documentos para alimentar el modelo en tiempo de ejecución mediante herramientas dinámicas. Esto implicaría crear tantas herramientas como carpetas tenga el usuario y permitirles consultar al agente sobre su propia documentación. Además, Wattwin quiere explorar más aplicaciones de IA, como la generación de respuestas a correos electrónicos basadas en el contexto y el análisis de sentimiento.

La inteligencia artificial está en constante crecimiento y ofrece numerosas oportunidades de mejora. Este proyecto es solo un pequeño ejemplo de todo lo que se puede lograr con una integración eficiente de la inteligencia artificial.

9 Agradecimientos

Agradecer principalmente a mis tutores, Ernest Valveny (UAB) y Mario Villacorta (Wattwin), por proporcionarme el equilibrio ideal entre libertad y orientación en las diferentes etapas del trabajo, ofreciéndome ayuda para alcanzar los objetivos. También a la empresa Wattwin por brindarme la oportunidad de desarrollar este proyecto y confiar en mis habilidades. Por último, quiero dar las gracias a mi familia y a mi pareja por su apoyo durante todo el proceso, y por darme su opinión sincera aunque no siempre entendieran de lo que les estaba hablando.

Referencias

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[2] Codificando bits. (2020) REDES TRANSFORMER. [Online]. Available: https://www.youtube.com/watch?v=Wp8NocXW_C4

[3] Xataka Basics. (2024) Chatgpt: qué es, cómo usarlo y qué puedes hacer con este chat de ia. [Online]. Available: https://www.xataka.com/basics/

[4] Konfuzio. (2023) Los límites de los llm y cómo los remedia el rag. [Online]. Available: https://konfuzio.com/es/limites-llms-recuperacion-generacion-aumentada/

[5] P. Lewis, J. Weston, A. Bordes, J. Uszkoreit, S. Chopra, and A. M. Rush, “Retrieval-augmented generation for knowledge-intensive nlp tasks,” 2020. [Online]. Available: https://arxiv.org/abs/2005.11401

[6] (2024) Salesforce. [Online]. Available: https://www.salesforce.com/

[7] (2024) Hubspot. [Online]. Available: https://www.hubspot.es/products/artificial-intelligence

[8] (2024) Mongodb atlas. [Online]. Available: https://www.mongodb.com/es/products/platform/atlas-database

[9] (2024) Nestjs. [Online]. Available: https://nestjs.com/

[10] Amazon Web Services. (2024) Amazon bedrock. [Online]. Available: https://aws.amazon.com/es/bedrock/

[11] (2023) Amazon s3. [Online]. Available: https://aws.amazon.com/es/s3/

[12] (2024) LangChain Library. [Online]. Available: https://js.langchain.com/docs/get_started/introduction

[13] (2023) Ragas. [Online]. Available: https://docs.ragas.io/en/stable/index.html

[14] (2024) Agents. [Online]. Available: https://js.langchain.com/v0.1/docs/modules/agents/

[15] (2024) Xml agent. [Online]. Available: https://js.langchain.com/v0.1/docs/modules/agents/agent_types/xml/

Apéndice

A Evaluación y optimización del agente

A.1. Preguntas para la el cálculo de métricas de recuperación y generación

Pregunta	Tipo
¿Existe alguna limitación en cuanto a la potencia máxima instalada de punto de recarga segun el real Decreto?	Simple
¿A partir de qué fecha pueden solicitarse las ayudas del Plan Moves III?	Simple
¿Que se considera como costes subvencionables del Plan Moves III?	Simple
¿Cual es el límite de ayuda por cada expediente?	Simple
¿Existe alguna limitación de porcentaje de ayuda para instalaciones mayores a 50 kW?	Simple
¿Cual es la documentación requerida para justificar la instalación de punto de recarga y asi poder acceder a las ayudas de plan Moves III?	Multicontexto
¿Las categorías M1 y N1 de vehículos eléctricos son tambien subvencionables?	Multicontexto

A.2. Tabla de pruebas de hyperparámetros: k y tamaño de fragmento

Tamaño de fragmento (tokens)	k: nº de fragmentos recuperados
3000	1
1500	2
1000	3
500	6
333	9

A.3. Gráficas sobre las métricas en la fase de recuperación

A.3.1. Context Recall

Esta métrica mide en qué medida el contexto recuperado corresponde con el *ground truth* (GT). Para ello se compara cada frase del GT con el contexto para ver si está se puede atribuir o no al contexto. Lo ideal sería que todas las frases del contexto se pudiesen encontrar en el GT. La fórmula es la siguiente:

$$\text{Context Recall} = \frac{\text{Número de frases en el GT que pueden ser atribuidas al contexto}}{\text{Número de frases totales en el GT}}$$

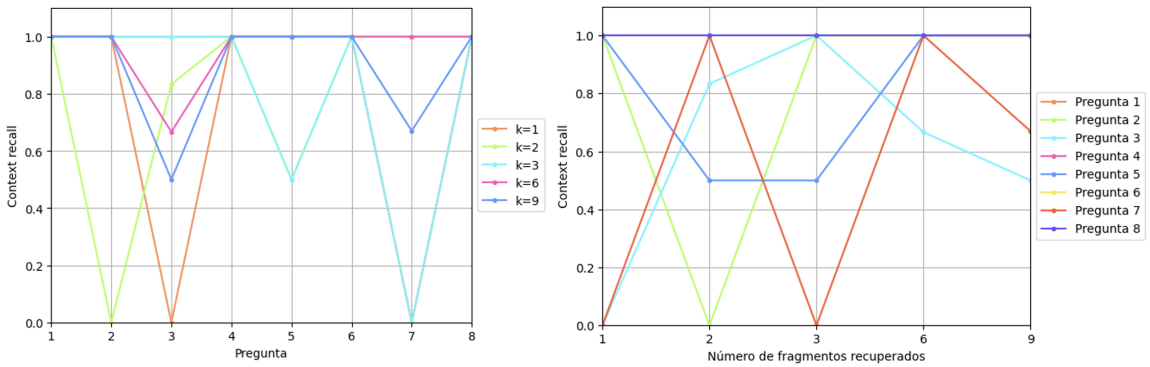

Fig. 15: Gráficas sobre la métrica: Context recall.

A.3.2. *Context precision*

Esta métrica hace referencia a si los elementos relevantes están presentes en el contexto y con qué importancia. Los *chunks* se recuperan en orden de importancia; por lo tanto, los fragmentos con mayor relevancia semántica se situarán primero. En este caso, tiene sentido observar qué sucede cuando se recuperan más de un fragmento. Es decir, para $k = 1$,

solo nos interesa si el fragmento recuperado está presente en el contexto, no su importancia. Se calcula mediante las siguientes dos fórmulas:

$$\text{Precision@k} = \frac{\text{true positives@k}}{(\text{true positives@k} + \text{false positives@k})}$$

Los *true positives* hacen referencia a los fragmentos que corresponden correctamente a la información del *ground truth* (GT), mientras que los *false positives* son aquellos que han sido recuperados pero no están incluidos dentro del GT.

$$\text{Context precision@K} = \frac{\sum_{k=1}^{K} (\text{Precision@k} \times v_k)}{\text{Número total de items relevantes en los K } K \text{ mejores resultados}}$$

K es el número de *chunks* dentro de la celda del contexto y v_k es la relevancia atribuida según la posición dentro de los *chunks*.

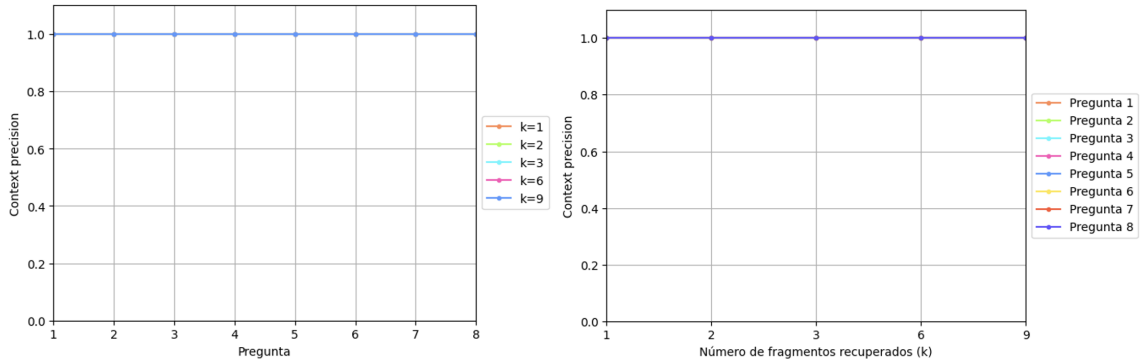

Fig. 16: Gráficas sobre la métrica: *Context precision*

A.4. Gráficas sobre las métricas en la fase de recuperación

A.4.1. *Faithfullness*

Esta métrica calcula la consistencia de los hechos, que se refiere a la coherencia de la respuesta proporcionada por el LLM con respecto a los hechos reales. En este caso, se evalúa utilizando la respuesta del modelo y los contextos recuperados para determinar si el LLM ha utilizado la información de los contextos para responder o no. Esta métrica calcula lo mismo que el *Context Recall*, pero para la respuesta generada por el modelo en lugar del *ground truth* (GT). Se calcula de la siguiente manera:

$$\text{Faithfullness} = \frac{\text{Número de frases en la respuesta que pueden ser inferidas del contexto}}{\text{Número de frases totales en la respuesta generada}}$$

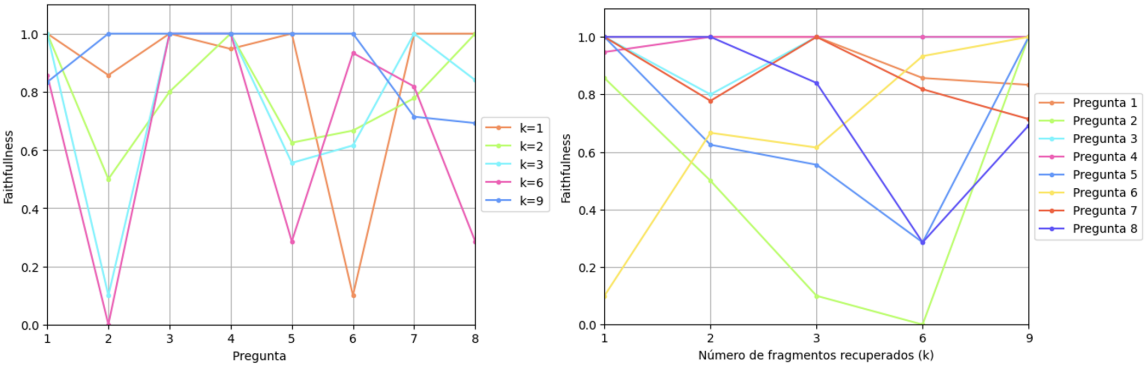

Fig. 17: Gráficas sobre la métrica: *Faithfullness*.

A.4.2. *Answer Relevancy*

Esta métrica utiliza un LLM para realizar ingeniería inversa, es decir, a partir de la respuesta generada debe generar n preguntas que podrían dar como resultado esa respuesta. Por defecto, n es 3 y no se ha modificado para realizar las pruebas. Con estas 3 preguntas nuevas generadas por el LLM a partir de la respuesta, se calcula la similitud semántica por coseno respecto a la pregunta original. La idea detrás de este cálculo es que si la respuesta es relevante, las preguntas generadas por el modelo serán similares a la pregunta original. Si esto no ocurre, podría significar que la respuesta generada no responde de manera coherente a la pregunta original. La fórmula que se utiliza es la siguiente:

$$\text{Answer Relevancy} = \frac{1}{N}\sum_{i=1}^{N} \cos\left(Eg_i, E_o\right)$$

Las notaciones Eg_i y E_o se refieren a las representaciones vectoriales de la respuesta generada i y de la respuesta original, respectivam

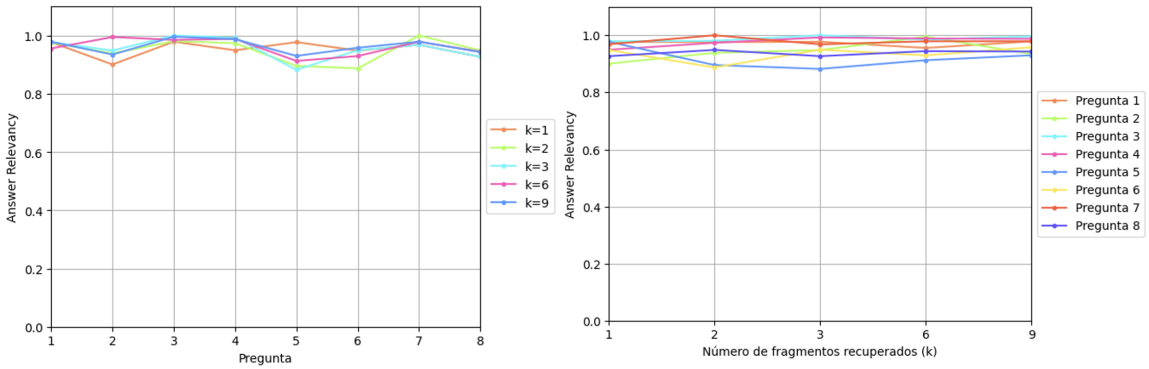

Fig. 18: Gráficas sobre la métrica: *Answer Relevancy*.

A.5. Preguntas para el cálculo de métricas sobre la respuesta

Pregunta	**Tipo**
Existe alguna limitación en cuanto a la potencia máxima instalada de punto de recarga segun el real Decreto 821/2023?	Documentación Técnica
Cual es el límite de ayuda por cada expediente?	Documentación Técnica
Existe alguna limitación de porcentaje de ayuda para instalaciones mayores a 50 kW?	Documentación Técnica
Que se considera como costes subvencionables del Plan Moves III?	Documentación Administrativa
Cual es la documentación requerida para justificar la instalación de punto de recarga y asi poder acceder a las ayudas de plan Moves III?	Documentación Administrativa
Las categorías M1 y N1 de vehículos eléctricos son tambien subvencionables?	Documentación Administrativa
Búscame todos los módulos que cumplan con 500 Wp del catálogo JA Solar.	Productos
Compárame las características del producto REACT-4.6-TL y el REAT-4.6-TL-OUTD-S.	Productos
De todo el catálogo de productos activos, sácame aquellos productos que sean de la marca Circuitor y categoria Aro Empotrado.	Productos
De todo el catálogo sácame aquellos Analizadores de red que sean de la marca Salicru y ordénamelos de menor a mayor coste.	Productos
¿Qué tareas tengo para hoy?	Tareas
¿En qué orden debería hacer las tareas de hoy según su prioridad?	Tareas

A.6. Gráficas sobre las métricas de evaluación End-to-End

A.6.1. *Answer Semantic Similarity*

En este caso, se calcula la similitud semántica de la respuesta respecto al *ground truth* (GT), lo que permite determinar si la respuesta generada por nuestra arquitectura RAG difiere significativamente de la respuesta esperada. La fórmula que se utiliza es la siguiente:

$$\text{Answer Semantic Similarity} = \cos\left(E_{GT}, E_{answer}\right)$$

Las notaciones E_GT y E_answer se refieren a las representaciones vectoriales del GT i y de la respuesta original, respectivamente.

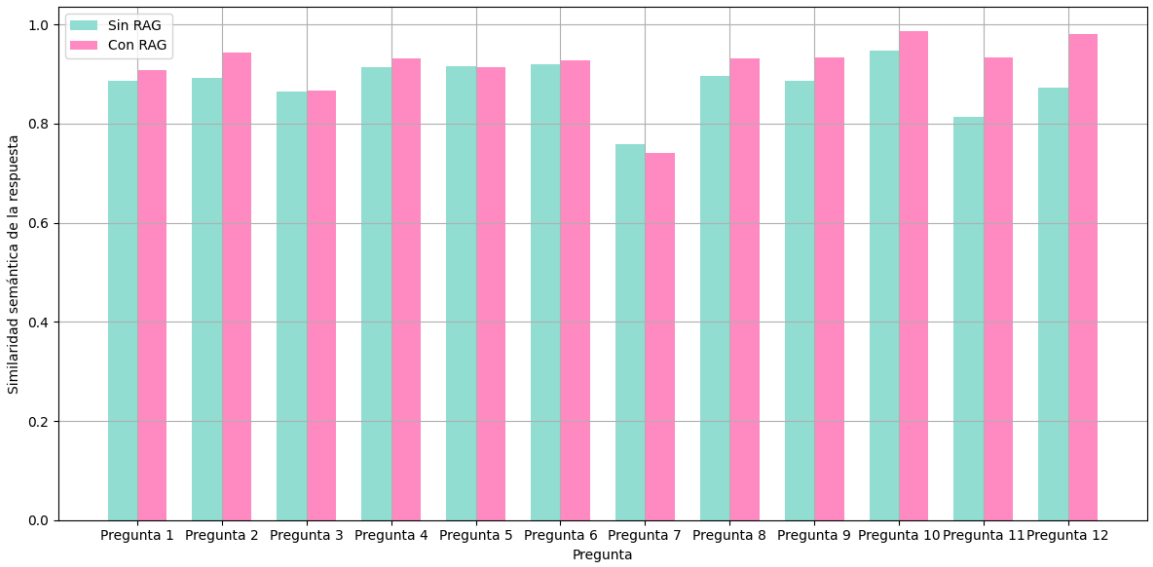

Fig. 19: Gráficas sobre la métrica: *Answer Semantic Similarity*.

A.6.2. *Answer Correctness*

Por último, se ha calculado la exactitud de la respuesta, que se refiere a la precisión de la respuesta generada por el modelo en comparación con el *ground truth* (GT). En esta métrica, además del cálculo de la similitud semántica, se incluye la similitud de hechos, que se refiere al F1 score. Se utilizan las siguientes ecuaciones:

$$\text{F1 score} = \frac{\text{True positives}}{(\text{True positives} + 0{,}5 * (\text{False positives} + \text{False negatives}))}$$

$$\text{Answer Correctness} = 0{,}5 * \text{F1 score} + 0{,}5 * \text{Answer Semantic Similarity}$$

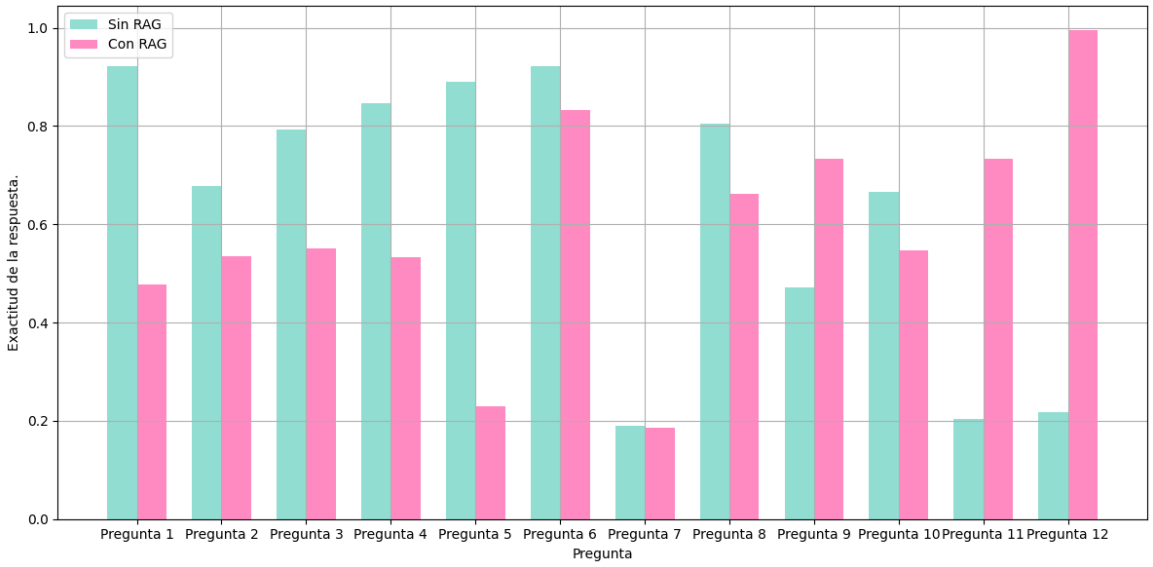

Fig. 20: Gráficas sobre la métrica: *Answer Correctness*.