****

TÍTULO DEL TFG: Aplicaciones de un LLM con un Motor de Búsqueda

**CONVOCATORIA:** 11 de abril de 2024

**ALUMNO:** Víctor Gutiérrez Tovar

**TUTOR:** Dr. Alfonso Castro Escudero

**GRADO:** Ingeniería de Software

# Índice

[Índice 3](#_Toc159264995)

[Índice de figuras 5](#_Toc159264996)

[Resumen 7](#_Toc159264997)

[Abstract 9](#_Toc159264998)

[1. Introducción 11](#_Toc159264999)

[1.1. Motivación y contexto 11](#_Toc159265000)

[1.2. Planteamiento del problema 11](#_Toc159265001)

[1.3. Objetivos del trabajo 12](#_Toc159265002)

[2. Estado de la cuestión 13](#_Toc159265003)

[2.1. Motores de búsqueda con Modelos Grandes del Lenguaje, *Large Search Models* (LSMs) 13](#_Toc159265004)

[2.1.1 Introducción a los Modelos Grandes de Búsqueda (LSMs) 13](#_Toc159265008)

[2.1.2 Los LSMs de la actualidad 13](#_Toc159265009)

[2.1.3 Diferenciación de los LSM al prototipo propuesto 14](#_Toc159265010)

[2.2. Motores de búsqueda 15](#_Toc159265011)

[2.2.1 Introducción 15](#_Toc159265013)

[2.2.2 Indexación inversa 15](#_Toc159265014)

[2.2.3 Algoritmo de puntuación por consulta (*Scoring*) 16](#_Toc159265015)

[2.2.4 Motores de búsqueda de código abierto 16](#_Toc159265016)

[2.3. Modelos Grandes del Lenguaje, *Large Language Models* (LLMs) 17](#_Toc159265017)

[2.3.1 Introducción 17](#_Toc159265019)

[2.3.2 LLM de código abierto 17](#_Toc159265020)

[2.3.3 Evolución de los LLM 18](#_Toc159265021)

[2.4. Dilemas éticos 18](#_Toc159265022)

[2.5. Aplicaciones específicas 19](#_Toc159265023)

[2.5.1 Aplicaciones en el mundo legal 19](#_Toc159265026)

[2.5.1.1 Diversificación de las leyes españolas 19](#_Toc159265027)

[2.5.2 Aplicación en Farmacología 20](#_Toc159265028)

[2.5.3 Aplicación en un sistema de ficheros 21](#_Toc159265029)

[3. Desarrollo del trabajo 22](#_Toc159265030)

[3.1. Planificación 22](#_Toc159265031)

[3.2. Diseño 22](#_Toc159265032)

[3.3. Desarrollo 22](#_Toc159265033)

[3.4. Pruebas 22](#_Toc159265034)

[3.5. Despliegue 22](#_Toc159265035)

[4. conclusiones y mejoras futuras 23](#_Toc159265036)

[5. Referencias 24](#_Toc159265037)

[5.1. Bibliografía. 24](#_Toc159265038)

[5.2. Webgrafía 24](#_Toc159265039)

# Índice de figuras

# Resumen

El estudio propuesto busca investigar la eficiencia de los Modelos Grandes de Búsqueda (LSM), la integración de Modelos Grandes del Lenguaje (LLM) con motores de búsqueda. Esta investigación surge de la necesidad de superar la limitación de los LLM, que al solo tener conocimiento del contenido con el que se le ha entrenado, no pueden ofrecer información actualizada. Por ello esta integración conseguiría que este sistema, puede mejorar la eficiencia en la obtención de información relevante y actualizada.

El enfoque del proyecto es crear un prototipo LSM capaz de entender consultas escritas con lenguaje natural y formular respuestas con contenido actualizado proporcionado por el motor de búsqueda y de esta manera conocer las fuentes originales de la información. Este prototipo se pondrá a prueba en tres áreas específicas: derecho, farmacología, y un sistema de ficheros, evaluando su capacidad para responder a consultas legales, recomendaciones de medicamentos basadas en síntomas y preguntas sobre los contenidos de ficheros PDF.

Se revisarán las tecnologías actuales de LSM, LLM y motores de búsqueda, mencionando innovaciones como ChatGPT con Bing, Microsoft Copilot, y el modelo Gemini de Google. Se discutirán las consideraciones éticas relacionadas con el uso de LLM, enfatizando la importancia de minimizar sesgos y verificar la fiabilidad de las fuentes.

Este proyecto busca avanzar en la forma de acceder y de utilizar la información. Se espera que los resultados den conocimiento del estado actual de los LSM usando tecnologías de código abierto y de cómo su uso puede facilitar el manejo de grandes cantidades de datos, mejorando nuestra productividad al reducir los tiempos de búsqueda de información.

# Abstract

The proposed study aims to investigate the efficiency of Large Search Models (LSM), the integration of Large Language Models (LLM) with search engines. This research comes from the need to address their main limitation. The LLMs only have knowledge of the content they have been trained on. So, they cannot offer updated information. The integration of both systems, the LLM and the search engine, would improve efficiency when obtaining relevant and constantly updated information.

The main goal of this investigation is to create an LSM prototype capable of understanding queries written in natural language and creating quality responses by processing the content and references provided by the search engine with the LLM. This prototype will be tested in three specific areas: law, pharmacology, and a file system. Then the responses generated to legal questions, drug recommendations based on symptoms, and questions about the contents of PDF files will be evaluated for its ability to respond.

Current technologies of LSM, LLM, and search engines will be reviewed, mentioning innovations such as ChatGPT with Bing, Microsoft Copilot, and Google's Gemini model. Ethical considerations related to the use of LLM will be discussed.

This project explores the ways to advance when accessing information. The results are expected to provide some context of the actual state of LSMs using open-source technologies and how their use can make easier handling large amounts of data, improving our productivity by reducing the time spent when searching for information.

# 1. Introducción

## Motivación y contexto

Las Inteligencias Artificiales (IAs) cada vez están obteniendo un mayor protagonismo en nuestro día a día. La capacidad de aprovechar estas tecnologías nos brinda múltiples ventajas, entre ellas ahorrar tiempo, aumentar la productividad y como resultado conseguir más tiempo que se puede utilizar para dar mayor calidad. Estas tecnologías tienen múltiples aplicaciones, principalmente realizar predicciones, clasificar imágenes, segmentar imágenes, generar imágenes, crear texto, entre otras.

Este estudio se centrará concretamente en las aplicaciones de generación de texto, estas son las que más polémicas dan, principalmente por la gran popularidad de ChatGPT creada por OpenAI o su alternativa *open source*, LLaMa 2. Estas tecnologías se las conocen como Modelos Grandes del Lenguaje (*Large Language Model*, LLM), por estar entrenadas de una gigantesca cantidad de textos de alta calidad.

Las IAs pueden ofrecer información de temas muy generales generalmente de forma rápida y precisa. Aunque existe siempre la posibilidad de que la respuesta dada sea errónea. Estos errores se acentúan sobre todo en respuestas de temas muy específicos. Al estar estas IAs alimentadas de información de múltiples fuentes, la IA es incapaz de identificar de que fuentes se está basando para generar la respuesta. Además, estas IAs solo conocen la información con la que este se entrenó, lo que quiere decir que siempre van a estar desactualizadas, ya que solo tienen conocimiento de la información del último entrenamiento. Actualizar estas IAs con la última información implicaría reentrenarlos, lo que resulta inviable por los costes computacionales y el tiempo requerido.

Para solucionar este problema, surgen los Modelos Grandes de Búsqueda (*Large Search Model*, LSM). Estos modelos ofrecen la posibilidad de acceder a información actualizada, utilizando un motor de búsqueda, que son muy rápidos y efectivos al buscar información actualizada. De esta manera se quita el problema de que el LLM este desactualizado. El LLM se encarga de generar la consulta al motor de búsqueda, procesar las respuestas generadas por el motor de búsqueda y basándose de esta información genera la respuesta.

Los LSM presentan aplicaciones para diferentes contextos. ChatGPT 4 con el motor de búsqueda de Bing, es uno de los LSM más destacados. Esta integración permite consultar información de internet de una manera mucha más cómoda. Siguiendo esta línea, las organizaciones pueden incorporar un LLM como LLaMA2, desarrollado por Facebook con filosofía Open Source con motores de búsqueda como Opensearch mantenido por Amazon Web Services. En esta investigación se busca determinar la eficacia de un LSM para mejorar la velocidad y acceso a la información en diferentes contextos.

## Planteamiento del problema

Para construir un *Large Search Model* (LSM), es necesario un *Large Language Model* (LLM), un motor de búsqueda y un *middleware* para integrar a estas dos.

Para que esta integración sea efectiva hay que determinar que LLM y motor de búsqueda. Estos deben de ser código abierto (*Open Source*), escalables y con un gran apoyo por la comunidad, para sea aplicable en los múltiples contextos.

Dentro de los LLMs de código abierto, LLaMa 2 es uno de los modelos más destacados. Desarrollado por Facebook, este modelo se caracteriza por generar texto con alta precisión en comparación a otros modelos de código abierto. Se adapta de forma eficaz a cualquier contexto, debido al gran cantidad de conocimiento con el que ha sido entrenado. Además, dispone de una gran comunidad. Hace escasos meses, Microsoft anunció un LLM *mobile*, con posibilidad a ejecutarse en dispositivos con recursos computacionales más limitados como ordenadores personales e incluso teléfonos móviles inteligentes. Se lanzó con licencia de código abierto. Por temas de licencia y su bajo coste computacional es otro LLM a considerar.

En el ámbito de los motores de búsqueda, OpenSearch es una de las opciones más acertadas en el ámbito de código abierto [7], ofreciendo una plataforma robusta y escalable para el manejo y acceso a grandes volúmenes de datos. Es de código abierto, tiene un gran apoyo de la comunidad software y la mantiene la gigante de Amazon Web Service.

La integración efectiva de modelos LLM como LLaMa 2 y Phi-2 con el motor de búsqueda OpenSearch implica implementar un *middleware* que facilite la interacción entre ambos sistemas. Python y Node.js son herramientas ideales para el desarrollo de este *middleware*, gracias a su versatilidad, el soporte extenso de bibliotecas, y una comunidad de desarrolladores amplia y colaborativa.

## Objetivos del trabajo

El trabajo tiene como objetivo investigar qué tan eficaces son los Modelos Grandes de Búsqueda (LSM) que tengan un motor de búsqueda alimentado de información propio o especializado. Las conclusiones obtenidas servirán para valorar cuando un sistema LSM puede ayudar a nuestra productividad y en el caso de no conseguirlo, indicar qué progresos son necesarios para que esto ocurra.

Para ello se creará un prototipo que sea flexible con cualquier tipo de información. Este prototipo, al mismo tiempo tendrá la capacidad de poder configurarse, para conseguir mejores resultados en un determinado contexto y tipo de información.

Se pondrán en práctica estos tres escenarios:

* Consultas legales, sobre el código legal español. Con el objetivo de responder a cuestiones legales de cualquier tipo. Estas respuestas ofrecerán las fuentes donde se obtiene dicha información y ayudará a interpretar los distintos resultados.
* Consultas para el mejor fármaco basándose en síntomas. Describiendo un conjunto de síntomas que una persona padezca el LLM procesará esta información para hacer una consulta al motor de búsqueda que nos brindarán de los mejores fármacos según los distintos síntomas y el LLM interpretará a través de los distintos resultados cual puede ser mejor o peor opción.
* Consultas sobre un sistema de ficheros. Crear un sistema que almacene la información de los distintos PDF de un sistema cualquiera, y este sea capaz de responder a las diferentes consultas haciendo referencias de los archivos donde está basando sus respuestas ayudando al usuario a encontrar información de manera más ágil es su sistema de ficheros.

A este prototipo se le evaluará la experiencia de usuario. Centrándose en características como los tiempos de espera (alta, moderada, baja), la calidad de las respuestas (son vagas o efectivas) y la experiencia de uso por los usuarios que la testen.

# Estado de la cuestión

En este apartado se va a repasar el estado actual de los motores de búsqueda, así como las tecnologías que harán posible el desarrollo del trabajo.

## Motores de búsqueda con Modelos Grandes del Lenguaje, *Large Search Models* (LSMs)



### Introducción a los Modelos Grandes de Búsqueda (LSMs)

[18] Un LSM es un concepto nuevo que cambia y cambiará la forma de cómo buscar la información a internet, al combinar las tecnologías de un motor de búsqueda y un Modelo Grande del Lenguaje (LLM). En lugar de seguir una serie de pasos necesarios para poder encontrar la información necesaria en nuestra investigación, esta tecnología simplifica el proceso a un solo *input* para manejar casi todo, dando instrucciones de cómo quieres que haga las tareas escribiendo comandos simples como si estuvieras hablando con él, es decir, utilizando lenguaje natural. Esto hace que las búsquedas por internet sean más directas y podría darnos mejores resultados ya que los LLMs más avanzados son cada vez más capaces de entender las instrucciones que se les solicita responder.

Estos avances nos permiten simplificar de manera considerable los procesos de investigación siendo estos más rápidos incluso con mayor calidad que investigaciones más vagas o cotidianas. Además, los LLM al tener cada vez mayor número de parámetros cada vez son más capaces de entender mejor los contextos de las preguntas y de mejorar mucho más las respuestas.

A diferencia de los modelos LLMs, los LSMs están preparados para comprender grandes volúmenes de datos, es decir son LLMs con adaptaciones para poder realizar consultas a internet con motores de búsqueda, pueden tener otras aplicaciones integradas como la creación de imágenes con texto, leer información directamente de un fichero, interpretar el contenido de una imagen, entre muchas otras. La gran ventaja respecto a un LLM tradicional, es la capacidad de obtener información actualizada en tiempo real.

\* [citaciones del paper]

"In this paper, we introduce a novel conceptual framework called large search model, which redefines the conventional search stack by unifying search tasks with one large language model (LLM)".

"Given a user query and potentially thousands of retrieved documents, the large search model generates the various elements that constitute the SERP, including the ranked document list, document snippets, direct answers, etc.".

"Large language models (LLMs) that leverage self-supervised pre-training and scaling laws have emerged as general-purpose interfaces for natural language understanding and generation... These desirable features position LLMs as a promising option for the unified modeling of search tasks." Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R., & Wei, F. (2024, January). Large Search Model: Redefining Search Stack in the Era of LLMs. In ACM SIGIR Forum (Vol. 57, No. 2, pp. 1-16). New York, NY, USA: ACM.

### Los LSMs de la actualidad

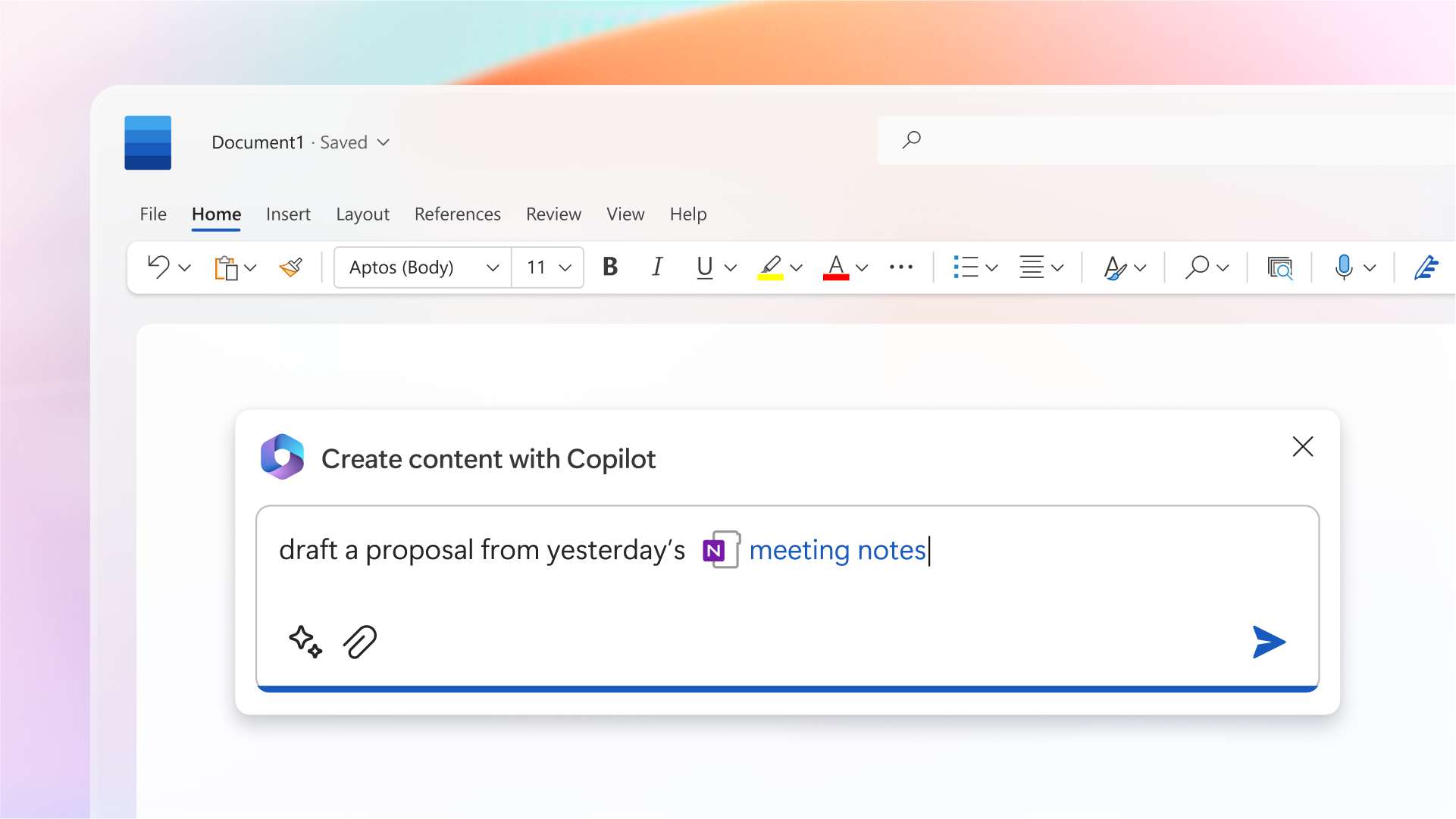
[19] ChatGPT, en colaboración con Bing, es uno de los primeros *chatbots* que integra un Modelo de Lenguaje de Gran Tamaño (LLM, por sus siglas en inglés) con un motor de búsqueda. GPT-4, en particular, tiene la capacidad de determinar cuándo es necesario recurrir al motor de búsqueda Bing para obtener información actualizada en la mayoría de los casos.

A close up of a computer screen

Description automatically generated

ChatGPT with Bing to search the web

[21] Por otro lado, Microsoft Copilot es otro chatbot que hace uso de GPT-4 y el motor de búsqueda de Bing, añadiendo a su funcionalidad la reciente API REST de Microsoft Graph. Esta API permite el acceso a datos empresariales de una organización, resultando en una amplia gama de aplicaciones. Entre las aplicaciones más destacadas se encuentra el uso de Copilot en las herramientas de Microsoft 365, como Word, Excel y PowerPoint. Esto representa una oportunidad significativa para los ejecutivos, ya que les permite acceder a sus datos de manera rápida y sencilla.



Microsoft Copilot

[22] Otro LLM alimentado por una sólida base de datos es Gemini 1.5 de Google, que representa la última novedad tecnológica de la compañía. Se presenta como una alternativa a las ya existentes LLM, como GPT-4, siendo esta última la más popular y utilizada. Gemini también ofrece posibilidades como la generación de imágenes a partir de texto y otras aplicaciones, como la consulta de información mediante su propio motor de búsqueda. Además, destaca por la capacidad de mantener contextos durante más tiempo en comparación con sus competidores.

A screenshot of a chat

Description automatically generated

Google Gemini

Existen otros LLM con capacidades similares como Gemini de Google, pero GPT-4 sigue manteniéndose a la cabeza por su gran popularidad y efectividad.

Los enfoques varían respecto al prototipo mencionado. Estos LSMs utilizan motores de búsqueda como Bing o Google, que fundamentan sus resultados en las múltiples páginas web disponibles en Internet. Dichas páginas reciben un puntaje o scoring basado en el algoritmo de optimización para motores de búsqueda (SEO, por sus siglas en inglés). Los resultados de estos motores presentan tanto ventajas como desventajas. Entre las ventajas, se destacan la recomendación de sitios web seguros, rápidos y bien argumentados. No obstante, la desventaja principal es que el algoritmo no asegura la fiabilidad del contenido, que podría incluso resultar fraudulento.

En el ámbito del software de código abierto, no se observan desarrollos a gran escala ni proyectos específicos orientados a cumplir con los objetivos de este prototipo utilizando LSMs con un enfoque similar. Sin embargo, existen numerosos LLMs de alta calidad que, a pesar de no estar diseñados para integrar directamente motores de búsqueda en sus funcionalidades, ofrecen capacidades significativas en la generación de texto. Un ejemplo destacado es LLaMA 2, desarrollado por Meta. Este modelo es altamente competente en la generación de texto, aunque no incluye la capacidad de utilizar motores de búsqueda para potenciar o complementar esta función, que es central para las necesidades de nuestro prototipo.

### Diferenciación de los LSM al prototipo propuesto

Los LLMs desarrollados por las destacadas corporaciones tecnológicas, como son Google, Facebook, Microsoft y OpenAI, se diseñan con el objetivo de abordar una amplia variedad de problemas mediante el aprovechamiento del conocimiento preentrenado que poseen sus LLMs. Estos modelos en el caso de carecer conocimiento sobre un tema concreto, recuren a motores de búsqueda, u otros sistemas análogos para adquirir datos de internet en tiempo real para posteriormente dar una respuesta. Adicionalmente, ofrecen la capacidad de procesar diversos tipos de entradas, principalmente el texto, imágenes, audio e incluso vídeo.

El propósito de esta investigación se centra en la utilización de un motor de búsqueda propio, el cual se alimenta de información específica y técnica. Esto contrasta con los LLM mencionados previamente, los cuales proporcionan información sin ofrecer citaciones, aun cuando dicha información pueda ser verídica. La inclusión de citas o referencias explícitas es esencial para respaldar la información, permitiéndonos así argumentar y corroborar la veracidad de los diferentes hechos comentados. Este aspecto tiene mucha importancia en diversas áreas, como el derecho, la investigación académica y la medicina.

La alternativa más cercana a la propuesta es la implementación de ChatGPT, configurado para realizar búsquedas y generar citaciones. No obstante, es relevante señalar que, en la mayoría de los casos, las citaciones proporcionadas pueden originarse de páginas web cuyo contenido, si bien es informativo, no alcanza el nivel académico requerido. Este fenómeno se debe, en gran medida, a estrategias orientadas a mejorar el posicionamiento en buscadores (SEO) con el objetivo de maximizar beneficios económicos, lo cual puede traducirse en una calidad de la información disminuida debido a la escasez o la inexistencia de fuentes confiables.

## Motores de búsqueda



### Introducción

Al realizar búsqueda para la definición de un motor de búsqueda como Google o Bing, se la define de una manera muy similar, esta en concreto.

“A search engine is a service that allows Internet users to search for content via the World Wide Web (WWW). A user enters keywords or key phrases into a search engine and receives a list of Web content results in the form of websites, images, videos, or other online data that semantically match with the search query.”

Margaret Rouse, (2023), <https://www.techopedia.com/definition/12708/search-engine-world-wide-web>

A pesar de que esa definición está enfocada a los motores de búsqueda de las páginas web de internet como Google o Bing, se puede extrapolar a una definición más general con la siguiente definición.

Un motor de búsqueda es aquel software que mediante términos o conjunto de estos es capaz de ofrecer respuestas de alta calidad al evaluar el número de repeticiones de los términos asociados a los documentos de manera instantánea. Esto se consigue al indexar de manera inversa toda la información a la que accede el motor de búsqueda.

### Indexación inversa

[8][9] Para entender qué es la indexación inversa, se puede explicar con la tecnología de código abierto Lucene, la cual se aplica en la mayoría de los motores de búsqueda como Elasticsearch y Opensearch, además de en múltiples bases de datos.

La indexación tradicional se basa en identificadores únicos que referencian a un documento. En la indexación inversa, la indexación se basa en términos, que referencian a documentos. Los términos se traducen a cadenas de texto, que en lo general son palabras o conjunto de ellas.

Al indexar un documento, el motor de búsqueda almacena los nuevos términos en el índice inverso. Cada índice inverso apunta a los diferentes documentos que contienen dicho término, como también datos de su frecuencia y donde se encuentra dentro del documento. Este índice inverso esta optimizado para para recuperar de manera eficiente y rápida como Arboles de búsqueda, Tablas Hash u otros sistemas de búsqueda eficiente.

La indexación inversa permite por ello búsquedas a tiempo real de información en grandes cantidades de datos, gracias a este proceso durante la escritura de los nuevos documentos. Por este mismo motivo esta tecnología se utiliza para los motores de búsqueda de internet, grandes sistemas de almacenamiento de información para textos grandes, detección de anomalías en logs entre muchas otras aplicaciones.

A diagram of a diagram

Description automatically generated

Figure 3 – Indexing process.

Proceso de indexación inversa:

1. Obtenemos un documento con campos(*fields*) y valores(*values*), identificamos directamente los términos, la posición de estos y otros metadatos para búsquedas más avanzadas como el desplazamiento(*offsets*).
2. Los términos una vez identificados, se modifican o rechazan para almacenar solo aquellos términos relevantes. Una vez pasan el filtro, estos se normalizan.
3. Se vincula cada término con los documentos en los que aparece junto a sus localizaciones, esto se almacena en las *posting lists*.
4. Se crean vectores de frecuencia *term vectors* para cada documento, donde por cada término se almacenan metadatos como la frecuencia

### Algoritmo de puntuación por consulta (*Scoring*)

Al realizar una consulta, se realiza una puntuación o *score* de los documentos que contengan los términos de la consulta. Esto permite al motor de búsqueda priorizar a aquellos documentos que tengan mayor prioridad mediante un algoritmo. En los motores de búsqueda más modernos se utilizan un conjunto de algoritmos para obtener esta valoración. En Elasticsearch y Opensearch la puntuación (*score*) se calcula con el módulo BM25, que es una mejora del algoritmo TF-IDF, que a diferencia del BM25, no tiene en cuenta los tamaños de los documentos.

[](https://www.instaclustr.com/blog/opensearch-and-elasticsearch-architecture/)

La puntuación es igual al producto de TF, IDF y Norm. A continuación, se explicarán brevemente cada una de estas siglas:

* Frecuencia de términos (*Term Frequency*, TF): Es un algoritmo que simplemente mide la cantidad de repeticiones de los términos, valorando más a aquellos documentos que tengan más repeticiones de términos.
* Frecuencia de términos inversa (*Inverse Document Frequency*, IDF): Es un algoritmo que se encarga de valorar la rareza de un término, cuando menos veces se repita este término, más raro es y mayor puntuación tendrá.
* Normalización (*Normalization*, NORM): Su propósito es evaluar la repetición de los términos en proporción al tamaño del texto.

Con este módulo se consigue evaluar los documentos de una manera muy ágil y rápida, en gran parte por su baja complejidad computacional.

### Motores de búsqueda de código abierto

[7] [10] En los últimos años Elasticsearch ha sido uno de los motores de búsqueda más utilizado y preferido por la comunidad de software en gran parte por de código abierto, pero en el año 2021 la organización que se encarga de mantener esta tecnología tomo la decisión de dejar de ser de código abierto.

Ante esta situación surgió la alternativa Opensearch que continuaría con el desarrollo de Elasticsearch desde el último *fork* con licencia de código abierto (Apache 2). Por esto, con apenas 3 años de desarrollo, Opensearch se coloca actualmente como el cuarto motor de búsqueda más utilizado. Al basarse de Elasticsearch toda la documentación sobre Elasticsearch puede ser reaprovechada en esta alternativa ya que la forma de configurarse y realizar las *queries* es prácticamente idéntica. Por estos motivos Opensearch será el candidato principal como motor de búsqueda en este prototipo.

A screenshot of a computer

Description automatically generated

DB-Engines Ranking of Search Engines

## Modelos Grandes del Lenguaje, *Large Language Models* (LLMs)



### Introducción

[11] Los Modelos Grandes del Lenguaje (LLMs), son sistemas que utilizan redes neuronales entrenados para entender y generar texto de manera avanzada. Para lograr esto actualmente se requieren enormes cantidades de datos que contengan texto para aprender patrones del lenguaje, lo que les permite realizar múltiples tareas como la traducción de idiomas, la síntesis de un texto y otras aplicaciones. Actualmente ya existen LLMs avanzados como son GPT-4, GPT-3, LLaMA 2, Phi-2 y PALM entre otros.

Los LLMs se basan en la arquitectura *Transformer*, que es lo que le permite entender y generar texto basándose en un contexto dado, al mismo tiempo es capaz de hacer transformaciones de texto, para que este se pueda adecuar mejor al contexto particular dando una gran flexibilidad.

[12] La arquitectura *Transformer*, es una arquitectura de Redes Neuronales que a diferencia de otras arquitecturas como las recurrentes (RNN) o convolucionales (CNN), logran mediante mecanismos de atención entender contextos y estructuras gramaticales.

El elemento clave de esta arquitectura están en las capas de atención autodirigidas. En lugar de procesar secuencias paso a paso, como las capas LSTM (*Long Short Term Memory*), el modelo observa y da peso a todas las partes de la entrada en una misma iteración. Este mecanismo es la clave de que los LLMs sean capaces de entender estructuras gramaticales.

Elementos relevantes de las Redes *Transformer*:

* Capa Multiconsulta
* Capa Multicabeza
* Embeddings RoPE
* Función de activación GeGLU

### LLMs de código abierto

[15] El LLM de código abierto más destacado es LLaMa 2 [14], tiene licencia “llama 2” que a pesar de ser de código abierto incluyen varias restricciones en su uso, como la prohibición de del uso para entrenar otros LLMs, uso de la marca Meta, para actividades ilegales como el tráfico de humanos y discriminatorias, uso comercial limitado a no más de 700 millones de usuarios activos al mes, entre los más destacados. Independientemente de las restricciones anteriormente mencionadas, se puede utilizar sin problemas para realizar investigaciones siempre que cumplamos con las restricciones, las cuales tienen bastante sentido para evitar malos usos.

Recientemente, el 12 de diciembre del 2023, Microsoft lanzo una alternativa de código abierto real, con licencia Apache 2, en la que el LLM de Microsoft phi-2 [13] era mejor en distintos escenarios al actual LLaMa 2, además de que este LLM es *mobile* ya que se puede ejecutar en un ordenador e incluso en un teléfono móvil.

A screenshot of a computer screen

Description automatically generated

Al tener menos de la mitad de los parámetros en comparación a la versión más reducida de LLaMa 2, 2.7 miles de millones de parámetros frente a 7 miles de millones de parámetros, tiene un costo computacional considerablemente menor y además consigue mejores resultados. Independientemente de estas ventajas, Phi-2 para el caso de uso de la investigación podría ser una muy buena opción ya que nos otorga un mejor rendimiento que la alternativa LLaMa 2. A pesar de estos resultados al testear este modelo, se observa que aunque su contenido pueda tener calidad, esta se ve comprometida por la gran cantidad de ruido que se introduce al generar texto.

[31] Durante los primeros meses del año 2024, se publicaron otros LLMs de código abierto relevantes, como Gemma de Google y Mistral con origen europeo, junto a LLaMa 2 en su versión más pequeña, todos ellos son modelos de 7B de parámetros los cuales pueden utilizarse en una amplia cantidad ordenadores personales con una tarjeta gráfica modesta. Por ello destacan en su eficiencia y sostenibilidad ambiental manteniendo un nivel de compresión alto a pesar del menor número de parámetros.

[A graph of different colored bars

Description automatically generated](https://medium.com/@kagglepro.llc/gemma-vs-llama-vs-mistral-a-comparative-analysis-with-a-coding-twist-8eb4d849e4d5)

Gemma vs Llama vs Mistral: A Comparative Analysis with a Coding Twist

Gemma, por ejemplo, integra mecanismos de atención multiconsulta y multicabeza, embeddings RoPE y activaciones GeGLU, entrenándose con un dataset enorme de hasta 6 trillones de tokens centrados en textos en inglés. Por otro lado, Mistral se centra en la eficiencia y la sostenibilidad ambiental, con técnicas de entrenamiento que mejoran la eficiencia de aprendizaje y la responsabilidad del modelo. En una evaluación comparativa, Gemma supera a los modelos LLaMa 2 y Mistral en benchmarks académicos, especialmente en tareas que requieren una comprensión profunda como matemáticas, ciencia, codificación y tareas de razonamiento. Estos benchmarks son orientativos ya que en los LLMs no suelen reflejar otros aspectos como la experiencia de usuario, el ruido que generan en las respuestas y otros aspectos relevantes que solo son percibidos al interactuar con el modelo.

### Evolución de los LLMs

En 1947 Alan Turing realizo su tan conocida prueba para determinar si un texto era generado por un ordenador o un ser humano. Desde ese momento, se creó la necesidad de intentar pasar estas pruebas, pero para ello es muy importante dotar a las máquinas de un manejo fluido del lenguaje, esa intrincada red de expresiones que nos define como seres humanos. A lo largo de los años, hemos pasado de simples modelos estadísticos a sofisticadas redes neuronales, centrando nuestros esfuerzos en entender y generar el lenguaje de manera cada vez más precisa.

En estos últimos años se ha observado un gran salto cualitativo en los LLMs, que han revolucionado el campo del procesamiento del lenguaje natural, gracias a su capacidad de aprender de extensos corpus de texto. Además, por las distintas investigaciones en este campo se observa que a mayor tamaño del modelo (es decir, más parámetros y más datos de calidad), mejor es su rendimiento y más sorprendentes son sus capacidades, incluso logrando aprender del contexto de cada conversación, cualidad que los modelos más pequeños no pueden alcanzar.

Entre ellos destaca ChatGPT, que ha causado una sensación de tener una compresión casi humana, por su avanzada capacidad de conversación y de generación de texto.

A pesar de estos logros mencionados, persisten grandes desafíos como la posibilidad de errores o la generación de contenido con poca ética, temas en los que se sigue trabajando e investigando para ir mitigándolos. Este esfuerzo busca mitigar los riesgos para sobre todo aprovechar al máximo el vasto potencial de los LLMs para aplicaciones futuras.

## Dilemas éticos

Los LLMs están entrenados utilizando una vasta cantidad de datos obtenidos de internet, los cuales pueden estar impregnados de sesgos y discriminación. Todos los LLMs invierten un esfuerzo considerable en restringir sus aplicaciones a fin de prevenir el fomento de actividades ilegales que podrían tener un impacto negativo significativo en la sociedad. No obstante, a pesar de estas precauciones, aún sigue existiendo la posibilidad de eludir dichas restricciones y hacer un uso malintencionado de estas tecnologías. Además, los LLMs son susceptibles de generar respuestas falsas o incorrectas, las cuales, al ser justificadas por el modelo, tienen el potencial de engañar a los usuarios e influir negativamente en la toma de decisiones, con diversas consecuencias adversas.

Al implementar un LSM, se mitiga uno de los principales dilemas éticos asociados a los LLMs, la ausencia de referencias a las fuentes originales. Esto permite verificar la información generada por el LLM con las fuentes primarias, facilitando así un mayor grado de confiabilidad.

Otro aspecto relevante es el impacto potencial de estas tecnologías en los métodos tradicionales de investigación. La facilidad y rapidez que ofrecen pueden llevar a ignorar fuentes importantes y relevantes que, por no estar digitalizadas o ser menos accesibles en línea.

## Aplicaciones específicas



### Aplicaciones en el mundo legal

Análisis de Legislación y Jurisprudencia

[23] En la investigación de Chay Brooks, Cristian Gherhes and Tim Vorley publicada el 27 de enero del 2020 muestra como la cultura de las corporaciones legales y otros modelos de negocio muestran una resistencia cultural o desinterés por adaptar sus maneras de trabajar con las nuevas tecnologías. Aunque las inteligencias artificiales (IA) puedan ofrecer una mayor eficiencia y posibilidad de mejorar la calidad del trabajo y por esta cultura en estos modelos de negocio está ralentizando el progreso en estos sectores.

La implementación de un sistema LSM en este sector, podría ser un buen punto de partida para cambiar esta perspectiva negativa a las nuevas tecnologías. Aunque estos sistemas ofrecen una inmensa cantidad de ventajas, su implementación podría verse como un paso más gradual en gran parte por la posibilidad a errores que existe, por este motivo seguirá existiendo la necesidad de que un abogado pueda confirmar lo que este LSM le está indicando y referenciando. El uso de estos LSM como *chatbots* representa una gran oportunidad a los abogados para poder consultar información legal que ellos conocen, pero necesitan revisar. Al mismo tiempo este chatbot se puede adaptar a los individuos que carezcan de conocimiento legal ayudándoles a resolver problemas legislativos más cotidianos sin la necesidad de ser resuelta por un profesional.

Automatización de búsqueda de precedentes: hacer búsquedas a tiempo real sobre la base entera de las leyes.

Entre las aplicaciones más destacadas en este LSM para este sector se encuentran:

* **Consultas legales** mediante uso de lenguaje natural, las leyes no concentran todas las posibles casuísticas que se pueden dar, por ello se enfocan a legislar principios generales en vez de explícitos. Por ejemplo, en el caso de la quema de un coche, al acusado no se le aplica la ley de quema de coches, ya que además no existe, si no la ley de daños materiales, este concepto se puede aplicar a contextos más complejos sin la necesidad de tener un alto conocimiento legal.
* Por otro lado, los profesionales con conocimiento legal pueden aprovechar este sistema para realizar **búsquedas más directas** sobre una ley concreta con citaciones directas sobre las referencias en las que se está basando.
* Al utilizar un motor de búsqueda este siempre puede ser actualizado con la **última información** ofreciendo siempre con mayor prioridad las últimas revisiones.

Esta aplicación se centrará en la ley española.

#### Diversificación de las leyes españolas

La página web www.conceptosJuridicos.com describe el orden jerárquico de la leyes y normas españolas de la siguiente forma:

“Sistema de jerarquía de las normas y leyes en España

1. Constitución Española
2. Los diferentes reglamentos y directivas de la Unión Europea, los cuales deben ser directamente aplicables y no necesiten trasposición al correspondiente ordenamiento español.
3. A nivel de jerarquía, le siguen todos los tratados internacionales que hayan sido debidamente ratificados por el Estado español.
4. En el mismo orden, le siguen todas aquellas leyes que hayan sido emanadas de las Cortes Generales, como es el caso de las leyes orgánicas y leyes ordinarias.
5. Siguen las normas con rango de ley, las cuales son emanadas por el poder Ejecutivo o Gobierno mediante un real decreto ley o un real decreto legislativo.
6. Finalmente, en la jerarquía normativa están todos los reglamentos dictados por el Gobierno, como es el caso de las órdenes de comisiones delegadas, los [reales decreto](https://www.conceptosjuridicos.com/real-decreto/), las órdenes ministeriales, las instrucciones y circulares, entre otros.
7. Junto a todo ello, se le puede agregar a dicha pirámide de normas jerárquicas, todas las leyes y reglamentos que hayan sido dictadas por las comunidades autónomas.” <https://www.conceptosjuridicos.com/jerarquia-normativa/>.

La jerarquía de las leyes y normas españolas presentan una gran complejidad sobre todo por la constante actualización de estas.

El Boletín Oficial del Estado es un medio de publicación que se utiliza para manifestar todos los nuevos cambios.

“El "Boletín Oficial del Estado", diario oficial del Estado español, es el medio de publicación de las leyes, disposiciones y actos de inserción obligatoria.

El "Boletín Oficial del Estado" se publica todos los días del año, salvo los domingos. No obstante, el Ministerio de la Presidencia, Relaciones con las Cortes y Memoria Democrática puede ordenar la publicación de números extraordinarios.” Ministerio de la Presidencia, (2008), [Real Decreto 181/2008](https://www.boe.es/buscar/act.php?id=BOE-A-2008-2389).

Esta constante actualización complica aún más que los ciudadanos se mantengan actualizados de las últimas novedades jurisdiccionales, por lo que un sistema que este actualizado y pueda ayudar a encontrar información actualizad de la ley puede ser muy útil para resolver consultas legales.

El Boletín Oficial del Estado (BOE) dispone de varios sistemas abiertos para consultar toda la ley española. Existe uno para los sumarios del BOE y otro para la legislación. Este último se basa en un sistema de la Unión Europea nombrado *European Legislation Identifier* (ELI) [27], en el que consiste en la generación de URLs únicas por cada reforma de la ley. Estos sistemas permiten acceder a documentos legislativos siguiendo unos patrones de búsqueda típicos de la abogacía, de manera secuencial siguiendo este patrón: tipo de legislación, año y mes y finalmente el número de legislación. Es un sistema muy efectivo para almacenar las leyes de manera ordenada y duradera, pero sin un motor de búsqueda, encontrar información más específica puede ser bastante costoso en tiempos. Por ello existe el motor de búsqueda propietario del BOE, para la legislación española, que se basa en términos, por lo que no ofrece la posibilidad de consulta con lenguaje natural, es decir, un individuo particular probablemente le resultará difícil encontrar información sobre sus cuestiones legales, además hay que tener en cuenta de la complejidad del vocabulario utilizado en estos textos legislativos, que dificultan aún más estas tareas.

### Aplicación en Farmacología

Implementar un sistema tipo chatbot en el mundo farmacéutico podría ser muy interesante ya que podría ayudar a médicos y farmacéuticos a la hora de tomar la decisión de que medicamentos recetar. En el hipotético caso de que nuestro prototipo pueda desempeñar esta función de manera efectiva, siempre este debe estar respaldado por un experto ya sea un médico o farmacéutico.

En el mundo Farmacológico existen gran variedad de bases de datos, muchas de ellas privadas y algunas públicas. Tras una investigación existen muchas bases de carácter médico, publicaciones médicas de las últimas novedades, alertas sobre riesgos encontrados en medicamentos entre muchos otros. Para este caso práctico nos interesa tener los datos de todos los medicamentos actuales y actualizados. La fuente DrugBank [28] tiene registrada todos los medicamentos existentes en un archivo con todos los datos de esta materia (*dataset*). Solo se puede acceder a este *dataset* para realizar estudios a nivel académico y sin ánimo de lucro. Independientemente DrugBank debe autorizar la aplicación. Independientemente de este *dataset* que puede ser muy interesante para la aplicación que queremos realizar existen varias alternativas. [29] Una de ellas es la API de openFDA donde se encuentran también estos medicamentos etiquetados de una manera muy técnica. La ventaja de esta última opción es la posibilidad de información más técnica, de mayor calidad y con la posibilidad de poder actualizarse de manera automatizada. La principal desventaja es la ingesta cantidad de información del medicamento que puede saturar la capacidad de procesamiento del LLM. Para evitar este problema se optaría por reducir el número de parámetros de todos los medicamentos para solo acceder a esos parámetros que sean estrictamente necesarios para esta aplicación.

En España existe la agencia española de medicamentos y productos sanitarios (CIMA), con la que podemos acceder a todos los médicos autorizados desde un Excel [30]. El inconveniente es que, para poder acceder a la información completa de un medicamento, es necesario consultar a una url, es decir habría que realizar un paso extra por cada uno de los medicamentos para poder almacenar la información de estos medicamentos, siendo esto un gran inconveniente a la hora de actualizar los datos. Al mismo tiempo, existe el mismo problema que con openFDA, la información de cada medicamento es muy extensa y puede saturar al LLM del prototipo.

Muchos otros gobiernos tienen su símil a estos sistemas y generalmente todos ofrecen a los médicos un buscador para poder consultar información de los medicamentos por nombre de medicamento o principio activo. A diferencia de nuestra propuesta que, a parte de esta posibilidad, también quiere ofrecer la posibilidad de consultar por síntomas y además con consultas utilizando lenguaje natural y respuestas más directas a las cuestiones propuestas, agilizando este proceso y ahorrar tiempo.

### Aplicación en un sistema de ficheros

Aplicar este prototipo a los sistemas de ficheros de nuestro propio ordenador. Podría ser desde todo el sistema, o implementarlo desde un directorio concreto para mantener un contexto con lo que el LLM y el motor de búsqueda puedan funcionar de manera óptima. Para simplificar el prototipo, se limitará a unos tipos de ficheros concretos como PDFs.

Como estudiante, alguna vez he sentido la necesidad de encontrar un contenido en un PDF inmenso o en una materia entera con PDFs muy extensos, la capacidad de poder explorar la información de todos estos documentos desde un mismo sitio es algo que años atrás no me podía imaginar y al mismo tiempo de que fuera posible, usar un LLM desde mi propio ordenador que procese esa información para poder resolver de una manera más directa las incertidumbres a la hora de estudio. Este sistema puede ser muy útil a la hora de estudiar y resolver dudas durante el aprendizaje. Como persona crítica este sistema tiene que ofrecer respuestas verídicas y justificadas del archivo del que este se está basando.

El prototipo para esta aplicación estará orientado en un contexto de estudiante de ingeniería del software, pensada para que esta pueda ser aplicada en otros contextos de una manera sencilla.

# Desarrollo del trabajo

## Planificación

Para realizar el diseño del prototipo y que alcance unas metas dignas donde este prototipo, tenga una gran base para la cual en un futuro pueda ser actualizada con nuevas versiones de motores de búsqueda o del LLM como también el refinamiento del *middleware*, el cual será la pieza clave de este prototipo.

Para crear esta aplicación o servicio LSM, se optará por desarrollarlo en Docker, por su simplicidad a la hora de desplegar los servicios y de poder replicarlos en cualquier otro sistema.

## Diseño

### Arquitectura

La aplicación se compone de tres principales pilares, los cuales los abstraemos a 3 distintos contenedores. Este prototipo se despliega desde un fichero “docker-compose.yaml” con las imágenes respectivas de cada container

A screenshot of a computer

Description automatically generated

* **LLM**: Se creará una imagen de Docker partiendo de un Ubuntu 22.04 con Nvidia Toolkit preinstalado para poder utilizar la aceleración hardware, es decir la GPU con el fin de lograr una mejor experiencia al utilizarlo. Por otro lado se utilizaran las abstracciones de Hagging Face, una herramienta DevOps similar a GitHub centrada en publicar modelos de Inteligencia Artificial, principalmente modelos generativos como los LLMs y los GANs (Modelos de Generación de Imágenes). Finalmente el modelo de generación de texto o LLM será utilizado mediante una API REST http, con el fin de obtener términos mediante un texto como de también conclusiones a partir de un texto y unos documentos relacionados al contexto de la pregunta que provendrán del LLM.
* ***Middleware***: Pieza clave que se encarga de orquestar la comunicación entre los servicios del LLM y Opensearch para poder resolver las distintas consultas. Al mismo tiempo, el middleware tiene que estar capacitado de poder conectarse a otros servicios para obtener datos, ejemplos pueden ser bases de datos o APIs ya sea con el fin de alimentar el motor de búsqueda como de incluso adaptar el flujo del *middleware* para otros fines que no se exploraran en esta investigación. En definitiva, el desarrollo de un middleware completamente configurable con el fin de poder adaptarse a las distintas aplicaciones como incluso a otras posibilidades para explorar en un futuro.
* **Motor de búsqueda** (OpenSearch): En este caso estará el container oficial open-source de OpenSearch. Este container se encargará de almacenar toda la información objetiva de cada aplicación y al mismo tiempo de resolver las preguntas en base a unos términos que el LLM capturará a partir de un texto o consulta.

### Flujo

Flujo que se recorre desde que se realiza la consulta hasta que obtenemos una respuesta:

1. Se captura el input de la consulta desde la API del middleware.
2. A partir del mensaje recibido, se obtienen los términos con el LLM.
3. Recibimos los términos procesados por el LLM y realizamos la consulta con dichos términos al motor de búsqueda (OpenSearch),
4. Obtenemos los documentos con mayor *scoring* que el motor de búsqueda nos ofrece.
5. Con los documentos y la pregunta inicial obtenemos unas conclusiones al procesarlo con el modelo LLM.
6. Finalmente, se mandará la respuesta con la pregunta realizada, los términos asociados a dicho texto, los documentos asociados a los términos y las conclusiones del LLM.

A black and white diagram with white text

Description automatically generated

## Desarrollo

* + 1. **.**
    2. **.**
    3. **.**

## Pruebas

.

## Despliegue

# conclusiones y mejoras futuras

* 1. **Conclusiones**
  2. **Mejoras futuras**

# Referencias

## Bibliografía.

[0] "Artificial Intelligence: A Modern Approach" de Stuart Russell y Peter Norvig

[1] "Deep Learning" de Ian Goodfellow, Yoshua Bengio y Aaron Courville

[3] "The Singularity is Near: When Humans Transcend Biology" de Ray Kurzweil.

[4] "Artificial Intelligence and Ethics: An Interdisciplinary Approach" de Lucas Introna y Sarah E. Turner.

[5] "Algorithms of Oppression: How Search Engines Reinforce Racism" de Safiya Umoja Noble.

## Webgrafía

[7] Search engine ranking <https://en.wikipedia.org/wiki/List_of_search_engines>, <https://db-engines.com/en/ranking/search+engine>

[8] Białecki, A., Muir, R., Ingersoll, G., & Imagination, L. (2012, August). Apache lucene 4. In SIGIR 2012 workshop on open source information retrieval (p. 17). sn.

[9] Indexación inversa de Lucene, explicación de la arquitectura y scoring de Elasticsearch / Opensearch <https://www.instaclustr.com/blog/opensearch-and-elasticsearch-architecture/>

[10] Cambio de licencia de Elasticsearch <https://www.elastic.co/pricing/faq/licensing>

[11] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong et al., “A survey of large language models,” arXiv preprint arXiv:2303.18223, 2023

[12] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

[13] Javaheripi, M., Bubeck, S., Abdin, M., Aneja, J., Bubeck, S., Mendes, C. C. T., ... & Gopi, S. (2023). Phi-2: The surprising power of small language models. Microsoft Research Blog.

[14] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., ... & Scialom, T. (2023). Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288.

[15] List of LLMs <https://github.com/RUCAIBox/LLMSurvey?tab=readme-ov-file#list-of-llms>

[18] Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R., & Wei, F. (2024, January). Large Search Model: Redefining Search Stack in the Era of LLMs. In ACM SIGIR Forum (Vol. 57, No. 2, pp. 1-16). New York, NY, USA: ACM.

[19] Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., ... & McGrew, B. (2023). Gpt-4 technical report. arXiv preprint arXiv:2303.08774.

[21] Introducing Microsoft 365 Copilot – your copilot for work <https://blogs.microsoft.com/blog/2023/03/16/introducing-microsoft-365-copilot-your-copilot-for-work/>

[22] Saeidnia, Hamid Reza. (2023). Welcome to the Gemini era: Google DeepMind and the Information Industry. Library Hi Tech News. 10.1108/LHTN-12-2023-0214.

[23] Brooks, C., Gherhes, C., & Vorley, T. (2020). Artificial intelligence in the legal sector: pressures and challenges of transformation. Cambridge Journal of Regions, Economy and Society, 13(1), 135-152.

[24] Sistema de jerarquía de las normas y leyes en España <https://www.conceptosjuridicos.com/jerarquia-normativa/>

[25] BOE <https://www.boe.es/diario_boe/ayuda.php>

[26] Como buscar información en el BOE <https://www.boe.es/buscar/ayudas/legislacion_actualizada.php#faq001>

[27] Sistema de búsqueda ELI BOE

<https://elidata.es/documentacion_tecnica/especificacion_tecnica.php#modelo>

[28] Dataset All drugs <https://go.drugbank.com/releases/latest>

[29] openFDA Drug Labeling <https://open.fda.gov/data/downloads/>

[30] CIMA <https://cima.aemps.es/cima/publico/buscadoravanzado.html>

[31] Gemma vs Llama vs Mistral: A Comparative Analysis with a Coding Twist

<https://medium.com/@kagglepro.llc/gemma-vs-llama-vs-mistral-a-comparative-analysis-with-a-coding-twist-8eb4d849e4d5>