Universidad Politécnica
de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Creación de una Interfaz para Extraer Conocimientos de Artículos Científicos sobre Química Solar

Autora: Ioana Alexandra Faje

Tutor: Daniel Garijo Verdejo
Cotutora: Maria Poveda Villalón

Madrid, noviembre 2024

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática
Título: Creación de una Interfaz para Extraer Conocimientos de Artículos Científicos sobre Química Solar
Noviembre 2024

Autor: Ioana Alexandra Faje

Tutor:
Daniel Garijo Verdejo
Departamento de Inteligencia Artificial
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

La cantidad de artículos científicos en el área de la química solar ha aumentado de manera importante. Esto tiene la ventaja de encontrar más soluciones para aprovechar la energía solar, pero también tiene una gran desventaja: al haber tal cantidad de información, los investigadores a la hora de buscar lo que necesitan, requieren de mucho tiempo y esfuerzo.

Este trabajo de fin de grado tiene como objetivo crear una interfaz web que ayude a los investigadores con el análisis de artículos científicos. Su principal función es extraer de manera automática los datos concretos que ellos necesitan de esos artículos. Para lograrlo, la interfaz se conectará a un sistema previamente desarrollado por el Ontology Engineering Group (OEG), el cual utiliza avanzados modelos de Inteligencia Artificial, como los modelos de lenguaje de gran tamaño (LLM). Estos modelos analizan la información contenida en los artículos y entregan, de manera precisa, los datos más relevantes para el usuario.

Esta interfaz no solo ayudará en la búsqueda de información más reciente de una forma rápida, sino que además evitará que los investigadores pierdan tiempo en la búsqueda manual de dicha información.

Con este proyecto, esperamos ayudar a los investigadores a la hora de encontrar las soluciones más rápido y de esta forma haya avances en dicha área. Además, simplifica la manera de investigar y trabajar en equipo para que toda la información se pueda ver de una forma más simple y rápida.

Abstract

The number of scientific articles in the area of solar chemistry has increased significantly. This has the advantage of finding more solutions for harnessing solar energy, but it also has a major disadvantage: with so much information, it takes a lot of time and effort for researchers to find what they need.

The aim of this final degree project is to create a web interface that helps researchers with the analysis of scientific articles. Its main function is to automatically extract the specific data they need from these articles. To achieve this, the interface will be connected to a system previously developed by the Ontology Engineering Group (OEG), which uses advanced Artificial Intelligence models, such as Large Language Models (LLM). These models analyze the information contained in the articles and deliver the most relevant data to the user with precision.

This interface will not only help in finding the latest information quickly but will also prevent researchers from wasting time manually searching for such information.

With this project, we hope to help researchers to find solutions faster and thus make progress in this area. In addition, it simplifies the way of research and teamwork so that all information can be seen in a simpler and faster way.

Agradecimientos

A mi madre,
Por ser la persona que más ha influido en mi vida. Aunque ya no está conmigo físicamente, sus valores y sus enseñanzas me acompañan cada día, recordando que, incluso cuando me equivoco y cometo errores, siempre hay que aprender y seguir adelante. Le prometí que acabaría esta carrera y hoy, con este trabajo, cumplo esa promesa. Este logro es tanto suyo como mío.
Te quiero, mamá.

A mi padre,
Por ser la persona que siempre ha creído en mí. Su paciencia, apoyo y constante insistencia en que retomara la carrera han sido fundamentales para que hoy esté aquí terminando este trabajo. Gracias por enseñarme, con tu ejemplo, que nunca es tarde para cumplir nuestras metas.
Este logro también es tuyo.

A mi hermana,
Mi pilar en la vida. Tu forma de quererme, cuidarme y estar siempre a mi lado ha sido esencial en este camino. Juntas hemos enfrentado este camino, y paso a paso, seguimos logrando metas juntas. Gracias por ser mi refugio y una fuente de amor.

A Pablo,
Por ser mi guía en la universidad. Tu dedicación y confianza en mí han hecho que este camino sea más llevadero y hoy pueda estar entregando este trabajo.
Gracias.

A mi familia elegida,

Gracias por acompañarme en este viaje. Este viaje ha sido más largo de lo esperado, pero cada palabra de ánimo y gesto de ayuda han hecho que nunca pierda de vista la meta. Gracias a todos y a todas.

Tabla de contenidos

1. Introducción

1.1 Motivación

Los científicos están cada vez más desbordados por el volumen de artículos que se publican. Un estudio realizado por Mark A. Hanson et al. [1], destaca que el rápido incremento en la cantidad de publicaciones científicas anuales genera una sobrecarga de información para los investigadores, complicando la tarea de analizar y comparar de manera eficiente los estudios más recientes en áreas específicas.

En el ámbito de la química solar, esta situación es especialmente crítica, ya que la comparación y comprensión de los estudios es fundamental para el avance en la investigación. En el área de la química solar, el objetivo es investigar nuevos materiales y técnicas que permitan captar energía solar de manera efectiva. Es por este motivo que los investigadores necesitan de la habilidad para identificar, analizar y extraer de forma rápida los conocimientos más importantes.

Estudios recientes han demostrado que los modelos de lenguaje de gran tamaño (LLM) ajustados específicamente para textos científicos son capaces de extraer información estructurada de artículos complejos, lo que facilita la identificación de datos relevantes y mejora significativamente la precisión de las respuestas obtenidas. Por ejemplo, Alexander Dunn et al. [2] analizaron el uso de LLM para la extracción estructurada de información y demostraron que estos modelos son capaces de extraer con precisión, registros útiles de conocimientos científicos complejos para tareas representativas de la química de materiales. Según este estudio, este planteamiento representa una forma sencilla, accesible y muy flexible para obtener grandes volúmenes de información. Gracias a esta tecnología, los investigadores pueden focalizarse en la información concreta de forma rápida y precisa, simplificando la comparación de investigaciones y agilizando el progreso en áreas como la química solar.

Dado que la cantidad de publicaciones está en aumento, los investigadores requieren instrumentos que les permitan economizar tiempo y disminuir el esfuerzo necesario para revisar los artículos más novedosos. Mark A. Hanson et al. [1] destacan que la necesidad de soluciones tecnológicas innovadoras es crítica para aliviar la presión que supone la continua proliferación de literatura científica. Alexander Dunn et al. [2] subrayan que los LLM ofrecen una manera eficiente de extraer y organizar información científica, reduciendo el tiempo necesario para analizar grandes cantidades de

datos.

Este trabajo sugiere una solución innovadora: el desarrollo de una interfaz web que integra un sistema de pregunta-respuesta basado en modelos de lenguaje de gran tamaño (Large Language Models, LLM). Con este sistema, los investigadores podrán obtener del artículo científico el tipo de catalizador y co-catalizador que se usa, la fuente de luz empleada, el tipo de reactor, el medio de reacción y el modo de operación. Este sistema no permite que el usuario haga una pregunta concreta, sino que devuelve de forma automática estos datos, siempre y cuando estos datos estén mencionados en el artículo.

Se espera que esta plataforma ayude a mejorar la manera en que se accede y gestiona la información científica. Gracias a los modelos de lenguaje de gran tamaño, el sistema podría mejorar la búsqueda y el análisis de grandes cantidades de datos científicos, para responder a las preguntas de forma clara y rápida. Se espera que esta herramienta no solo haga más eficiente la revisión de estudios, sino que también impulse el desarrollo de la ciencia en áreas altamente especializadas, como la química solar, donde es crucial aplicar los conocimientos de manera ágil.

1.2 Objetivos

1.2.1 Objetivo general

En resumen, el objetivo principal de este proyecto consiste en crear una interfaz web que facilite a los investigadores el acceso rápido y de una forma simple a datos importantes en el área de la química solar. Esta interfaz, diseñada en Streamlit, devuelve respuestas a preguntas concretas y justificaciones de forma detallada, mostrando los párrafos de los que se extrae la información. Para alcanzar este objetivo, la interfaz se conecta a un sistema que fue desarrollado anteriormente por el Ontology Engineering Group (OEG), sistema que se encarga de recibir los artículos científicos en formato PDF, analizar el texto y devolver la respuesta en formato JSON usando técnicas de procesamiento de lenguaje natural y modelos de Inteligencia Artificial.

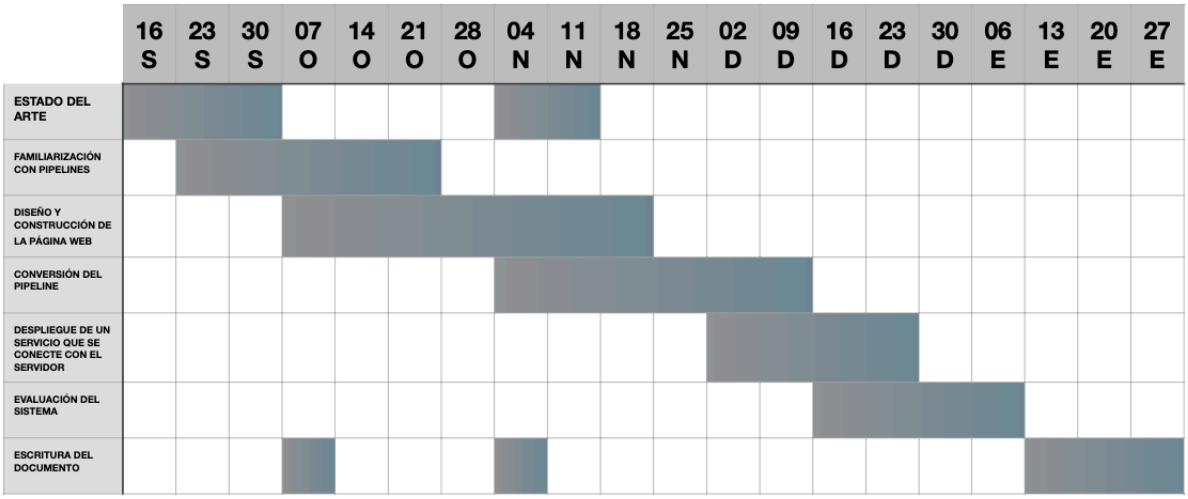
1.2.2 Objetivos específicos

- Investigar los fundamentos de los sistemas Question Answering QA para poder comprender su funcionamiento.
- Investigar y entender los sistemas de extracción Retrieval Augmented Generation (RAG) y modelos de lenguaje de gran tamaño (LLM) ya

implementados por el Ontology Engineering Group (OEG), con la finalidad de entender su función para prepararme para su integración.

- Diseño de la interfaz web utilizando Streamlit.
- Diseño de una API que permite la conexión de la interfaz web con el servidor para acceder al procesamiento de lenguaje natural basado en LLM.
- Despliegue de un servicio que se conecte con el servidor
- Analizar y documentar los resultados obtenidos.

1.2.3 Diagrama de Gantt



S = Septiembre; O = Octubre; N = Noviembre; D = Diciembre; E = Enero

A continuación se explica la estructura de la memoria a partir de este punto. Dicha memoria tiene el siguiente formato:

- Apartado 2: Estado del arte.

  En este apartado, lo primero que vamos a hacer es definir lo que es la Inteligencia Artificial y algunos de sus campos. Después hablaremos de los sistemas Question Answering y sus avances, ya que este trabajo se basa en un sistema similar. Por último, definiremos algunos conceptos necesarios para la comprensión del lector.

- Punto 3: Desarrollo.

  Es la parte técnica que se centra en todo el desarrollo del trabajo. En este punto es donde vamos a explicar cómo se ha conseguido el desarrollo de la interfaz web. Describiremos paso a paso cómo hemos ido creando el frontend en Streamlit, y su funcionamiento.

- Punto 4: Evaluación , comparación y limitaciones del proyecto

  En el punto 4 comentaremos las distintas pruebas que hemos hecho durante el desarrollo de la interfaz web, las complicaciones surgidas y sus principales limitaciones.

- Punto 5: Análisis de impacto.

  Este punto describe el impacto que tiene esta herramienta, es decir, cómo podría influir en la sociedad. Este análisis tiene como objetivo, mostrar el valor que tiene este trabajo y también las posibles repercusiones económicas, tecnológicas, sociales, etc.

- Punto 6: Trabajo futuro.

  En trabajo futuro hablaremos de las posibles mejoras y nuevas funcionalidades que podrían añadirse en versiones futuras del sistema. Aunque el desarrollo actual funciona correctamente, se puede mejorar esta interfaz para ampliar el impacto del proyecto.

- Punto 7: Discusión de los objetivos.

  En este apartado vamos a enumerar todos los objetivos que se nos han pedido, tanto el objetivo general como los objetivos específicos para comprobar si los hemos alcanzado o no.

- Punto 8: Conclusiones.

  Este último apartado se centra en las conclusiones finales y los puntos débiles y los puntos fuertes a nivel personal de este proyecto.

2. Estado del arte

2.1 Pequeña introducción a la Inteligencia Artificial (IA)

La Inteligencia Artificial es una disciplina que tuvo sus inicios en 1950 con el trabajo del matemático británico Alan Turing. En su famoso artículo "Computing Machinery and Intelligence", Turing introdujo el concepto del "test de Turing" como una herramienta para evaluar si una máquina puede demostrar un comportamiento inteligente.. Según se describe en el trabajo de Amirhosein Toosi et al. [3], este test requería que los sistemas contarán con capacidades clave como el procesamiento del lenguaje natural y el razonamiento automatizado. Aunque inicialmente no tuvo mucho éxito, hoy en día, la Inteligencia Artificial se ha convertido en una parte esencial de nuestra vida cotidiana, presente en sistemas como el reconocimiento facial, los videojuegos y los asistentes virtuales, etc.

La Inteligencia Artificial, conocida más bien por sus siglas "IA", es un campo muy amplio y considerado uno de los más enigmáticos de la ciencia. Según Amirhosein Toosi et al. [3], el término "IA" fue acuñado en el verano de 1956 durante un taller de dos meses celebrado en el Dartmouth College, liderado por John McCarthy, Marvin Minsky, Claude Shannon y Nathaniel Rochester, quienes exploraron intereses comunes en teoría de autómatas, redes neuronales y ciencia cognitiva. McCarthy definió la IA como "la ciencia y la ingeniería de fabricar máquinas inteligentes", marcando un hito en la historia de esta disciplina. El objetivo principal es "simular" los procesos de inteligencia humana mediante la creación y aplicación de algoritmos en un entorno informático. En otras palabras, es la habilidad de las máquinas para usar algoritmos, aprender a partir de los datos y aplicar ese conocimiento que han aprendido y adquirido para tomar decisiones. Este comportamiento es igual al humano, con la diferencia de que las máquinas no necesitan descansar y pueden trabajar con grandes volúmenes de datos a la vez.

En la actualidad, la IA se utiliza para ayudar a los humanos a obtener beneficios importantes, como mejoras en la eficiencia en casi todos los ámbitos de la vida. Sin embargo, como señala Amirhosein Toosi et al. [3], el rápido crecimiento del campo también nos obliga a estar atentos sobre las posibles desventajas, así como prevenir los riesgos que pueda generar la Inteligencia Artificial.

Una de las áreas clave de la IA es la Ingeniería Lingüística, que se encarga del estudio y tratamiento del lenguaje natural, es decir, el lenguaje humano. Esta rama incluye la recuperación de información (RI, Information

Retrieval), un modelo basado en la investigación de motores de búsqueda y sistemas capaces de devolver una lista de documentos basados en características definidas por el usuario.

Otro campo importante es el de Procesamiento del Lenguaje Natural (PLN), que estudia cómo entender el lenguaje humano. Según Nikolaos-Ioannis Galanis et al. [5], el PLN comienza por convertir el lenguaje humano en texto estructurado para que pueda ser procesado por un ordenador. Posteriormente, este texto se analiza basándose en la gramática y el contexto antes de determinar la intención del usuario mediante un proceso llamado comprensión del Lenguaje Natural. Por otro lado, la generación de Lenguaje Natural, traduce los datos generados por los ordenadores en texto comprensible para los humanos.

Entre el RI y el PLN encontramos los sistemas de respuesta automática, conocidos como Question Answering (QA), que representan una de las aplicaciones más relevantes de estas disciplinas.

2.2 Definición y evolución de los sistemas Question Answering (QA)

Dentro del campo del Procesamiento del Lenguaje Natural, el Question Answering (QA) es una de las tareas de más relevancia. Según Zhen Wang [6], el QA consiste en desarrollar sistemas que respondan automáticamente a preguntas formuladas en lenguaje natural.

El objetivo principal de estos sistemas es entender las preguntas que formulen los usuarios y que, a través de la búsqueda de información en documentos, sean capaces de extraer la información que necesitan para proporcionar respuestas correctas a las preguntas de los usuarios de una manera concreta y clara.

La evolución en la actualidad de los sistemas QA ha sido muy rápida. Al principio solo se basaban en diferentes reglas y en buscar patrones, ahora se implementan modelos de aprendizaje profundo muy complejos y sofisticados.

Los primeros sistemas QA fueron muy sencillos, basándose en unas simples reglas lingüísticas y procesamiento de texto. Uno de los primeros modelos fue Eliza, desarrollado en 1966 por Joseph Weizenbaum. Según Jeff Shrager [4], se trataba de una simple simulación de una conversación con un terapeuta. Weizenbaum no pretendía inventar el chatbot, sino construir una plataforma para investigar la conversación entre humanos y máquinas y los

importantes procesos cognitivos de la interpretación y la mala interpretación.

Otro programa fue "PARRY" de Colby, en 1975 que imitaba a un esquizofrénico paranoico. Como bien lo explica Nikolaos-Ioannis Galanis et al. [5], con los años, los programas se fueron haciendo más «inteligentes», como Eugene Goostman o Cleverbot, que analizaban enormes bases de datos de conversaciones reales para determinar las mejores respuestas. Estas primeras aproximaciones, aunque rudimentarias, fueron clave para entender la interacción humano-computadora.

Con el auge del aprendizaje profundo, los conjuntos de datos de PLN evolucionaron hacia escalas más grandes, mayor complejidad, más diversidad y mayores retos. En 1980 llega la primera arquitectura de redes neuronales convolucionales (CNN) que fue propuesta por Fukushima [3]. Poco tiempo después llegaron las redes neuronales recurrentes (RNN) y las redes a memoria a largo y corto plazo (LSTM), las cuales mejoraron significativamente los resultados en los sistemas de QA. El gran problema de estas redes era la capacidad de procesar mucho volumen de datos.

Pero esto se soluciona con la llegada de los Transformers. Esta nueva arquitectura, propuesta por Vaswani Ashish et al. [12], se basa únicamente en mecanismos de atención, que prescinde por completo de la recurrencia y las convoluciones. El modelo transformer consiguió solucionar muchos problemas de los RNN y LSTM, además de permitir trabajar en paralelo y encontrar mejores coincidencias entre palabras dentro de un texto.

El modelo más influyente fue BERT (Bidirectional Encoder Representations from Transformers), propuesto por Google en 2018. Según Jacob Devlin et al. [13], fue una revolución de los sistemas QA y destacó por ser simple y empíricamente poderoso. Ha sido entrenado para ser capaz de captar el contexto bidireccional de las palabras de una oración. Esto, permitía al modelo entender el contexto de una palabra antes y después de la palabra objetivo en una frase. Esto ayudó a entender mejor el significado de las preguntas y encontrar una mejor respuesta, al menos más concreta. La llegada de este modelo, fue una gran mejora en las tareas de PLN y por lo tanto también en las tareas de QA.

Después surgieron modelos más avanzados como GPT y T5 (text-to-text transfer transformers). GPT fue desarrollado por OpenAI. Como explican Gokul Yenduri et al. [14], es un modelo unidireccional y se usa para generar textos. Su principal innovación es la capacidad de generar respuestas coherentes en tareas de diálogo y QA conversacional.

T5, en cambio, fue propuesto por Google Research. Según Colin Raffel et al. [15], este modelo es capaz de convertir cualquier tipo de tarea de PLN, incluyendo en los sistemas de QA, como un problema de traducción de texto a texto. En el contexto de QA, el modelo transforma una pregunta en una secuencia que el modelo traduce en una respuesta. Esto se ha demostrado ser muy efectivo en tareas de recuperación de información y razonamiento lógico.

El desarrollo de los sistemas QA ha dependido en gran medida de conjuntos de datos de alta calidad. Zhen Wang [6], destaca que datasets como SQuAD, Natural Questions ó HotpotQA, han sido fundamentales para entrenar y evaluar modelos QA. A continuación vamos a hablar sobre los más utilizados:

- **SQuAD** (Stanford-Question-Answering-Dataset). En los estudios realizados por Zhen Wang [6] y Pranav Rajpurkar et al. [7], podemos observar que este modelo es uno de los conjuntos de datos más importantes y populares para entrenar modelos de QA. Este tiene preguntas formuladas en lenguaje natural sobre fragmentos de texto, con respuestas extraídas directamente del texto. Dada una pregunta y un párrafo como contexto, los modelos tienen que extraer la respuesta posible del mismo. Cada respuesta es un tramo de texto. Para dificultar la tarea, en SQuAD 2.0 se incluyen en el conjunto de datos algunas preguntas sin respuesta.

- **Natural Questions** (NQ). Estudiado por Tom Kwiatkowski et al. [8], este conjunto de datos se basa en preguntas reales hechas por usuarios de Google. Es un conjunto de datos de comprensión lectora recopilados a través del motor de búsqueda. Y la principal diferencia es que este dataset no solo proporciona el párrafo relevante como respuesta larga, sino también una respuesta corta anotada por humanos. Este usa la página de Wikipedia como contexto. Además, según el análisis de Zhen Wang [6], Natural Questions se ha consolidado como un estándar para medir la adaptabilidad de los modelos en diferentes tipos de respuestas.

- **HotpotQA.** Es el primer conjunto de datos de QA con múltiples saltos y las preguntas sólo pueden responderse haciendo inferencia en múltiples documentos. Según el estudio realizado por Zhilin Yang et al. [9], este dataset incluye anotaciones específicas para verificar si los modelos utilizan correctamente las frases relevantes, fomentando la transparencia en las inferencias realizadas. Para que el procedimiento

de razonamiento sea más fiable, también se proporcionan datos útiles a nivel de frase para ayudar a los investigadores a comprobar si los modelos generan respuestas utilizando las fuentes correctas.

- **MS MARCO**. En este conjunto de datos revisado por Payal Bajaj et al. [10], las preguntas son las consultas buscadas por los usuarios de Bing y las respuestas son generadas por anotadores humanos. Hay tres tipos de tareas en este dataset: la primera consiste en determinar si se puede responder a una pregunta, la segunda en generar una respuesta adecuada a una pregunta dada y la última en clasificar los pasajes recuperados según su relevancia.

- **CoQA**. Es el primer conjunto de datos de respuesta a preguntas conversacionales. Según el estudio de Siva Reddy et al. [11], a diferencia de otros conjuntos de datos, en CoQA cada pregunta está relacionada con otras anteriores, por lo que la respuesta no sólo debe tener en cuenta el contexto, sino también el par de preguntas respondidas anteriormente. Zhen Wang [6], también enfatiza cómo este dataset ha fomentado el desarrollo de modelos que pueden gestionar interacciones en diálogos largos.

Los sistemas de QA se dividen en tres bloques principales, pero unidos de manera coordinada, donde cada etapa cumple una función para garantizar la precisión y claridad de las respuestas proporcionadas.

- El primer bloque es el análisis de la pregunta. Esto se lleva a cabo mediante herramientas de PLN. Transforma la pregunta para que el siguiente bloque lo entienda. Es decir, analiza la estructura de la pregunta, identifica las palabras y frases clave y determina qué tipo de información se solicita.

- El segundo bloque es la búsqueda de la información solicitada. Esto se suele hacer mediante un RI de recuperación de texto que examina la consulta que le pasa el bloque anterior y devuelve una colección de documentos o fragmentos de ellos, relacionado con la búsqueda inicial introducida.

- Por último, el tercer bloque divide la información obtenida en el bloque anterior, filtra las posibles respuestas y evalúa mediante varios métodos la probabilidad de que sea alguna la respuesta. Siempre devuelve la respuesta mejor valorada.

2.3 Definiciones

A continuación vamos a definir los conceptos claves que sustentan este proyecto.El objetivo es ofrecer una comprensión de estas tecnologías, facilitando así que cualquier lector, independientemente de su nivel de experiencia y conocimientos, pueda entender el desarrollo y funcionamiento de nuestro sistema.

2.3.1 LLM

Los modelos de lenguaje de gran tamaño (Large Language Models, LLM) son avances significativos en el campo de la inteligencia artificial, diseñados para comprender, generar y manipular texto en lenguaje natural. Con el lanzamiento de Chat GPT en 2022 en tan solo unos meses, los modelos de lenguaje de gran tamaño han atraído cada vez más la atención de los investigadores y se han convertido en un campo de investigación de gran interés.

Estos modelos están construidos a partir de redes neuronales profundas y se entrenan con una cantidad muy grande de textos. Según el estudio de Yang Liu et al. [16], los LLM han mostrado un potencial considerable al procesar grandes volúmenes de datos textuales, permitiendo la identificación de patrones complejos en el lenguaje, así como la detección de contextos y relaciones entre palabras y conceptos. Gracias a su gran escala y su capacidad de abordar muchos datos, han ayudado a la automatización de tareas y creación de texto.

Las tareas más destacadas de los LLMs son:

- Análisis de sentimientos: identificar emociones y opiniones en el texto.
- Generación de texto: Desarrollan contenido creativo y concreto en respuestas a un tema o preguntas.
- Traducción y resumen: Ayudan a la traducción automática de textos, haciendo que sea una tarea más fácil y rápida, además de resumir textos muy extensos.

- Respuesta a preguntas: Son capaces de proporcionar respuestas precisas y adaptadas al contexto de las preguntas formuladas.

Aunque estos modelos de Inteligencia artificial han transformado áreas como la atención al cliente, la salud y las finanzas, también presentan desafíos importantes. Entre ellos, el problema de estos modelos es el recurso computacional que requieren, además de las preocupaciones éticas que debemos abordar, como por ejemplo el sesgo de inventarse las respuestas de los modelos cuando no encuentran datos claros o no saben la respuesta a nuestra pregunta, a lo que conocemos como “alucinaciones".

2.3.2 Retrieval Augmented Generation (RAG)

Los sistemas de Retrieval-Augmented Generation (RAG), son muy populares en el campo del procesamiento del lenguaje natural. Estos sistemas combinan diferentes técnicas para recuperar información y a continuación usan modelos generativos para que la respuesta generada sea más concisa y correcta. Este modelo se diferencia de los sistemas generativos tradicionales por un mayor enfoque en el uso del conocimiento. Estos sistemas enriquecen las respuestas buscando información en bases de datos de conocimientos (a menudo de dominios específicos) en tiempo real y la recupera para poder dar una mejor respuesta. Este tipo de sistemas son muy relevantes para áreas donde la información avanza rápidamente y está en proceso de cambio. Según Yunfan Gao et al. [17], una de las principales ventajas de RAG es su capacidad de acceder a bases de datos externas en tiempo real, lo que resulta particularmente valioso en dominios donde la información evoluciona constantemente, como la investigación científica. Su mayor ventaja es la reducción de alucinaciones, y una mayor adecuación a las peticiones del usuario.

RAG se diferencia por combinar dos técnicas: recuperación y generación de información. A continuación, se definen estas técnicas:

- Recuperación: mediante unos algoritmos de búsqueda y recuperación de información, el RAG consultará diferentes bases de datos y documentos para identificar y encontrar la mejor información a la pregunta hecha por el usuario. Esto se realiza mediante la representación semántica en un espacio vectorial común de los documentos y la pregunta o petición del usuario.

- Generación: una vez recuperada la información, este sistema utiliza un modelo generativo, normalmente de tipo Transformer decodificador, para sintetizar los datos y generar la mejor respuesta al usuario.

Los RAG y LLM son una perfecta combinación, ya que los LLM ayudan a comprender y generar textos en lenguaje natural; RAG al añadir la técnica de recuperación de información externa permite devolver unas respuestas más concretas y adaptadas a la información actual. Esta combinación se utiliza con frecuencia para tareas de QA, ya que requieren los conocimientos más recientes y tener una respuesta concreta y rápida.

2.3.3 Streamlit

El despliegue de los modelos de IA en aplicaciones web está siendo más demandado que nunca por la creciente popularidad de los frameworks de desarrollo web ligeros. Según la documentación oficial de Streamlit [18], este framework, basado en Python, se ha posicionado como una de las herramientas más utilizadas para crear aplicaciones web de forma rápida, sin la necesidad de tener grandes conocimientos de frontend.

De acuerdo con la documentación, las aplicaciones Streamlit tienen una estructura cliente-servidor. El backend de Python de una aplicación es el servidor. El frontend, que es lo que vemos a través de un navegador es el cliente. Cuando desarrollas una aplicación localmente, tu computadora ejecuta tanto el servidor como el cliente. Es muy importante comprender esta estructura cliente-servidor para compartir o implementar una aplicación para evitar errores comunes.

Streamlit permite a los desarrolladores crear aplicaciones e interactuar en tiempo real con ellas, además de tener fácil integración con modelos de Procesamiento del Lenguaje Natural y Machine Learning. Este framework es particularmente útil para tareas de QA, ya que permite desplegar modelos como BERT o GPT en una interfaz web donde los usuarios pueden formular preguntas y obtener respuestas de manera interactiva.

En resumen, todos los avances de la Inteligencia Artificial y los desarrollos de frameworks han hecho que los sistemas de QA estén al alcance de los desarrolladores para poder mejorar la interacción con los usuarios.

3. Desarrollo

En este apartado vamos a describir la parte técnica del proyecto. Comentaremos paso a paso el diseño de la interfaz web y también describiremos las herramientas que hemos usado, el funcionamiento de esta interfaz, las pruebas que hemos ido haciendo para asegurarnos que el sistema funciona bien y, por último, las limitaciones y los problemas que fueron surgiendo.

El primer paso fue crear un mockup de la interfaz web. Durante el desarrollo de este mockup hicimos varias versiones con cambios, ajustando cada versión según el feedback recibido. Una vez que se llegó a un acuerdo sobre cómo sería el diseño final que se quería, empezamos con el proceso del desarrollo del sistema.

3.1 Requisitos funcionales

Lo primero que vamos a definir en este apartado son los requisitos funcionales del sistema desarrollado. Estos requisitos son las principales funcionalidades que debe cumplir la plataforma para asegurarnos su correcta ejecución y que además, se cumple con las necesidades del usuario.

1. **Cargar el documento PDF**. Nuestro sistema tiene que permitir subir cualquier archivo PDF para analizarlo en la página Home.

2. **Cargar el archivo JSON**. El archivo JSON se tiene que subir de forma correcta en la página JSON.

3. **Visualización de los resultados**. En ambas páginas, el sistema tiene que permitir al usuario ver los resultados de una forma clara y simple.

4. **Información clave.** Las respuestas extraídas del PDF tienen que contener los datos clave que buscamos como el tipo de catalizador o el tipo de lámpara que se usa.

5. **Votación**. El sistema de votación tiene que funcionar correctamente mediante unos botones intuitivos.

6. **Descargar los resultados**. El usuario tiene que poder descargar el archivo JSON con los resultados del backend y también el archivo JSON con el nombre del usuario y sus votaciones.

3.2 Diseño de la interfaz web

Antes de describir el diseño de la interfaz, vamos a explicar de una forma resumida y simple la arquitectura general de nuestro sistema (ver Figura 1).

El primer paso es realizado por el usuario cuando sube un archivo PDF a través de la interfaz. Este archivo se envía al backend mediante una solicitud HTTP POST. En el backend, el documento es procesado usando Grobid para extraer su contenido estructurado. Después, los datos extraídos son analizados con modelos de lenguaje LLM y técnicas RAG, generando las respuestas, que serán devueltas en un archivo en formato JSON al frontend. Por último, el usuario puede visualizar los resultados de manera organizada y descargarlos.

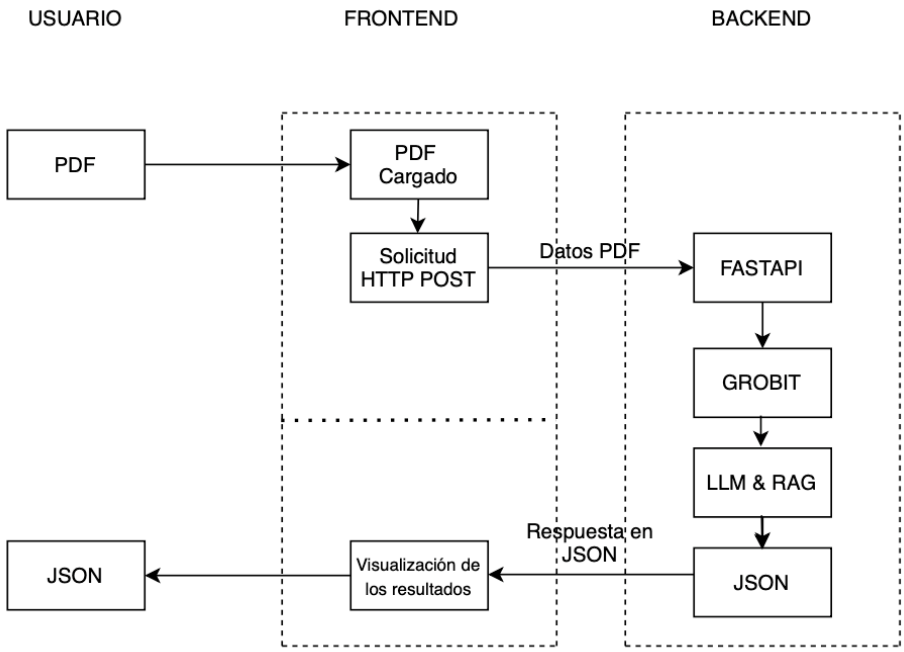


Figura 1. Arquitectura del sistema

Ahora que ya tenemos una idea clara de la arquitectura podemos profundizar en el diseño y funcionamiento de nuestra interfaz web. Esta destaca por ser simple y organizada (ver Figura 2). Está diseñada de una forma muy clara para que los usuarios puedan cargar sus artículos científicos, visualizar las respuestas de forma atractiva, evaluar las respuestas a las preguntas predefinidas mediante un sistema de votación y descargar los resultados. Más adelante hablaremos de cada parte.

Esta interfaz web es el punto de acceso de los investigadores químicos al sistema. Debido a esto, nuestro diseño se centró en conseguir que sea intuitiva, que tenga una fácil accesibilidad y lo más importante, una presentación clara.

Figura 2. Interfaz web desarrollada con Streamlit.

Por último, para una mayor comprensión de los siguientes apartados, vamos a explicar en resumen lo que es el frontend y el backend en este sistema.

- **Backend**: Es la API desarrollada por el OEG. Es el encargado de procesar los documentos PDF y generar los resultados a las preguntas

predefinidas en formato JSON a partir de modelos de lenguaje y embeddings.

- **Frontend**: Es la interfaz web del usuario construida con Streamlit. Es la parte que se encarga de permitir cargar los archivos científicos para ser enviados al backend, y también en donde se puede visualizar y descargar los resultados recibidos.

3.2.1 Herramientas utilizadas

Para la creación de esta interfaz web se han usado herramientas muy reconocidas en el ámbito del desarrollo de software y la Inteligencia Artificial. A continuación, vamos a detallar en resumen para que hemos usado cada una de ellas.

- **Figma.** Figma fue la herramienta que se usó para diseñar el mockup de la interfaz web para tener una guía para el desarrollo en Streamlit.

- **Streamlit.** Streamlit fue el framework elegido para desarrollar la interfaz web del proyecto por su fácil uso y su capacidad de crear aplicaciones de forma rápida.

- **Python.** Python ha sido el lenguaje de programación elegido para nuestro desarrollo, debido a que Streamlit está diseñado para trabajar exclusivamente con él. Además, es el lenguaje de programación más utilizado en el campo de la Inteligencia artificial.

- **HTML y CSS:** Aunque Streamlit no necesita HTML o CSS, los hemos usado en algunas partes del código con el objetivo de mejorar y personalizar algunos aspectos de diseño. Por ejemplo, añadimos espacios, encabezados y estilos personalizados usando st.markdown con la opción unsafe_allow_html=True. Con esto se consigue una interfaz visualmente más atractiva.

- **HTTP Requests:** La interfaz web desarrollada en Streamlit se comunica con el backend mediante peticiones HTTP.

- **GitHub:** GitHub se ha usado como herramienta para ir controlando y revisando las distintas versiones y cambios que se han hecho. Además,

esta herramienta es muy importante, porque permite trabajar en equipo durante el desarrollo de un proyecto.

3.2.2 Diseño del frontend

El desarrollo del frontend en Streamlit se centra en todo momento en el usuario final, que son los investigadores científicos. El frontend consta de tres páginas (ver Figura 3):

- **Home:** Página principal en la que los usuarios pueden cargar documentos PDF y enviar solicitudes al backend para su análisis. Además permite descargar en formato JSON los resultados devueltos por el backend, para evaluarlos y validarlos si se desea en la página JSON.

- **JSON:** Página donde se pueden visualizar los resultados obtenidos de una forma más clara que en un archivo JSON, evaluar los resultados devueltos mediante votaciones y descargar el JSON con los votos realizados.

- **About**: Esta página es el espacio donde viene la información de nuestro proyecto, así como el objetivo de este proyecto y las personas que hay detrás de esto.

En resumen, cada una de estas páginas desempeña una función diferente en el proceso del trabajo. A continuación, vamos a detallar de una forma más extensa las características y las funcionalidades de cada una de estas páginas.

Figura 3. Las distintas páginas de la interfaz.

3.2.2.1 Página Home

Esta página, es la pantalla de inicio de nuestra interfaz en la que el usuario puede cargar el artículo científico en formato PDF y enviarlo al backend para su análisis (ver Figura 4). Una vez recibido el resultado, se podrá visualizar y descargar los resultados. Los pasos que el usuario tiene que seguir para completar esta interacción son los siguientes:

Figura 4. Página Home de la interfaz web.

1. **Cargar el documento.** Es una sección donde el usuario puede cargar el documento PDF para su análisis (ver Figura 5). Solo se permite cargar documentos en formato PDF.

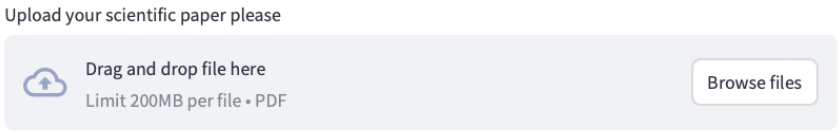


Figura 5. Espacio para cargar el artículo científico en formato PDF

2. **Añadir el DOI opcional**. Digital Object Identifier (DOI) es un identificador único y permanente asignado a documentos digitales,

como artículos científicos, que permite su localización y acceso de manera precisa y confiable. Según el estudio de Norman Paskin [19], el sistema DOI fue diseñado para proporcionar una infraestructura robusta que facilite la gestión, el intercambio y la preservación de contenidos digitales. En caso de que el usuario tenga el DOI del artículo científico que vaya a analizar, puede introducirlo manualmente (ver Figura 6). En caso de no tenerlo, se devolverá en el JSON recibido del backend, siempre y cuando en el documento subido se menciona en alguna parte este identificador, si no se devolverá como “None Given”.

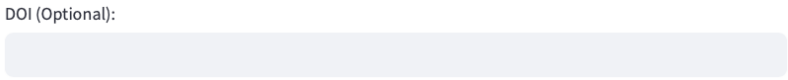


Figura 6. Espacio para añadir de manera opcional el DOI

3. **Enviar archivo al backend para su análisis**. Cuando el usuario presione el botón “Submit”, el archivo se enviará al backend para su análisis. Esto durará unos minutos. Mientras el backend trabaja, en el frontend aparece el siguiente mensaje: “Analyzing your paper. Please be patient...” (ver Figura 7).

Figura 7. Visualización de la interfaz web mientras el backend genera los resultados

4. **Visualizar las respuestas.** Una vez recibidos los resultados del backend, estos se verán directamente de manera organizada cómo se observa en la *Figura 8*. A continuación, vamos a detallar cada punto de lo que se puede ver.

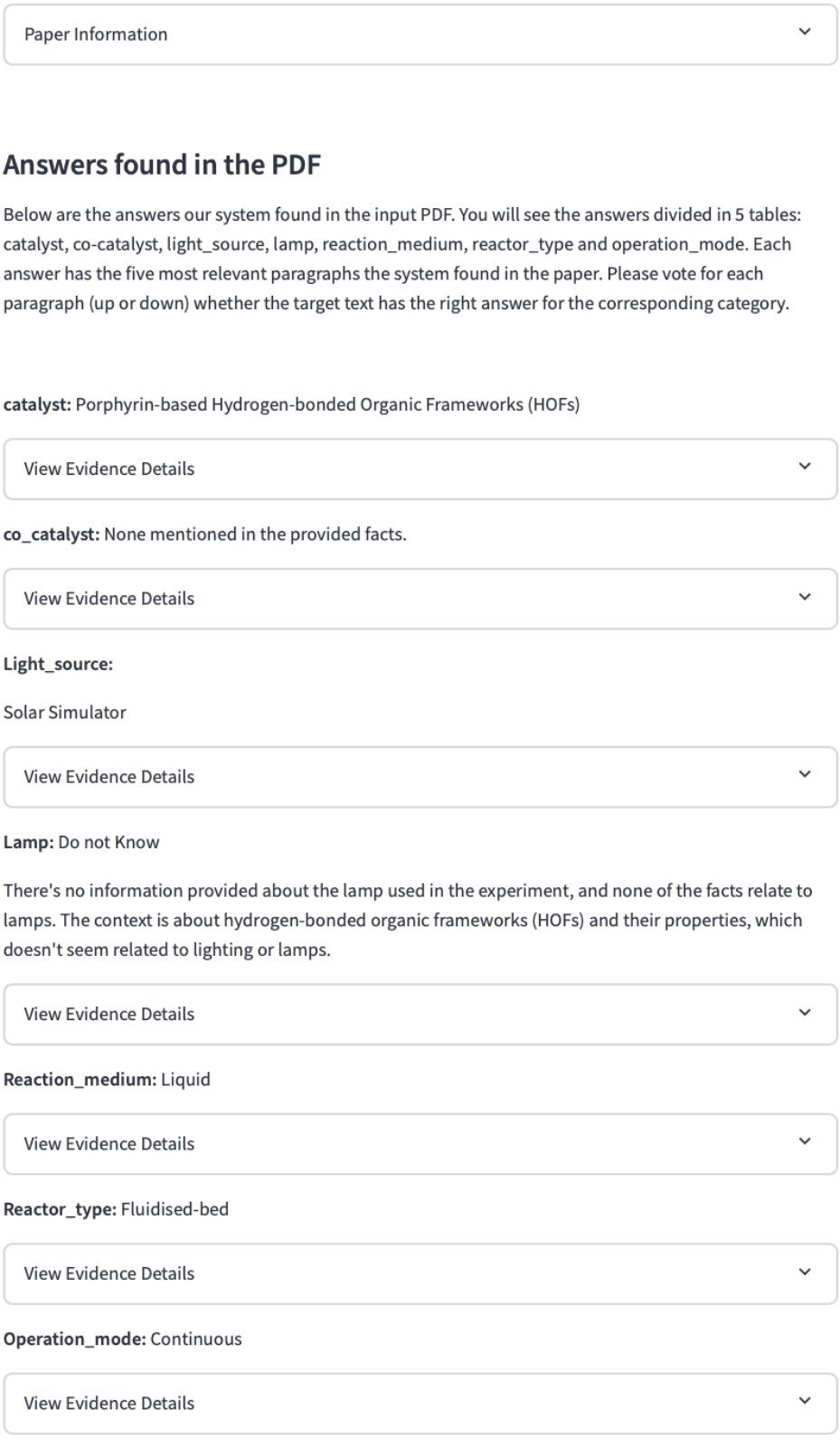


Figura 8. Visualización de los resultados

Una vez recibidos los resultados, lo primero que se puede ver es una sección en la que aparece "*Paper Information*" (ver *Figura 8*). Esta sección está en una caja desplegable. Al presionar la flecha hacia abajo, se despliega la información que incluye el título del artículo científico y el “DOI” (ver *Figura 9*).

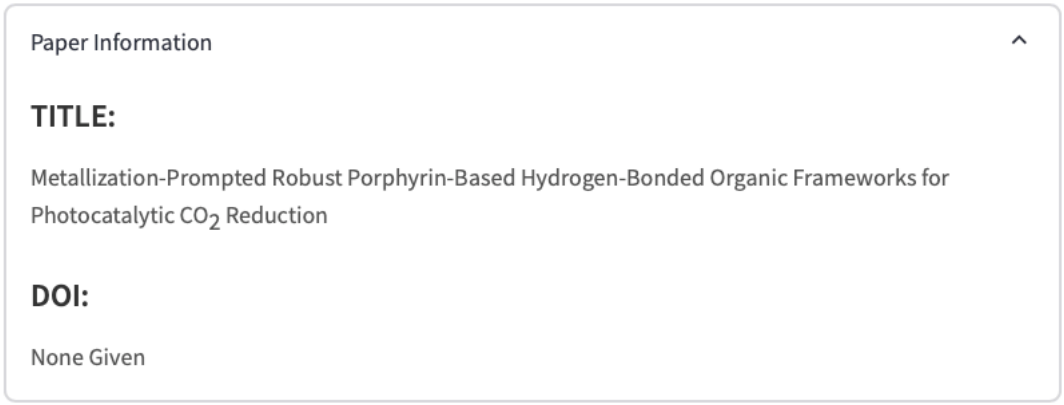


Figura 9. Información del artículo científico

Debajo de la información del documento, hay una sección "Answers found in the PDF". Aquí es donde el sistema muestra las respuestas a las preguntas predefinidas que el modelo ha identificado durante el análisis. Está organizada en tablas con las respuestas agrupadas en categorías como Catalyst (Catalizador) o Light Source (Fuente de luz). Cada categoría tiene una caja desplegable. Al hacer clic en "View Evidence Details", se muestran los cinco párrafos más relevantes del artículo donde se encontró esa respuesta (ver Figura 10).

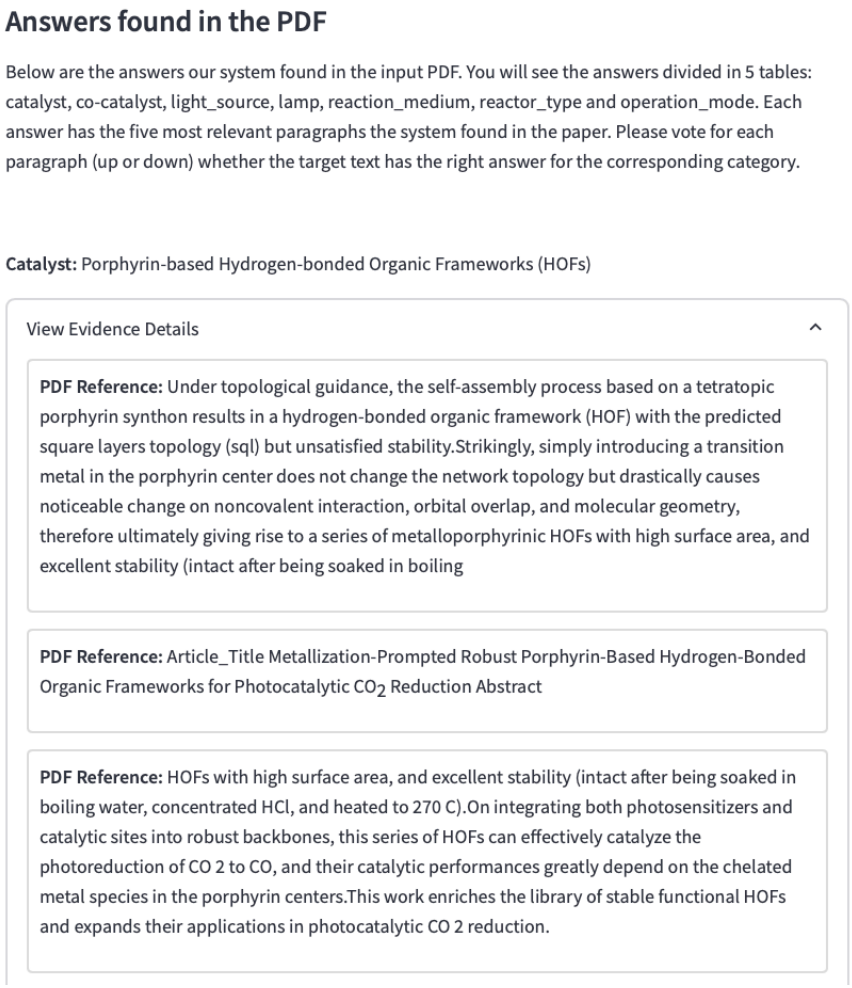


Figura 10. Párrafos más relevantes

5. **Descarga de los resultados en formato JSON.** Por último, el usuario puede descargar los resultados en un archivo JSON presionando el botón “Download JSON” (ver Figura 11).

Figura 11. Descarga del archivo JSON

3.2.2.2 Página JSON

La página JSON (ver Figura 12), está pensada para evaluar el archivo JSON con los resultados, devuelto por el backend en la página "Home". Aquí el usuario podrá visualizar el archivo JSON en un formato más claro y atractivo, evaluar cada respuesta recibida del backend mediante un sistema de votación y descargar el archivo JSON otra vez, pero añadiendo el nombre del anotador y de los votos que se hayan hecho. Los pasos que el usuario tiene que seguir para completar esta interacción son los siguientes:

Figura 12. Página JSON de la interfaz web.

1. **Cargar el documento.** Es una sección donde el usuario puede cargar los archivos JSON descargados en la página "Home" para su análisis (ver Figura 13).

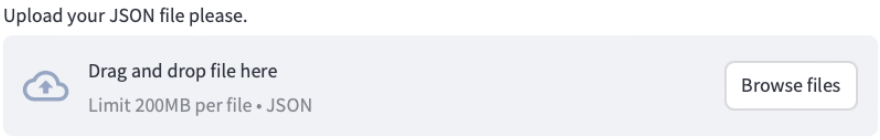


Figura 13. Espacio para cargar el artículo científico en formato PDF

2. **Ingresar el nombre del usuario.** Una vez subido el archivo, el usuario tendrá que introducir su nombre (ver Figura 14), para que en el archivo JSON aparezca la persona que haya votado y los votos que ha realizado, para una mayor exactitud.

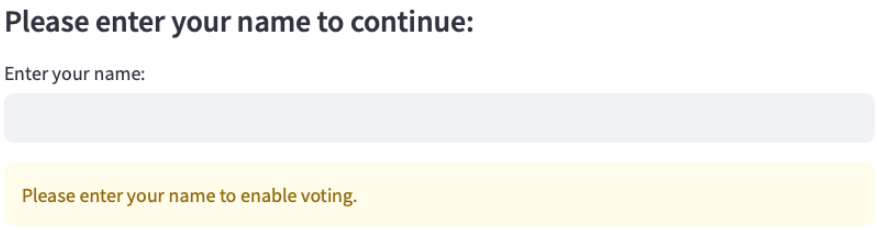


Figura 14. Añadir el nombre del usuario

3. **Visualización de los resultados y sistema de votación.** Una vez introducido el nombre del usuario, se accede a la pantalla donde se podrá visualizar todo el contenido del archivo JSON (ver Figura 15). A continuación, vamos a detallar cada punto de lo que se puede ver.

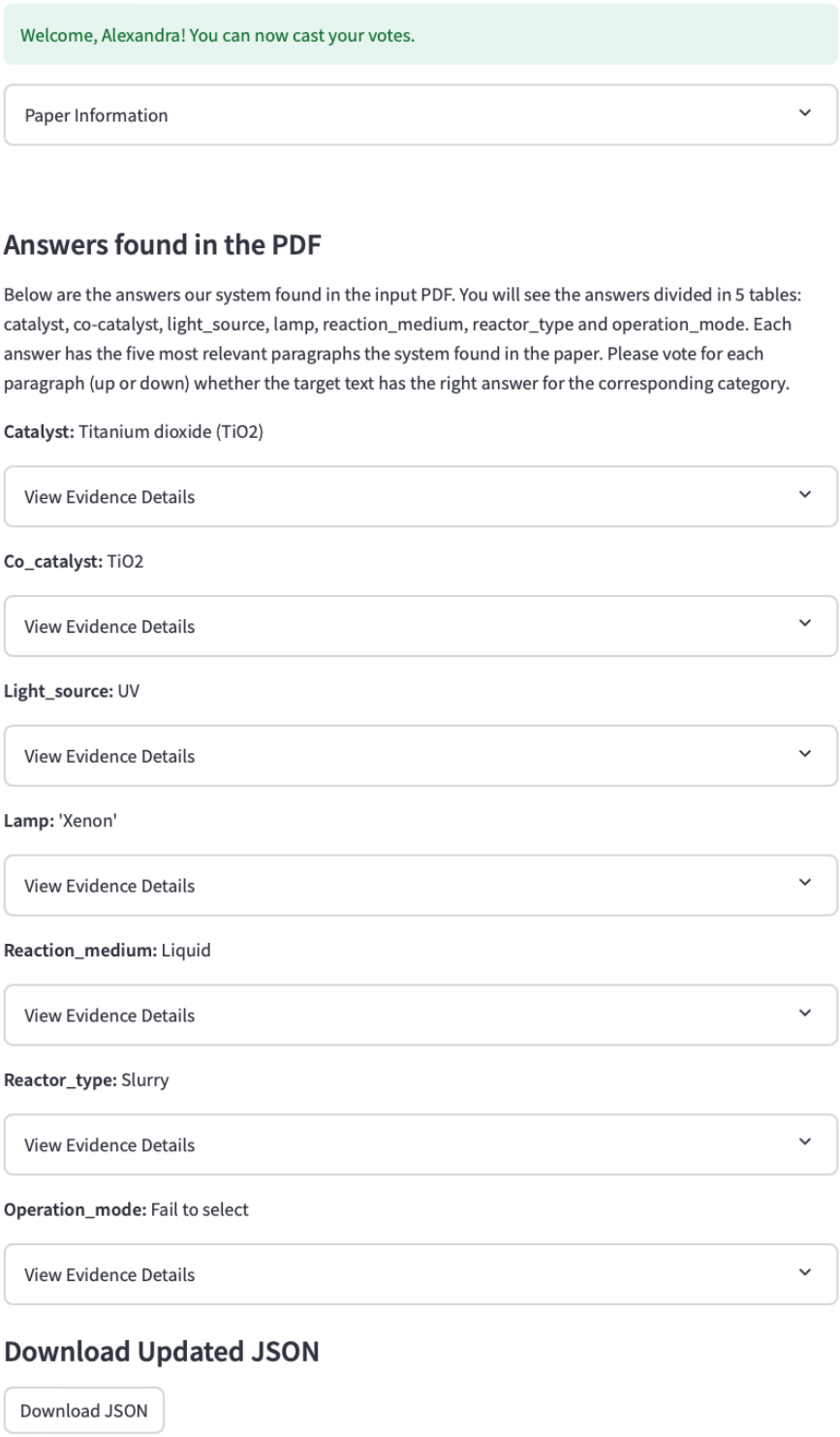


Figura 15. Visualización del archivo JSON

Lo primero que se muestra es "Paper Information" (ver Figura 15). Esta sección está inicialmente en una caja desplegable. Al presionar la flecha hacia abajo, se despliega la información que incluye el título del artículo científico y el “DOI” (ver Figura 16).

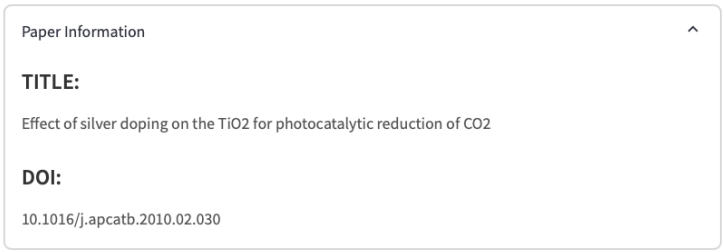


Figura 16. Información del artículo científico

Debajo de la información del documento, hay una sección "Answers found in the PDF". Aquí es donde el sistema muestra las respuestas a las preguntas predefinidas que el modelo ha identificado durante el análisis. Está organizada en tablas con las respuestas agrupadas en categorías como Catalyst (Catalizador) o Light Source (Fuente de luz). Cada categoría tiene una caja desplegable. Al hacer clic en "View Evidence Details", se muestran los cinco párrafos más relevantes de dentro del artículo donde se encontró esa respuesta (ver Figura 17).

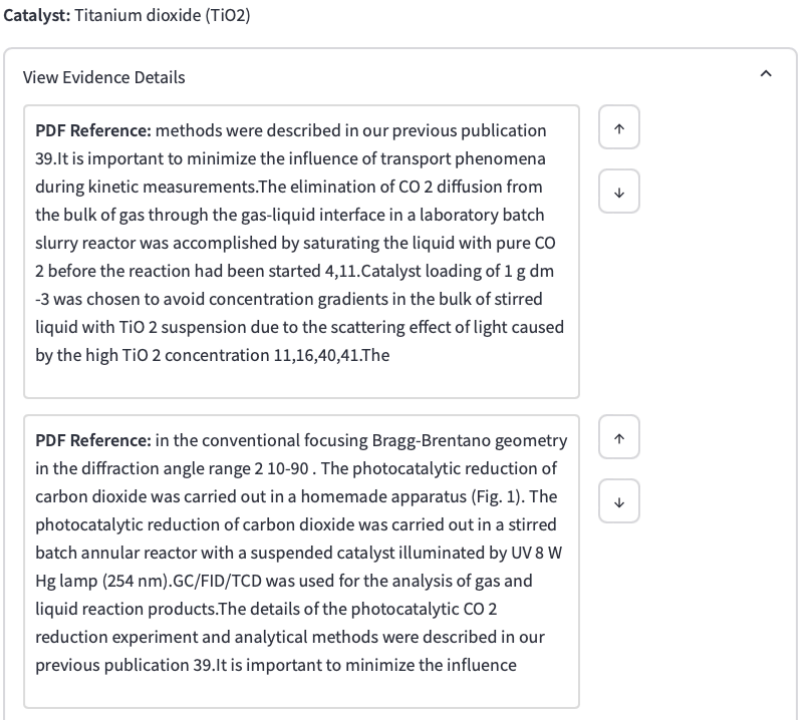


Figura 17. Párrafos más relevantes.

Al lado de cada párrafo, hay dos botones, que son los botones que el usuario tiene que usar para votar (ver Figura 18). Si presiona el botón de flecha hacia arriba (↑) está indicando que el párrafo es correcto o relevante. Al votar positivo, el botón se vuelve verde, indicando que la respuesta ha sido evaluada de forma positiva. Pero sí presiona el botón de flecha hacia abajo (↓) está indicando que el párrafo no es correcto o no es relevante. Al votar negativo, el botón se vuelve rojo, indicando que el párrafo ha sido evaluado de forma negativa, es decir que no cumple con las expectativas.

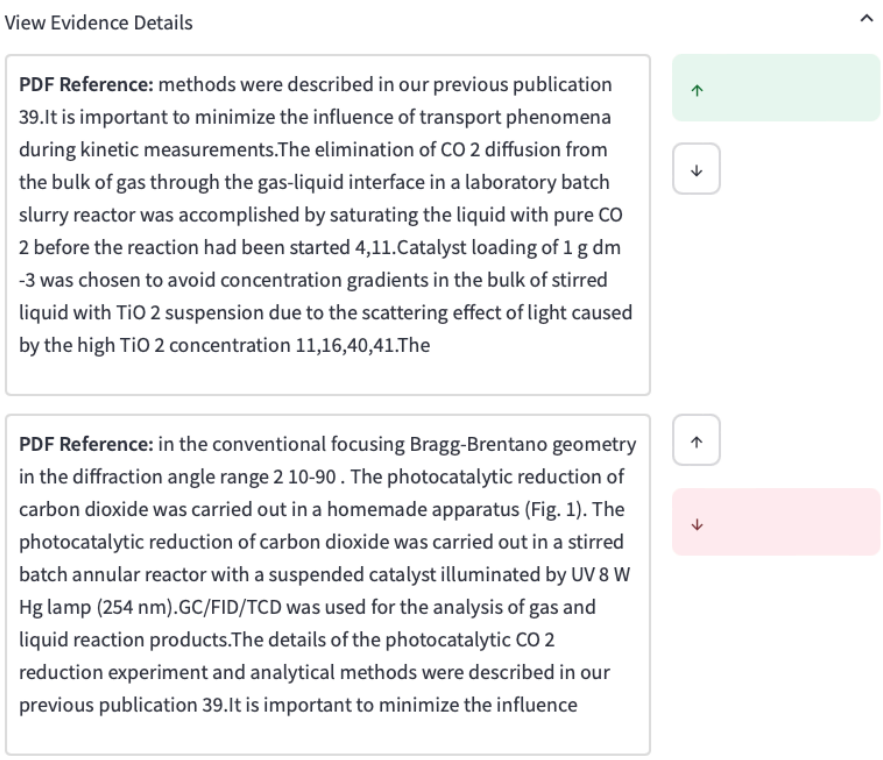


Figura 18. Visualización del archivo JSON

4. **Descarga del archivo JSON**. Por último, una vez realizados los votos, podremos descargar el JSON con nuestro nombre, los votos que hayamos hecho y todos los datos devueltos por el backend (ver Figura 19). Para eso solo tenemos que presionar el botón "Download JSON".

Figura 19. Botón para descargar el archivo JSON

Antes de pasar a la página “About”, se va a explicar de forma más detallada porque es tan importante nuestro sistema de votación.

Este mecanismo permite a los investigadores y científicos evaluar directamente la calidad de las respuestas generadas por el sistema. Al votar sí una respuesta es válida o no, nos proporcionan retroalimentación que nos ayuda a ajustar y mejorar los modelos de Inteligencia Artificial utilizados. El impacto del sistema de votación se puede ver en varios aspectos clave:

- **Validación**. Los investigadores pueden revisar las respuestas que se han generado y votar inmediatamente para evaluarlas. Esto nos permite detectar, de forma rápida, errores o inconsistencias para asegurarnos que la información que ha sido devuelta por el modelo sea correcta y relevante para el análisis.

- **Mejora continua del modelo**. Mediante las votaciones podemos obtener un conjunto de datos. Tener datos etiquetados por expertos, facilita el entrenamiento y ajuste de los modelos de lenguaje, además de la extracción de información.

- **Aumento de la confiabilidad.** Al estar involucrados de forma directa los científicos en el proceso de validación de las respuestas, conseguimos generar una mayor confianza en los resultados.

- **Identificación de patrones de error.** El sistema de votación nos permite identificar áreas donde el modelo suele cometer errores de forma continua. Esto sirve para poder realizar mejoras en los puntos críticos del sistema.

3.2.2.3 Página About

La página "About" proporciona una descripción general de la interfaz web, explicando su propósito, las tecnologías que se usan y el equipo que hay detrás del desarrollo de esta interfaz web (ver Figura 20).

Tiene un diseño simple y claro, lo que permite a las personas que visiten esta plataforma comprender rápido el objetivo del proyecto y sus beneficios.

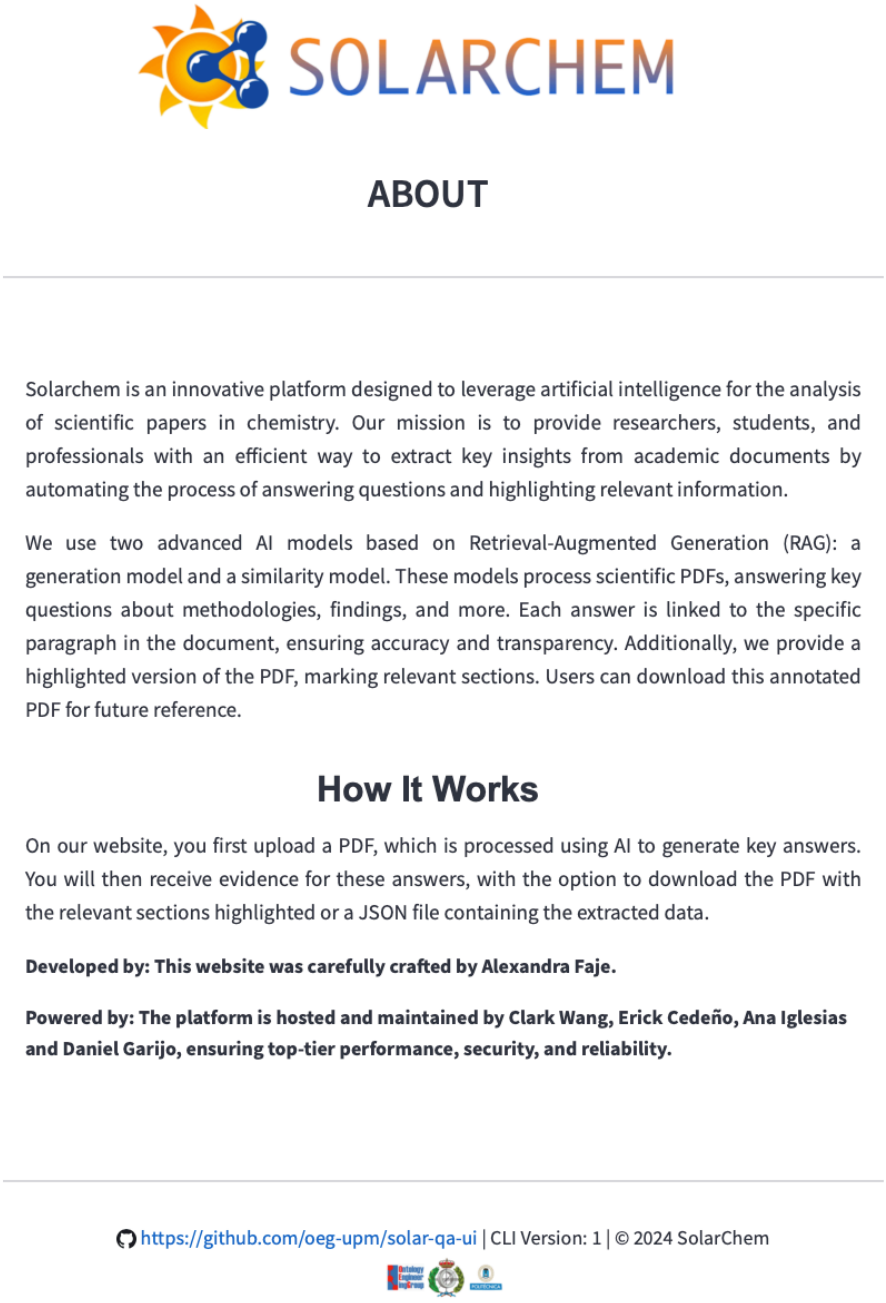


Figura 20. Página About de la interfaz web

En resumen, la página “About" no solo sirve para dar una presentación general de la interfaz web y del equipo que hay detrás, sino que con esto conseguimos aumentar la confianza de los usuarios mostrando profesionalidad y transparencia al explicar de una forma simple y clara el funcionamiento y reconocimiento del trabajo de todo el equipo.

3.3 Descripción del backend y su estructura

El desarrollo de este sistema conlleva la conexión de nuestra interfaz web con el backend desarrollado por el Ontology Engineering Group (OEG). El backend constituye el núcleo funcional del sistema, que va desde que el usuario carga el documento PDF hasta la generación del JSON con las respuestas. A continuación, vamos a describir de forma general la estructura y el funcionamiento del backend, para que se tenga una mayor comprensión del sistema.

- **Carga de documentos**. Una vez que el usuario haya cargado su documento en formato PDF, el backend lo recibe y este es procesado por Grobid para extraer textos. Grobid es la herramienta que se usa para extraer textos de los documentos PDF, para garantizar que los modelos que se usan trabajen con datos de calidad.

- **Análisis del texto.** Los datos que se extraen se envían al modelo RAG. Este convierte el texto del PDF en vectores (embeddings) utilizando un modelo de embeddings preentrenado. Como ya explicamos en el capítulo 2 de este documento, el RAG lo que hace es combinar recuperación de información con generación de texto para poder dar respuestas concretas.

- **Generación de respuestas.** El modelo de lenguaje de gran tamaño que se usa, toma la información recuperada y genera las respuestas a las preguntas, en lenguaje natural. Las preguntas predefinidas que el sistema tiene que responder, son unas preguntas específicas sobre química solar como por ejemplo, —¿Cuál es el catalizador utilizado en el experimento?, o — ¿Cuál fue la fuente de luz utilizada?

- **Envío de los resultados.** Por último, el backend envía los resultados en formato JSON al frontend, mediante el protocolo HTTP.

3.4 Conexión entre la interfaz web y el backend

El proceso de comunicación entre el frontend y el backend es lo que permite el intercambio de datos entre ambas partes (ver Figura 21).

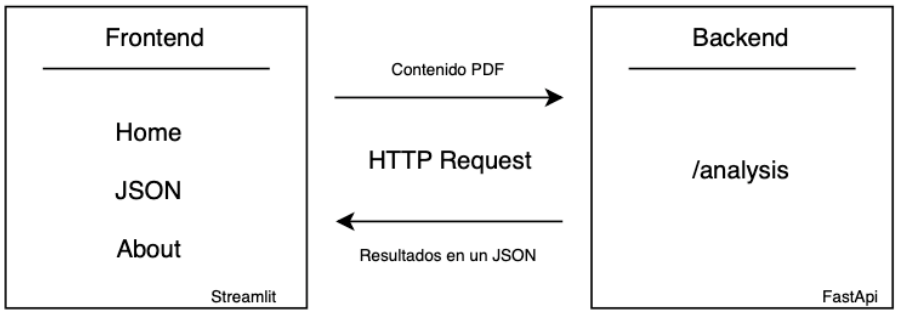


Figura 21. Diagrama de flujo de la conexión entre el backend y el frontend

La conexión entre el frontend y el backend en este proyecto se realiza a través del protocolo de comunicación estándar HTTP (Hypertext Transfer Protocol), que se usa para la transferencia de datos entre sistemas distribuidos. Según la documentación de Mozilla Developer Network [20], HTTP es un protocolo fundamental para la web que permite el intercambio de información entre un cliente (como el navegador o la aplicación frontend) y un servidor (el backend en este caso). A continuación, se detalla el funcionamiento de este proceso, la importancia que tiene y sus diferentes etapas.

En este proyecto, el frontend envía solicitudes HTTP al backend para procesar archivos PDF y después, una vez tenga los resultados, devolverlos en un JSON. Las peticiones HTTP permiten que el frontend actúe como cliente y el backend como servidor, haciendo de esta forma una fácil comunicación bidireccional.

Nuestro sistema solo utiliza el método POST para enviar datos. A diferencia del método GET que se usa para recuperar datos, POST lo que permite es enviar información y archivos adjuntos.

Características de la petición HTTP POST:

- **Cuerpo (Body):** El body es el que contiene la información que se quiere enviar. Por ejemplo, en nuestro sistema, el body son los artículos científicos en formato PDF.
- **Cabeceras (Headers)**: Las cabeceras describen el tipo de contenido enviado (en nuestro caso, application/json).
- **URL del endpoint**: Aquí es donde se define la ruta en el backend que debe recibir la solicitud, en nuestro caso /analysis/.

A continuación, vamos a describir las diferentes etapas que hay entre el frontend y el backend en este proyecto:

1. **Creación de la petición HTTP POST (Frontend):**

- Cuando el usuario carga el documento PDF y presiona el botón "Submit" en la interfaz de Streamlit, se crea una solicitud HTTP POST.
- Esto genera un diccionario de datos que contiene todos los parámetros clave como el identificador del modelo de lenguaje, el archivo de entrada y otros datos de configuración.
- Esta solicitud se envía al endpoint /analysis/ del backend usando la biblioteca requests de Python.

2. **Recepción de la petición (Backend):**

- FastAPI recibe la petición HTTP y extrae el cuerpo de la solicitud. Se hace una validación de los datos para asegurar que están todos los campos necesarios y que además cumplen con el formato que se requiere.

3. **Procesamiento de la petición:**

- El backend procesa el archivo PDF usando modelos de lenguaje y algoritmos de embeddings.

4. **Generación de la respuesta (JSON):**

- Una vez completado el análisis, el backend devuelve los resultados en un archivo en formato JSON que incluye los resultados del análisis, el título del artículo, el DOI (si está disponible) y otros datos como

"generation model" que es el modelo de lenguaje que se ha usado para el análisis o el "rag_type", el tipo de modelo RAG.

5. **Envío de la respuesta al Frontend:**

- El backend envía la respuesta en formato JSON al frontend a través de la misma conexión HTTP.

- Streamlit recibe el JSON y visualiza los resultados directamente en la interfaz del usuario, es decir en el frontend.

Para la comunicación entre el frontend y el backend, este proceso de conexión HTTP es muy importante, ya que mediante el uso de POST, conseguimos tener una comunicación entre ellos rápida y estructurada.

4. Evaluación, comparación y limitaciones del proyecto

4.1 Evaluación del sistema

Para conseguir un sistema con un funcionamiento correcto no solo depende del código, sino que también es importante realizar pruebas técnicas y considerar el feedback recibido durante el desarrollo. Nuestra evaluación se centra en el diseño, en el valor de las respuestas recibidas y en la satisfacción de los usuarios finales. Durante este proyecto, hicimos diferentes pruebas a nivel de interfaz, pruebas de funcionamiento y pruebas de conectividad entre el frontend y el backend para comprobar sí cumple con los requisitos y expectativas planteadas sin errores.

4.1.1 Pruebas de interfaz y experiencia de usuario

Durante el desarrollo de la interfaz web hicimos distintas pruebas para mejorar el diseño y la experiencia del usuario. Además, tuvimos la oportunidad de enseñar la interfaz a un equipo de investigadores especializados en química solar. Su feedback fue de gran ayuda para algunos detalles del sistema. A continuación vamos a detallar algunas de las pruebas de interfaz y diseño que hicimos.

- **Diseño**. Mediante las pruebas del diseño de la interfaz, nos aseguramos de que todos los elementos están correctamente alineados y que la experiencia resulte intuitiva para el usuario final.

- **Funcionalidad de los botones y campos**. Sé revisó que los botones funcionan correctamente y que los campos de texto, como por ejemplo el de introducir el nombre del usuario, responden correctamente a las interacciones del usuario.

- **Añadir nuevas funcionalidades**. Durante estas pruebas no solo se cambiaron algunas partes del diseño, sino que también se añadieron nuevas funcionalidades como el sistema de votación.

4.1.2 Pruebas de rendimiento y precisión del modelo

Con estas pruebas se pretendía observar el comportamiento del sistema en diferentes condiciones. Nuestro objetivo era asegurarnos que este sistema es capaz de analizar diferentes tipos de documentos PDF, devolver resultados claros y sin errores y visualizarlos en Streamlit. Para esto lo que hicimos fue cargar diferentes artículos científicos sobre química solar con diferentes estructuras y tamaños para comprobar que el análisis y los resultados generados se realizaban de forma correcta. Los resultados fueron los esperados, ya que el sistema nos devolvió los resultados en formato JSON sin ningún error.

Otra prueba fue la de validar manualmente las respuestas generadas. Esta prueba consistía en cargar varios artículos científicos sobre química solar y verificar que las respuestas generadas por el modelo coincidían con el contenido que había en los artículos científicos. Cómo ajuste a esta prueba se añadió el sistema de votación para mejorar el contenido de las respuestas. De esta manera, vamos a tener siempre una forma continua de analizar las respuestas generadas.

En cuanto a métricas de rendimiento, no hicimos ninguna prueba durante el desarrollo, pero será un trabajo futuro. Sin embargo, de manera informal, calculamos el tiempo que se tardaba en este proceso, desde que se cargaba el documento PDF hasta que el modelo devolvía una respuesta con los resultados obtenidos. El tiempo de respuesta era aproximadamente unos 2 o 3 minutos por documento PDF, aunque en alguna ocasión más tiempo dependiendo del tamaño y complejidad de cada documento.

4.1.3 Pruebas de conectividad entre frontend y backend

También se realizó diferentes pruebas para confirmar que la conexión entre el backend y el frontend fuera correcta, y que el archivo JSON se recibiera de forma correcta sin ningún tipo de error o falta de información. Como resultado, se consiguió que la comunicación sea estable y que el sistema responda de forma correcta a las solicitudes del usuario, devolviendo los resultados de forma correcta.

4.2 Complicaciones encontradas y soluciones implementadas

Durante el desarrollo de la interfaz web surgieron varias complicaciones que frenaron el progreso del proyecto. A continuación, vamos a detallar las complicaciones principales encontradas y las soluciones implementadas:

1. **Visualización de resultados.** Al principio, las respuestas de los párrafos destacados del artículo científico devueltas por el sistema se podían ver en un formato de lista y los botones de las votaciones estaban debajo de cada párrafo. Este formato no era el más adecuado a la hora de conseguir una experiencia intuitiva de los usuarios.

   - Solución: Se decidió rediseñar la visualización de las respuestas en tablas donde cada tabla corresponde a una respuesta. Después se cambiaron los botones de votación, ubicándolos a la derecha de cada párrafo. De esta forma, conseguimos un diseño más atractivo e intuitivo.

2. **Problemas de diseño con los botones**. El desafío más grande fue la creación de botones dinámicos, como por ejemplo los botones de votación. Conseguir cambiar de color al ser presionados fue un gran desafío ya que, Streamlit no permite la actualización directa de CSS dinámico en botones (st.button).

   - Solución: Empezamos a usar st.session_state y callbacks (on_click) para la actualización del estado de los botones al instante para así dar una respuesta visual inmediata. Con estos cambios conseguimos que los botones seleccionados quedaran marcados, o que cambiaran de color.

3. **Respuestas subrayadas en el artículo científico**. En un principio, queríamos ofrecer la posibilidad de descargar el artículo científico original pero con los resultados subrayados. Es decir, los párrafos detectados como más importantes y analizados por el backend se pudieran marcar directamente en el documento para facilitarles a los investigadores la revisión. El problema fue cuando nos dimos cuenta de que el modelo interpretaba de manera diferente ciertas palabras o fórmulas que estaban en el documento. Para poner un ejemplo, palabras como "CO_2" se interpretaban como "CO2" y por tanto, ya no eran iguales para el modelo y no subrayaba esa parte.

   - Solución: Finalmente, se decidió no añadir esta función de subrayado, ya que este proceso implicaba un proceso de entrenamiento

del modelo para identificar y distinguir las diferentes tipografías y contextos de cada palabra.

4. **Conexión con el Backend**. La conexión entre nuestro frontend y el backend fue otro desafío importante. La comunicación entre ambos componentes no funcionaba correctamente en las primeras pruebas. Esto afectó en el avance del proyecto, al tener problemas con el envío de los artículos científicos y con la generación del JSON y recepción de las respuestas.

   - Solución: Realizamos pruebas de forma manual utilizando curl desde la terminal. Esto nos permitía enviar peticiones HTTP directamente al backend simulando el comportamiento del frontend. Además, hicimos varias pruebas con peticiones HTTP directas usando la librería requests en Streamlit. Con estas pruebas conseguimos solucionar el problema en la conexión con el backend y verificar si la recepción de los artículos científicos y el proceso de generación de respuestas estaban ejecutándose correctamente.

5. **Formato y contenido del archivo JSON final**. En este apartado hemos tenido varios desafíos a lo largo del proyecto. El primero fue que al principio el formato del archivo JSON con los resultados y votos estaba en CSV. El segundo fue que la variable vote_name que almacena el nombre del usuario no era un nombre intuitivo. Por último, el JSON que se descargaba no contenía la misma información que el JSON devuelto por el backend, lo que causaba un problema.

   - Solución: Lo primero se modificó el código para que la descarga final de los resultados fueran en un formato JSON y no en un CSV. Vote_name se reemplazó por annotator_name y se eliminó cualquier campo vacío o innecesario. También se ajustó el JSON para tener todos los datos necesarios, consiguiendo así el mismo formato que el JSON devuelto por el backend.

6. **Visualización de los votos en el JSON**. En un inicio, cuando el usuario votaba, aparecía en el JSON una lista con su nombre junto con el voto, lo que se hacía muy repetitivo. Otro problema a solucionar, fue cuando el usuario no votaba, este campo aparecía como null, siendo esto información relevante.

   - Solución: Ajustamos el código para que en el JSON final solo apareciera una sola vez el nombre del anotador. Para el caso de que el anotador no votase, cambiamos el diseño para que ese campo no

apareciera en el archivo JSON. Por tanto en el diseño final solo aparecerá en el campo del voto un 1 y 0, siendo 1 el voto positivo y 0 el voto negativo.

7. **Uso de GitHub**. El uso de GitHub fue un reto personal y técnico, debido a la falta de uso y conocimiento sobre esta plataforma. Para poder colaborar en equipo, esta plataforma es esencial.

   - Solución: Investigando y haciendo consultas en diferentes foros y documentación sobre el uso de esta plataforma poco intuitiva y teniendo una práctica constante, se consiguió entender y manejar mejor GitHub.

4.3 Limitaciones del proyecto

Aunque la mayoría de los resultados han sido positivos, a continuación vamos a comentar las limitaciones principales.

- **Limitaciones computacionales.** A pesar de que este sistema se ejecuta de una forma eficiente en un hardware común, el integrar modelos avanzados de Inteligencia Artificial requiere recursos importantes. Esto puede limitar su uso en entornos que tengan una infraestructura más limitada, afectando la capacidad de procesamiento y sobre todo en el tiempo de respuesta del modelo.

- **Limitaciones de Streamlit.** Streamlit es una herramienta muy buena para el desarrollo rápido de aplicaciones web. Sin embargo, tiene algunas limitaciones de diseño en comparación con otros frameworks, como JavaScript y HTML, por ejemplo. Aunque Streamlit genera componentes en HTML y JavaScript, la personalización es limitada y no permite modificar el diseño con la misma libertad que otros frameworks.

Con estas limitaciones, queremos decir que para proyectos que se usen modelos de inteligencia artificial se necesita un hardware en mejores condiciones. Además, para proyectos que necesiten un diseño fijo o diseños más avanzados y complejos, es mejor usar otros frameworks como JavaScript, por ejemplo.

5. Análisis de impacto

En este apartado, vamos a comentar el impacto de este proyecto en el ámbito de la investigación de la química solar, así como sus repercusiones sociales, éticas y económicas.

5.1 Impacto en la investigación de la química solar

Este proyecto pretende impactar de una forma directa y positiva en la investigación en la química solar. Con esta herramienta se pretende conseguir acelerar la extracción de datos importantes y de esta forma permitir a los investigadores centrarse más en otras tareas como los análisis profundos y en la generación de nuevas hipótesis. A continuación, vamos a nombrar la aportación principal que ofrece este sistema:

- **Agilizar el análisis científico**. Los investigadores que dedican mucho tiempo a las tareas manuales, con este sistema les estamos ofreciendo más tiempo de dedicación para otras tareas, además de facilitar la comparación entre estudios. Es decir, los investigadores pueden trabajar con grandes volúmenes de datos en menos tiempo. Con esto se conseguiría llegar de forma más rápida a datos importantes.

5.2. Repercusiones sociales y económicas

5.2.1 Repercusiones sociales

- Investigadores de diversas disciplinas, incluso aquellos con más limitaciones en el manejo de literatura técnica, pueden beneficiarse de un acceso más rápido y directo a la información importante. De esta forma, incluso un estudiante sin mucha experiencia podría ayudar con el avance de las investigaciones.

5.2.2 Repercusiones económicas

- En cuanto a repercusiones económicas, se consigue una reducción de costos en investigación. Las empresas ahorran dinero al optimizar los procesos de investigación. Esto permite a los investigadores pasar menos tiempo buscando información en artículos científicos, lo que reduce el costo de horas laborales para un proyecto. Además, con la

Inteligencia Artificial evitamos errores que podrían llegar a retrasar proyectos generando gastos adicionales.

En conjunto, este proyecto no solo es un avance técnico, sino una herramienta importante en el campo científico y la sociedad en general. Su desarrollo tiene como potencial, transformar la manera en la que se investiga la ciencia en áreas críticas como la química solar.

6. Trabajo futuro

Nuestro sistema actual está desarrollado para generar respuestas a unas preguntas, ya integradas en el modelo, dentro de los artículos científicos. A continuación vamos a plantear algunas mejoras como trabajo futuro para esta herramienta. Las propuestas principales son:

1. Resumen

Una mejora sería añadir una función dentro de la interfaz que hiciera un resumen del artículo científico.

- Descripción: Esta función generaría un resumen de los artículos científicos que se desea leer, destacando los datos más importantes y dando una visión general del contenido.

- Impacto: Los investigadores tendrían un resumen general sobre el artículo sin la necesidad de leerlo entero, agilizando el trabajo de lectura y comprensión de artículos científicos, y así ganar tiempo para otras tareas.

- Desarrollo: Para esto se podrían usar modelos de lenguaje especiales para la generación de texto y que tengan la capacidad de resumir la información de manera correcta y con sentido.

2. Sistema para extracción de datos de tablas e imágenes

Otra funcionalidad que se podría añadir a la interfaz sería un sistema de extracción de respuestas basado en tablas e imágenes de los artículos analizados.

- Descripción: Con esta funcionalidad los usuarios podrían subir un documento y obtener resultados más concretos basados en tablas e imágenes. Con esto, se podría obtener respuestas que podrían pasar desapercibidos.

- Impacto: Nuestro sistema sería una herramienta más completa, al permitir a los investigadores obtener datos de tablas e imágenes de los artículos científicos de forma automática.

- Desarrollo: Para conseguir esta funcionalidad, se necesita entrenar modelos de Inteligencia Artificial que no solo procesen texto, sino que también sean capaces de interpretar las imágenes y las tablas.

3. Mejorar la interfaz de usuario

Aunque nuestra interfaz actual funciona correctamente, se podría mejorar la experiencia del usuario y con esto aumentar el número de usuarios.

- Descripción: Para mejorar la interfaz de usuario hay muchas cosas que se podrían cambiar con el objetivo de que sea más compleja y más atractiva. Una mejora sería incorporar otras funciones como por ejemplo tener la posibilidad de ver en el artículo científico las respuestas generadas subrayadas. Otra funcionalidad sería tener una memoria de todos los artículos científicos ya analizados, así de esta manera no se repetiría el análisis de un mismo artículo científico varias veces. Y como último ejemplo, se podría rediseñar la pantalla inicial para que no fuese la de "Home" y tuviese una pantalla de inicio más atractiva visualmente.

- Impacto: Con esto se conseguiría un diseño más atractivo, lo que podría aumentar el número de usuarios. Además, ganaría más valor la interfaz añadiendo nuevas funcionalidades además de reducir la complejidad en análisis de artículos científicos.

- Desarrollo: Tendríamos que usar otros framework, como JavaScript, para evitar las limitaciones de Streamlit y así poder diseñar y añadir cualquier función sin restricciones.

A pesar de que el sistema actual ya ofrece una gran ayuda a los investigadores de química solar y además les ofrece muchos beneficios, un trabajo futuro podría darles muchas posibilidades para un mejor funcionamiento y aumentar el impacto entre los investigadores científicos. Con estas mejoras se podría conseguir que esta herramienta sea la solución completa para el análisis y extracción de datos en artículos científicos.

7. Discusión de los objetivos

En este apartado vamos a discutir los objetivos planteados inicialmente, con la intención de analizar cuáles se han cumplido y cuáles no. El objetivo general era crear una interfaz web que tenga buen funcionamiento y así conseguir un análisis más rápido de los artículos científicos en el área de la química solar, dando respuestas concretas y justificadas.

El objetivo general se ha cumplido, ya que nuestra interfaz desarrollada en Streamlit se conecta con éxito al backend desarrollado por el Ontology Engineering Group (OEG), devolviendo este las respuestas correctamente basadas en los documentos PDF.

A continuación, se evalúa el cumplimiento de cada uno de los objetivos específicos planteados al inicio del proyecto:

1. **Investigar los fundamentos de los sistemas Question Answering (QA) para poder comprender su funcionamiento.**

   - Objetivo cumplido. Durante la fase inicial de este proyecto, se investigó sobre los sistemas de QA para comprender su funcionamiento.

2. **Investigar y entender los sistemas de extracción Retrieval Augmented Generation (RAG) y modelos de lenguaje de gran tamaño (LLM) ya implementados por el Ontology Engineering Group (OEG), con la finalidad de entender su función para prepararme para su integración.**

   - Objetivo cumplido. Una parte del proyecto se destinó al análisis y comprensión del backend desarrollado por el OEG. Este estudio fue necesario para comprender el funcionamiento del backend y así asegurarnos de que su conexión con el frontend se haga de una forma correcta.

3. **Diseño de la interfaz web utilizando Streamlit.**

   - Objetivo cumplido. La interfaz web se desarrolló con éxito utilizando Streamlit. Permite a los investigadores subir artículos científicos, visualizar las respuestas y descargar resultados en formato PDF o

JSON anotado.

4. **Diseño de una API que permite la conexión de la interfaz web con el servidor para acceder al procesamiento de lenguaje natural basado en LLM.**

   - Objetivo cumplido. Se consiguió conectar la interfaz web con el backend del OEG, permitiendo que la aplicación analice los documentos PDF, y devuelva respuestas en lenguaje natural.

5. **Despliegue de un servicio que se conecte con el servidor.**

   - Objetivo parcialmente cumplido. Aunque no hemos realizado el despliegue final de la interfaz web, actualmente la conexión con el backend se realiza de forma local a través del CLI (Command Line Interface). Esto permite ejecutar la aplicación y procesar documentos desde el entorno local.

6. **Analizar y documentar los resultados obtenidos.**

   - Objetivo cumplido. Se han analizado y documentado los resultados del sistema, permitiendo así evaluar y discutir su funcionamiento.

En general, el proyecto ha cumplido con la mayoría de los objetivos planteados, permitiendo conseguir una herramienta que sirva para el futuro en el ámbito del análisis automatizado de artículos científicos.

8. Conclusiones

Este proyecto ha sido una experiencia muy satisfactoria tanto a nivel académico como personal. A lo largo de todos estos meses, hasta finalizar el proyecto he tenido que superar diferentes desafíos. Estos no solo me hicieron adquirir nuevos conocimientos y explorar nuevas herramientas, sino también poner a prueba mis conocimientos y habilidades. Ha sido una experiencia que me ha ayudado a mejorar mi habilidad de aprendizaje autónomo, resolver distintos problemas que surgen en un proyecto y adaptarme a nuevas tecnologías.

Desde un principio, me sentía muy motivada simplemente por tener la posibilidad de participar en un proyecto que tenga un impacto directo en la investigación científica. Trabajar en el desarrollo de una interfaz para poder devolver respuestas útiles representa un paso importante en mi formación.

Además, para mí haber terminado este trabajo significa haber alcanzado una meta personal, dándome una satisfacción de logro y muy feliz de haber conseguido superar las limitaciones y obstáculos de este proyecto.

Por último, a continuación se mencionan los puntos fuertes y débiles de este proyecto a nivel personal.

8.1 Puntos fuertes

- **Motivación y pasión por aprender.**
  Desde el primer momento, tenía mucha curiosidad por este proyecto, que fue lo que hizo que mantenga un ritmo constante de trabajo, investigando y buscando soluciones para cada problema que me surgía. La motivación personal ha sido mi gran apoyo y lo que ha hecho que me mantenga enfocada y trabajando bien.

- **Adquisición de nuevos conocimientos.**
  Aprendí a utilizar herramientas como Streamlit para el desarrollo de interfaces, o cómo usar conexiones HTTP. Esta combinación me ha permitido aprender y comprender mejor la conexión entre frontend y backend.

- **Resolución de problemas y autonomía.**
  Al enfrentar errores y limitaciones, me encontré en la necesidad de aprender a buscar soluciones por mi cuenta, investigando en

documentación y foros. Este punto fue al principio muy difícil para mí pero poco a poco me fui adaptando a trabajar de forma autónoma. Este proceso de autoaprendizaje me ha hecho más independiente.

- **Impacto del proyecto.**
  Saber que este sistema desarrollado puede ayudar y facilitar el trabajo de los investigadores en el campo de la química solar, añade más valor a todo el esfuerzo invertido.

8.2 Puntos débiles

- **Limitaciones del hardware.**
  Uno de los principales obstáculos fue el uso de mi ordenador personal. Es un poco antiguo, y me iba muy lento al trabajar con modelos de Inteligencia Artificial y documentos extensos. Esta limitación me afectó mucho con el avance del proyecto porque en las pruebas que hacíamos mi ordenador tardaba mucho en ejecutarlas.

- **Dependencia del backend.**
  Al depender del backend desarrollado por el Ontology Engineering Group (OEG), en ocasiones, cuando me surgía un problema, esta dependencia me impedía seguir con el desarrollo.

- **Limitaciones de diseño.**
  Me hubiese gustado que la aplicación fuera más atractiva, pero como llevo diciendo a lo largo del trabajo, Streamlit tiene muchas limitaciones. La personalización y algunas funcionalidades visuales no pudieron hacerse de la forma deseada.

En resumen y como conclusión final, este proyecto ha sido un logro de crecimiento personal y profesional. A pesar de todas las dificultades y obstáculos, ahora me siento mucho más preparada para enfrentar proyectos similares en el futuro. De las cosas más importantes que he aprendido, es que cada error que surge en cualquier ámbito no es más que una oportunidad para aprender y mejorar. Y además, que con paciencia, constancia en el trabajo, ganas de aprender y saber adaptarte a las nuevas tecnologías se puede conseguir cualquier proyecto y avance. Terminar este proyecto no solo significa el cierre de mi etapa académica, sino también el comienzo de nuevas oportunidades y desafíos.

9. Bibliografía

[1]: Mark A. Hanson, Pablo Gómez Barreiro, Paolo Crosetto, Dan Brockington (2023). *The strain on scientific publishing*

[2]: Alexander Dunn, John Dagdelen, Nicholas Walker, Sanghoon Lee, Andrew S. Rosen, Gerbrand Ceder, Kristin Persson, Anubhav Jain (2022). *Structured information extraction from complex scientific text with fine-tuned large language models*

[3]: Amirhosein Toosi, Andrea Bottino, Babak Saboury, Eliot Siegel, Arman Rahmim (2021). *A brief history of AI: how to prevent another winter (a critical review)*

[4]: Jeff Shrager (2024). ELIZA Reinterpreted: The world's first chatbot was not intended as a chatbot at all

[5]: Nikolaos-Ioannis Galanis, P. Vafiadis, K.-G. Mirzaev, G.A. Papakostas (2021). *Machine Learning Meets Natural Language Processing -- The story so far*

[6]: Zhen Wang (2022). *Modern Question Answering Datasets and Benchmarks: A Survey*

[7]: Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, Percy Liang (2016). *Know What You Don't Know: Unanswerable Questions for SQuAD*

[8]: Tom Kwiatkowski, Jennimaria Palomaki, Olivia Rhinehart, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov (2019). *Natural Questions: A Benchmark for Question Answering Research*

[9]: Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, Christopher D. Manning (2018). *HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering*

[10]: Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, Tong Wang (2016). *MS MARCO: A human generated machine reading comprehension dataset*

[11]: Siva Reddy, Danqi Chen, Christopher D. Manning (2019) CoQA: *A Conversational Question Answering Challenge*

[12]: Vaswani Ashish, Noam Shazeer, Niki Parmar, Jacob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin (2017) *Attention Is All You Need*

[13]: Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

[14]: Gokul Yenduri, Ramalingam M, Chemmalar Selvi G, Supriya Y, Gautam Srivastava, Praveen Kumar Reddy Maddikunta, Deepti Raj G, Rutvij H Jhaveri, Prabadevi B, Weizheng Wang, Athanasios V. Vasilakos, Thippa Reddy Gadekallu (2023). *Generative Pre-trained Transformer: A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions*

[15]: Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J. Liu (2019). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*

[16]: Yang Liu, Jiahuan Cao, Chongyu Liu, Kai Ding, Lianwen Jin (2024). *Datasets for Large Language Models: A Comprehensive Survey*

[17]: Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, Haofen Wang (2023). *Retrieval- augmented generation for large language models: A survey.*

[18]: Streamlit, “Streamlit Documentation” https://docs.streamlit.io/

[19]: Norman Paskin (2009). Digital Object Identifier (DOI®) System

[20]: Mozilla Developer Network, “HTTP”,
https://developer.mozilla.org/en-US/docs/Web/HTTP

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Tue Jan 14 20:51:37 CET 2025
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)