|  |  |
| --- | --- |
| **Generación aumentada por recuperación para grandes modelos de lenguaje de código abierto:**  **Un caso de estudio en ayudas públicas del Gobierno de España.** | |
|  | |
|  | **Desiderio Martí Alcaraz**  Máster en Ciencia de Datos  Área 2 - Aula 1    **Tutor/a de TF**  José Luis Iglesias Allones  **Profesor/a responsable de la asignatura**  Josep-Anton Mir Tutusaus, Esther Ibáñez Marcelo  14/12/2024 |

[Llicència de Creative Commons](http://creativecommons.org/licenses/by-nc-nd/3.0/es/)

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada [3.0 España de Creative Commons](http://creativecommons.org/licenses/by-nc-nd/3.0/es/)

Ficha del Trabajo Final

|  |  |
| --- | --- |
| **Título del trabajo:** | Generación aumentada por recuperación para grandes modelos de lenguaje de código abierto: Un caso de estudio en ayudas públicas del Gobierno de España. |
| **Nombre del autor/a:** | Desiderio Martí Alcaraz |
| **Nombre del Tutor/a de TF:** | Jose Luis Iglesias Allones |
| **Nombre del/de la PRA:** | Josep-Anton Mir Tutusaus, Esther Ibáñez Marcelo |
| **Fecha de entrega:** | 12/2024 |
| **Titulación o programa:** | Máster en Ciencia de Datos |
| **Área del Trabajo Final:** | Área 2 - Aula 1 |
| **Idioma del trabajo:** | Castellano |
| **Palabras clave** | Web scraping, RAG, Open LLMs |
| **Resumen del Trabajo** | |
| Este Trabajo de Fin de Máster (TFM) tiene como objetivo desarrollar un sistema basado en Retrieval-Augmented Generation (RAG) y modelos de lenguaje abiertos (Open LLMs) para la extracción automatizada de información relevante de la web del Sistema Nacional de Ayudas y Subvenciones Públicas.  La finalidad del proyecto es mejorar la accesibilidad y eficiencia en la búsqueda de subvenciones, permitiendo a los usuarios obtener respuestas concisas y contextualmente correctas basadas en datos públicos disponibles en la web.  En el contexto actual, el acceso a la información sobre ayudas y subvenciones puede ser complicado debido a la gran cantidad de datos y la forma de consumo de esta información. Para abordar este desafío, el sistema combina el web scraping con técnicas RAG proporcionando respuestas basadas tanto en el contenido recuperado como en las capacidades de los LLMs.  Queremos plantear la realización de una especie de "asistente" virtual que pueda responder a preguntas concretas sobre el tema comentado  Se pretende implementar una aplicación que mediante estas tecnologías podamos (1) extraer los datos mediante web scraping de las páginas de subvenciones, (2) almacenar esta información, (3) usar un modelo LLM abierto para generar respuestas a partir de esta información, y (4) evaluar del rendimiento del sistema en términos de precisión, relevancia y utilidad de las respuestas.  Los resultados del proyecto deben demostrar que la combinación de RAG y Open LLMs mejora significativamente la experiencia de usuario al ofrecer respuestas precisas y rápidas sobre las ayudas disponibles.  Las conclusiones deben destacar el potencial de esta tecnología para aplicaciones en áreas con grandes volúmenes de datos públicos y su aplicabilidad a datos privados. | |
| **Abstract** | |
| This Master's Thesis (TFM) aims to develop a system based on Retrieval-Augmented Generation (RAG) and open language models (Open LLMs) for the automated extraction of relevant information from the website of the National Aid and Subsidies System Public.  The purpose of the project is to improve accessibility and efficiency in grant searching, allowing users to obtain concise and contextually correct answers based on public data available on the web.  In the current context, access to information on aid and subsidies can be complicated due to the large amount of data and the way this information is consumed. To address this challenge, the system combines web scraping with RAG techniques providing answers based on both the retrieved content and the capabilities of the LLMs.  We want to propose the creation of a kind of virtual "assistant" that can answer specific questions about the topic discussed.  The aim is to implement an application that, using these technologies, can (1) extract data through web scraping from grant pages, (2) store this information, (3) use an open LLM model to generate responses from this information, and (4) evaluate the performance of the system in terms of accuracy, relevance and usefulness of the responses.  The results of the project must demonstrate that the combination of RAG and Open LLMs significantly improves the user experience by offering accurate and fast responses on the available aids.  Conclusions should highlight the potential of this technology for applications in areas with large volumes of public data and its applicability to private data. | |

Índice

[1. Introducción 6](#_Toc184231485)

[1.1. Descripción de la propuesta, justificación del interés y la relevancia de la propuesta 6](#_Toc184231486)

[1.2. Explicación de la motivación personal 8](#_Toc184231487)

[1.3. Definición de los objetivos (principales y secundarios) 8](#_Toc184231488)

[1.4. Impacto en sostenibilidad, ético-social y de diversidad 9](#_Toc184231489)

[1.5. Descripción de la metodología empleada en el desarrollo del proyecto 11](#_Toc184231490)

[1.6. Planificación y plan de investigación del proyecto 12](#_Toc184231491)

[1.7. Breve descripción de los capítulos de la memoria 15](#_Toc184231492)

[2. Estado del Arte 15](#_Toc184231493)

[2.1. Contexto 15](#_Toc184231494)

[2.2. Recuperación de Información 16](#_Toc184231495)

[2.2.1. Web scraping 18](#_Toc184231496)

[2.2.1.1. Introducción 18](#_Toc184231497)

[2.2.1.2. Consideraciones Éticas y Legales 21](#_Toc184231498)

[2.3. Procesamiento de Lenguaje Natural (NLP) y LLMs 21](#_Toc184231499)

[2.3.1. Introducción al NLP y evolución hasta los LLMs 21](#_Toc184231500)

[2.3.1.1. Contexto general del NLP 21](#_Toc184231501)

[2.3.1.2. Evolución hasta los LLMs 22](#_Toc184231502)

[2.3.2. Representaciones vectoriales (Embeddings) 23](#_Toc184231503)

[2.3.2.1. Contexto 23](#_Toc184231504)

[2.3.2.2. Definición 24](#_Toc184231505)

[2.3.2.3. Técnicas de Generación de Embeddings 25](#_Toc184231506)

[2.3.3. Almacenamiento de la información 27](#_Toc184231507)

[2.3.4. Bases de datos vectoriales 28](#_Toc184231508)

[2.3.4.1. Introducción 28](#_Toc184231509)

[2.3.4.2. Comparativa entre Bases de Datos Vectoriales 29](#_Toc184231510)

[2.3.5. Grandes Modelos de Lenguaje (LLMs) 31](#_Toc184231511)

[2.3.5.1. Introducción y transformers 31](#_Toc184231512)

[2.3.5.2. Transformers (entrar en detalle si no lo explicaste previamente) 31](#_Toc184231513)

[2.3.5.3. Modelos históricos: GPT y BERT (si no lo explicaste previamente) 32](#_Toc184231514)

[2.3.5.4. Comparativa entre modelos 34](#_Toc184231515)

[2.3.5.5. Avances recientes: capacidades multimodales 36](#_Toc184231516)

[2.3.5.6. Importancia y aplicaciones 37](#_Toc184231517)

[2.3.6. Técnicas avanzadas para optimizar LLMs 37](#_Toc184231518)

[2.3.6.1. Fine-tuning: personalización de modelos para tareas específicas. 37](#_Toc184231519)

[2.3.6.2. RAG: integración con recuperación de información para respuestas fundamentadas. 38](#_Toc184231520)

[2.3.6.2.1. Introducción 38](#_Toc184231521)

[2.3.6.2.2. Ventajas 40](#_Toc184231522)

[2.3.6.3. Prompt Engineering 40](#_Toc184231523)

[2.3.7. Evaluación de modelos 41](#_Toc184231524)

[2.3.7.1. Introducción 41](#_Toc184231525)

[2.3.7.2. Retrieval-Augmented Generation Analytics and Scoring (RAGAS) 42](#_Toc184231526)

[2.4. Desafíos y Consideraciones Éticas 42](#_Toc184231527)

[2.4.1. Centralizar los aspectos técnicos, éticos y sostenibles. 42](#_Toc184231528)

[2.5. Resumen y aplicabilidad al trabajo actual 44](#_Toc184231529)

[3. Materiales y métodos 45](#_Toc184231530)

[3.1. Descripción general del enfoque 45](#_Toc184231531)

[3.2. Fuentes de datos y Web Scrapping 46](#_Toc184231532)

[3.2.1. Obtención de los datos mediante el API REST 49](#_Toc184231533)

[3.2.2. Estructura y características de los datos 53](#_Toc184231534)

[3.2.3. Consideraciones técnicas 53](#_Toc184231535)

[3.3. Preprocesamiento de los datos y generación de Embeddings 53](#_Toc184231536)

[3.3.1. Limpieza y estructuración de los datos 54](#_Toc184231537)

[3.3.2. Uso de JSON para los metadatos 54](#_Toc184231538)

[3.3.3. Conversión a formato YAML 54](#_Toc184231539)

[3.3.4. Segmentación y enriquecimiento del texto 55](#_Toc184231540)

[3.3.5. Almacenamiento y organización 55](#_Toc184231541)

[3.4. Almacenamiento y búsqueda de datos 55](#_Toc184231542)

[3.5. Modelos y tecnologías utilizadas 55](#_Toc184231543)

[3.5.1. Modelos y tecnologías utilizadas 56](#_Toc184231544)

[3.5.1.1. Modelo de lenguaje: Llama 3.2:1b 56](#_Toc184231545)

[3.5.2. Base de datos vectorial: ChromaDB 56](#_Toc184231546)

[3.5.3. Integración mediante Ollama 57](#_Toc184231547)

[3.6. Interfaz de usuario 57](#_Toc184231548)

[3.6.1. Objetivos de la interfaz 57](#_Toc184231549)

[3.6.2. Herramientas y tecnologías utilizadas 58](#_Toc184231550)

[3.6.3. Diseño de la interfaz 58](#_Toc184231551)

[3.6.4. Flujo de interacción 58](#_Toc184231552)

[3.6.5. Ventajas del diseño basado en Streamlit 59](#_Toc184231553)

[3.7. Evaluación del modelo con RAGAS 59](#_Toc184231554)

[4. Resultados y resumen de la evaluación 60](#_Toc184231555)

[4.1. Resultados 60](#_Toc184231556)

[4.2. Resumen de la evaluación del modelo 60](#_Toc184231557)

[5. Conclusiones y trabajos futuros (22/12) 60](#_Toc184231558)

[6. Glosario 61](#_Toc184231559)

[7. Bibliografía 62](#_Toc184231560)

[8. Anexos 65](#_Toc184231561)

[8.1. Planificación detallada 65](#_Toc184231562)

[8.2. Embeddings 69](#_Toc184231563)

[8.3. Retrieval-Augmented Generation Analytics and Scoring (RAGAS) 73](#_Toc184231564)

Lista de Figuras

Figura 1: Diagrama Gantt Resumido 13

Figura 2: Temáticas Presente Trabajo 15

Figura 3: Etapas Web Scraping 19

Figura 4: Diagrama Embeddings [22] 21

Figura 5: Diagrama Generación Embeddings [22] 22

Figura 6: Diagrama Embeddings WORD2VEC [22] 23

Figura 7: Diagrama Embeddings BERT WORD2VEC [22] 24

Figura 8: Comparativa BD Vectoriales desde 2 perspectivas [25] 32

Figura 9: Capas Transformers [24] 34

Figura 10: Evolución técnicas PLN 35

Figura 11: Diagrama LLMs desde distintas variables [25] 37

Figura 12: Diagrama RAG Architecture Model. [38] 41

Figura 13: Árbol Métricas RAGAS [7] 45

Figura 14: Diagrama de Gantt (parte 1) 58

Figura 15: Diagrama de Gantt (parte 2) 59

Lista de Tablas

Tabla 1: Recursos Hardware 11

Tabla 2: Recursos Software 11

Tabla 3: Comparativa de bases de datos vectoriales 31

Tabla 4: Clasificación Métricas RAGAS 46

# Introducción

## Descripción de la propuesta, justificación del interés y la relevancia de la propuesta

El punto de partida de este proyecto es la creciente complejidad y volumen de información relacionada con las ayudas y subvenciones públicas en España. En la actualidad, tanto ciudadanos como empresas que buscan acceder a estas ayudas enfrentan dificultades para encontrar información relevante de manera eficiente. Esta situación genera una barrera de entrada para muchos potenciales beneficiarios, quienes a menudo carecen de tiempo, conocimientos o recursos para efectuar búsquedas exhaustivas.

El problema se agrava por la forma en que se presenta la información: aunque está disponible, los datos no siempre son fácilmente comprensibles o accesibles para todos los usuarios. Además, el proceso de búsqueda manual no es solo ineficiente, sino que también puede llevar a la pérdida de oportunidades debido a la falta de visibilidad de algunas subvenciones o de información contenida en documentos que no está estructurada ni permite su búsqueda. Este es un tema fundamental porque las ayudas y subvenciones públicas son herramientas clave para impulsar el desarrollo económico y social, y su accesibilidad es fundamental para garantizar una distribución equitativa de los recursos públicos.

Al inicio de este trabajo, las soluciones disponibles se limitaban a sistemas de búsqueda tradicionales o portales con filtrado básico. No existe un sistema capaz de integrar grandes volúmenes de información, procesarla y proporcionar respuestas precisas y personalizadas a las consultas de los usuarios de una manera simple.

La aportación principal de este proyecto es la implementación de un sistema basado en RAG (Retrieval-Augmented Generation) y Open LLM (Open Large Languaje Model) que permita la extracción automatizada, almacenamiento y búsqueda de información relevante sobre ayudas y subvenciones.

Existen muchas definiciones de lo que es un modelo de lenguaje grande (LLM, por sus siglas en inglés). Una primera aproximación la encontramos en Wikipedia [1], que describe un LLM como un modelo de lenguaje basado en una red neuronal con un número muy alto de parámetros (generalmente miles de millones o más), entrenado con enormes volúmenes de texto sin etiquetar mediante aprendizaje autosupervisado o semisupervisado. De forma más sencilla, podríamos decir que un LLM es una inteligencia artificial entrenada para comprender y generar texto de manera natural. Este modelo aprende de un conjunto finito de enormes cantidades de texto, como libros, artículos y páginas web, y es utilizado para responder preguntas, crear resúmenes o mantener una conversación coherente.

Estos modelos de lenguaje presentan diversas limitaciones, siendo una de las más importantes la cantidad limitada de información que pueden almacenar. Esto los hace útiles para responder preguntas relacionadas con el conocimiento incluido en el modelo; sin embargo, ¿qué ocurre si deseamos obtener información que no ha sido incorporada en su entrenamiento? En estos casos, existen diferentes enfoques, entre ellos los siguientes:

* Fine-tuning: consiste en reentrenar el modelo original con la información específica que necesitamos consultar, aunque este enfoque implica un alto costo en términos de tiempo y recursos computacionales.
* Bases de datos de conocimiento: se basa en el uso de bases de datos semánticas, grafos de conocimiento o sistemas de preguntas y respuestas para organizar y acceder a la información.
* Prompts personalizados: afinando las consultas mediante prompts estructurados, lo cual facilita la recuperación de información específica a través de un modelo de lenguaje.
* Retrieval-Augmented Generation (RAG): una técnica en la que se recupera información externa, no almacenada en el modelo, para proporcionarle al modelo un contexto adicional que le permita generar respuestas más precisas y relevantes.

En este trabajo nos centraremos en la técnica de Retrieval-Augmented Generation (RAG). Para comenzar, es importante entender qué es RAG; en Wikipedia [5], se define como un tipo de inteligencia artificial generativa que incorpora capacidades de recuperación de información. Esta técnica combina dos conceptos clave: en primer lugar, la obtención de información actualizada a partir de fuentes como documentos, sitios web, bases de datos, entre otros, que se almacenan en bases de datos vectoriales o grafos de conocimiento; en segundo lugar, el uso de un modelo de lenguaje grande (LLM) que se alimenta de esta información específica, permitiendo así generar respuestas más precisas y actualizadas.

La propuesta de este trabajo se basa en el uso de técnicas de Retrieval-Augmented Generation (RAG) mediante modelos de lenguaje grandes (LLMs) de código abierto, los cuales pueden implementarse en infraestructuras on-premise. Esta configuración permite garantizar la privacidad de los datos, una característica esencial para muchas empresas que prefieren no utilizar LLMs como servicio, evitando así que la información del modelo se almacene en sistemas de terceros.

## Explicación de la motivación personal

Mi motivación para desarrollar este Trabajo de Fin de Máster surge de mi experiencia profesional como ingeniero de software y manager, especializado en la creación y gestión de intranets, portales documentales y plataformas colaborativas. A lo largo de mi carrera, he trabajado con grandes volúmenes de información interna en entornos empresariales, enfrentándome a uno de los principales desafíos: asegurar que los usuarios accedan a la información correcta de manera rápida y eficiente.

A lo largo de los años, he utilizado soluciones de terceros Out of the Box (OOB), como Google Mini, para implementar búsquedas en intranets corporativas, en su momento ofrecidas por Google. También he empleado productos de terceros en bases de datos de conocimiento y asistentes virtuales para optimizar la recuperación de información y mejorar la precisión en la resolución de consultas.

Este proyecto me ofrece la oportunidad de explorar y probar la técnica de Retrieval-Augmented Generation (RAG) combinada con modelos de lenguaje abiertos (LLMs) en el contexto de acceso a información pública, como son las ayudas y subvenciones del Gobierno de España. En el futuro, el objetivo es que estas técnicas se conviertan en una solución altamente efectiva para mejorar las búsquedas en intranets corporativas y sistemas documentales internos de las empresas.

Uno de los principales impulsores de este trabajo es la posibilidad de aplicar, en el futuro, los conocimientos adquiridos a entornos empresariales internos. Mi objetivo a largo plazo es implementar esta tecnología en portales documentales internos, donde la dispersión de información y la falta de herramientas avanzadas de búsqueda son problemáticas comunes. Considero que un sistema basado en Retrieval-Augmented Generation (RAG) y modelos de lenguaje (LLMs) tiene el potencial de transformar la forma en que se accede y gestiona la información dentro de las organizaciones, proporcionando una ventaja competitiva en términos de eficiencia operativa y en la toma de decisiones.

## Definición de los objetivos (principales y secundarios)

El presente Trabajo de Fin de Máster (TFM) tiene como objetivo general desarrollar un sistema que, mediante el uso de técnicas de Retrieval-Augmented Generation (RAG) y modelos de lenguaje abiertos (Open LLMs), facilite la extracción automatizada de información relevante del Sistema Nacional de Ayudas y Subvenciones Públicas del Gobierno de España. Este objetivo se desglosa en los siguientes objetivos principales y secundarios:

* Extracción de datos estructurados y no estructurados de webs
* Identificar y seleccionar fuentes de información relevantes en el Sistema Nacional de Ayudas y Subvenciones Públicas para extraer datos clave mediante técnicas de web scraping.
* Desarrollar un sistema de extracción automática de información, utilizando web scraping para capturar datos de ayudas y subvenciones públicas de forma estructurada y no estructurada.
* Indexación de información extraída
* Implementar un modelo RAG que integre la recuperación de documentos con la generación de texto, optimizando las respuestas generadas basadas en los datos recuperados.
* Integrar la solución con un modelo de lenguaje abierto (Open LLM) con el fin de que sea capaz de procesar y generar respuestas personalizadas y contextualizadas a las consultas de los usuarios sobre ayudas y subvenciones.
* Optimizar el sistema para diferentes tipos de consultas, garantizando que las respuestas proporcionadas sean claras, comprensibles y accesibles para un amplio espectro de usuarios, desde ciudadanos hasta empresas.
* Evaluar la precisión, relevancia y utilidad del sistema a través de pruebas con usuarios y métricas de rendimiento, comparando los resultados con los métodos tradicionales de búsqueda de información. Aquí utilizaremos el framework RAGAS [7] que nos ayuda a esta evaluación.
* Implementación de una interfaz web que, a través de un chatbot, proporcione una experiencia de uso accesible para la consulta y consumo de esta información.
* Documentación de las metodologías, procesos y resultados obtenidos a lo largo del desarrollo del sistema, incluyendo un análisis crítico sobre las ventajas y limitaciones de la solución propuesta.

## Impacto en sostenibilidad, ético-social y de diversidad

Este trabajo, centrado en el contexto de ayudas públicas, tiene como objetivo facilitar un acceso más equitativo a la información para todos los ciudadanos. Buscamos lograr un equilibrio entre eficiencia, sostenibilidad y ética, incorporando principios de transparencia, optimización energética y diversidad en el diseño.

En lo que respecta a la sostenibilidad, es fundamental optimizar el uso de datos y controlar el consumo computacional para mitigar el impacto ambiental. Se propone la implementación de modelos ligeros ejecutados localmente, lo que nos permitirá optimizar los costes energéticos y evitar la dependencia de grandes infraestructuras en la nube.

Por otro lado, el impacto ético-social del uso de información pública es sumamente relevante, ya que afecta directamente a los ciudadanos y a su bienestar económico. En este sentido, existen varios aspectos clave que deben considerarse en el análisis ético-social de estos sistemas:

* Privacidad y uso responsable de datos: Es fundamental respetar la privacidad de los usuarios y cumplir con normativas de protección de datos, como la Ley Orgánica de Protección de Datos (LOPD).
* Prevención de sesgos: Es crucial que los modelos de lenguaje grande (LLM) se desarrollen y ajusten para minimizar los sesgos en las respuestas generadas.
* Inclusión de mecanismos de retroalimentación: Es necesario ofrecer a los usuarios la capacidad de proporcionar retroalimentación, lo que permitirá que el sistema mejore con el tiempo y se adapte mejor a las necesidades de diferentes grupos poblacionales. Esto fomentará una relación más ética y dinámica entre los ciudadanos y la tecnología

Otro aspecto importante de este trabajo es la diversidad, que debe garantizar la satisfacción de las necesidades de todos los grupos de la sociedad. Para ello, es fundamental considerar los siguientes principios:

* Datos representativos: Para que sean útiles, los datos deben reflejar la totalidad de los sectores de la sociedad. Esto implica integrar información sobre ayudas para todos los colectivos, por muy minoritarios que sean, así como para diferentes niveles socioeconómicos y personas con capacidades diversas.
* Pluralidad cultural y lingüística: El sistema debe ser capaz de comprender y adaptarse a las diversas formas en que las personas expresan sus necesidades. En este contexto, utilizaremos un lenguaje común que abarque a toda la sociedad española, dado que los modelos de lenguaje actuales no logran representar la diversidad lingüística existente.
* Accesibilidad: La aplicación debe ser accesible para personas con discapacidad o limitaciones tecnológicas, especialmente en el caso de personas de la tercera edad.
* Lenguaje inclusivo y adaptado a la audiencia: Las respuestas generadas deben ser inclusivas y adecuarse a personas con diferentes niveles de educación, de modo que sean comprensibles para todos.

## Descripción de la metodología empleada en el desarrollo del proyecto

Para abordar el desarrollo del proyecto se opta por utilizar la metodología Waterfall separando el desarrollo de 4 módulos diferentes:

* Desarrollo del módulo web scraping.
* Desarrollo almacenamiento en bases de datos vectoriales: Identificación de una base de datos adecuada.
* Desarrollo e integración con un Open LLM. Identificación de un Open LLM que sirva para el propósito de este proyecto.
* Desarrollo de aplicación web.
* Evaluación del modelo mediante RAGAS.

Estos cinco módulos se implementarán inicialmente con los aspectos básicos, con el objetivo de desarrollar un MVP (Minimum Viable Product) que cumpla con los requisitos mínimos. En iteraciones sucesivas durante la fase de desarrollo, se mejorará y perfeccionará cada uno de los módulos con el fin de alcanzar el producto final.

Se puede afirmar que se sigue una metodología Waterfall con refinamiento, ya que se establecen fases que tienen como objetivo principal obtener un MVP, y en las fases posteriores se busca afinar cada vez más la entrega. Los objetivos secundarios que no puedan ser abordados en este trabajo se dejarán como tareas futuras.

## Planificación y plan de investigación del proyecto

**Recursos necesarios**

Para la realización y materialización del proyecto se han empleado los siguientes elementos:

|  |  |
| --- | --- |
| **Hardware** |  |
| **PC sobremesa** | Intel(R) Core(TM) i5-9600K CPU @ 3.70GHz |
| **Tarjeta gráfica** | NVIDIA GeForce RTX 4070 SUPER |
| **Memoria** | 16GB RAM |

Tabla 1: Recursos Hardware

|  |  |
| --- | --- |
| **Software** |  |
| **Sistema Operativo** | Windows 11 Pro 23H2 |
| **Gestor extensiones Python** | Anaconda |
| **Editor de desarrollo** | Visual Studio Code |
| **Extensiones VS Code** | Python  Python Debugger  Jupyter |
| **Python** | Python 3.9.20 |
| **Princiales Librerías Python** | LangChain, OpenAI, faiss-gpu, chromadb, SentenceTransformer, BeautifulSoup, Request, torch con soporte de CUDA |

Tabla 2: Recursos Software

* + 2. **Tareas y Planificación**

A continuación, se presenta un resumen detallado de las fases, las principales tareas y los hitos del proyecto, junto con un diagrama de Gantt también resumido. Las entregas de documentación en el campus están identificadas como hitos clave.

**FASE DE PLANIFICACIÓN**

Definición del tema, objetivos y análisis de herramientas

- **HITO**: Definición del TFM: enunciado y entrega (M1)

- **HITO**: Entrega comité ética y convenios (M1)

- **HITO**: Estado del arte: enunciado y entrega de la actividad (M2)

**ANÁLISIS DE REQUISITOS**

Revisión de requisitos y especificación de casos de uso

**DISEÑO DEL SISTEMA**

Diseño detallado de cada uno de los módulos del sistema

**DESARROLLO DEL PROYECTO**

Desarrollo y pruebas unitarias

**FASE DE PRUEBAS**

Pruebas de integración y UAT.

**DOCUMENTACIÓN**

Documentación del proyecto

**REVISIÓN FINAL DEL TFM**

Redacción del documento final del TFM, revisión y corrección

- **HITO**: Redacción de la memoria: entrega preliminar (M4)

- **HITO**: Redacción de la memoria: entrega final (M4)

Preparación del video TFM

- **HITO**: Presentación audiovisual del trabajo (M4)

**DEFENSA DEL TFM**

Preparación de la presentación y ensayos de defensa.

- **HITO**: Presentación ante el tribunal

**Planificación resumida - Diagrama de Gantt**

**Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente**

Figura 1: Diagrama Gantt Resumido

En el Anexo 12.1 se presenta un listado detallado de las fases y tareas, así como el diagrama de Gantt completo.

## Breve descripción de los capítulos de la memoria

# Estado del Arte

**Figuras**

Siempre referenciar, contextualizar y explicar en el texto todas las figuras que incluyas.

En vez de:

A continuación, se puede visualizar

Mejor poner:

A continuación, en la Fig. 2 se puede visualizar

*Figura 2 -> Esta figura creo que no aporta demasiado. En todo caso, mejoraría la explicación del pie de figura*.

**Referencias bibliográficas**

Asegúrate de cumplir una citación bibliográfica estándar. En la sección Recursos de

aprendizaje (M1) del Aula Virtual tienes los detalles:

En [14] (Thakur et al., 2021) presentan BEIR, → Creo que estás haciendo una combinación

poco común de estilos (para referenciar) que no sigue estrictamente ningún estándar conocido.

Creo que en ocasiones no seguiste un orden creciente en las referencias (la 1ª vez que las

introduces). Veo saltos, por ejemplo, de la 11 pasas a la 27.

“Posteriormente en [30] (Ferrara et al., 2021) proporciona un análisis exhaustivo” → Creo que

aquí la referencia debe ser la [31]

## Contexto

La adopción de modelos de lenguaje de gran tamaño (Large Language Models, LLMs) ha transformado notablemente la manera en que las tecnologías de procesamiento de lenguaje natural (PLN) interactúan con el usuario y extraen información de fuentes diversas. Estos modelos han demostrado una habilidad sin precedentes para comprender, generar y sintetizar texto humano, permitiendo el desarrollo de aplicaciones avanzadas como asistentes virtuales, sistemas de recomendación o chatbots, entre otros. Sin embargo, hay un largo recorrido en lo que respecta al uso de LLMs en ámbitos de información privada y en dominios específicos, como es el caso de uso que se presenta aplicar en el presente trabajo centrado en la extracción y consulta de información de organismos públicos.

En este contexto, este trabajo se centra en la aplicación de técnicas de recuperación aumentada de generación (RAG) y el uso de modelos de lenguaje de código abierto para crear un sistema de asistencia conversacional que facilite la búsqueda de información relevante sobre subvenciones y ayudas públicas del Gobierno de España.

La metodología para la revisión bibliográfica ha consistido en hacer una búsqueda exhaustiva de publicaciones científicas que nos permitan conocer la evolución de estas tecnologías de inteligencia artificial y procesamiento del lenguaje natural (PLN) a lo largo del tiempo, así como conocer su estado de la actualidad. Se han seleccionado temáticas y tecnologías clave sobre la recuperación de información (mediante técnicas de web scraping), codificación y procesamiento de la información (mediante técnicas de embedding), almacenamiento y consulta de esta información (mediante bases de datos vectoriales) y generación de respuestas de esta (mediante grandes modelos del lenguaje (LLMs). A continuación, se puede visualizar de una forma visual y esquemática la estructuración de estas temáticas y tecnologías que ayudarán a entender la estructuración de los siguientes temas, contribuyendo así a fundamentar el desarrollo de este proyecto.

Figura 2: Temáticas Presente Trabajo

## Recuperación de Información

La recuperación de información (RI, por sus siglas en inglés de Information Retrieval) constituye un área de investigación centrada en el diseño y desarrollo de sistemas destinados a buscar y extraer información desde diversos orígenes de datos, abarcando tanto información estructurada como no estructurada.

Se entiende por información estructurada aquella que permite la identificación de datos específicos, al estar almacenada en bases de datos, webs o en documentos organizados de manera clara y precisa, facilitando así su recuperación y análisis. En contraste, la información no estructurada procede de fuentes sin una estructura definida, tales como registros de eventos (logs) o documentos sin un formato organizado o etiquetas explícitas. La gestión y análisis de este tipo de datos presentan desafíos adicionales en comparación con la información estructurada, dada la ausencia de un orden preestablecido que permita la identificación directa de datos específicos. Los sistemas de recuperación de información intentan satisfacer las necesidades informativas del usuario, procesando esta información y almacenándola de una forma persistente para su posterior explotación.

El campo de la recuperación de la información ha evolucionado desde los primeros modelos basados en vectores (Salton, Wong, & Yang, 1975) [8] los autores nos presentan una representación matemática de documentos y consultas como vectores en un espacio multidimensional, donde cada dimensión corresponde a un término. La relevancia entre un documento y una consulta se mide mediante la similitud del coseno entre sus vectores, lo que permite la recuperación basada en términos ponderados. Este modelo sentó las bases para métodos de búsqueda eficientes y sigue siendo un pilar en los sistemas modernos de recuperación.

Posteriormente (Baeza-Yates & Ribeiro-Neto, 1999) en [10] se abordan tanto los fundamentos como los avances modernos en esta disciplina. En este libro los autores exploran técnicas de búsqueda y clasificación, así como cuestiones sobre escalabilidad y rendimiento en grandes volúmenes de datos. También se abordan otros temas importantes como el modelado de documentos, la interfaz de usuario y la evaluación de sistemas de recuperación, dándonos una visión integral como base para investigadores y profesionales del campo.

En este otro artículo [10] (Singhal, 2001) ofrece una visión general de la evolución y los desafíos de la recuperación de información en la era digital. Plantea una transición hacia sistemas basados en el aprendizaje automático, al igual que (Manning, Raghavan, & Schütze, 2009) en [11]. En ambos artículos se realiza una introducción exhaustiva de los conceptos y métodos en la recuperación de información, desde los modelos clásicos hasta las técnicas avanzadas basadas en aprendizaje automático. En este segundo (Manning, Raghavan, & Schütze, 2009) en [11], Manning, Raghavan y Schütze cubren temas como el procesamiento de lenguaje natural, la clasificación y el clustering de documentos, la indexación y los motores de búsqueda. Esta obra tiene un enfoque más pedagógico y es usada como recurso formativo en el campo de la recuperación de información.

Todos estos trabajos contribuyeron a la consolidación de la recuperación de información, abordando sus fundamentos y desafíos, y estableciendo metodologías que hoy en día aún siguen siendo relevantes.

# Web scraping

## Introducción

El web scraping es una técnica que permite obtener información de sitios web de manera automatizada a través de programas especializados que recorren y procesan el contenido de las páginas. El web scraping es ampliamente utilizado en aplicaciones como la recolección de precios de productos en e-commerce, la monitorización de tendencias en redes sociales o en ámbitos financieros como revisar tendencias y establecer avisos en la bolsa.

Esta técnica ha cobrado gran relevancia en los últimos años debido a la ingente cantidad de información existente en la web que no está disponible a través de interfaces de programación de aplicaciones (API). Las APIs son interfaces estándar que, mediante protocolos específicos, permiten acceder de forma segura y controlada a los datos que los sitios deciden exponer. Sin embargo, cuando no existe una API o cuando la información requerida no está accesible mediante este tipo de interfaces, el web scraping se convierte en una alternativa para obtener datos tanto estructurados como no estructurados desde múltiples fuentes.

Esta información extraída puede luego integrarse en sistemas de análisis de datos y explotación de información, permitiendo procesarla de forma centralizada. Esto resulta especialmente valioso en el desarrollo de aplicaciones que requieren grandes volúmenes de datos, como los modelos de inteligencia artificial, entre ellos los modelos de lenguaje (LLMs) y otros algoritmos de aprendizaje automático.

Si miramos la bibliografía vemos que uno de los primeros trabajos en explorar y clasificar las herramientas de extracción de datos web es [27] realizado por (Laender et al., 2002). Este artículo destaca las dificultades técnicas de extraer datos de la web debido a la heterogeneidad de las estructuras HTML y al hecho de que la información en la web no está diseñada para ser extraída en masa. Los autores clasifican los métodos en dos enfoques: los basados en reglas, conocidos como wrappers, y los métodos basados en aprendizaje automático. La principal limitación identificada para los wrappers es su alta dependencia de la estructura de las páginas, lo que requiere constantes actualizaciones cuando los sitios web cambian. Los métodos de aprendizaje automático ofrecen más adaptabilidad, aunque son limitados por la necesidad de grandes cantidades de datos etiquetados y la dificultad para generalizar entre diferentes sitios web. Laender et al. subrayan la necesidad de métodos que equilibren la precisión con la capacidad de adaptación y mantenimiento, una cuestión que sigue siendo relevante a medida que la web se expande y se vuelve más compleja.

En [28] (Arasu & Garcia-Molina, 2003) proponen el sistema DEPTA (Data Extraction and Processing of Tree-like Attributes) para extraer datos estructurados de páginas web no estructuradas, especialmente aquellas que presentan información en forma de listas o tablas. DEPTA utiliza el modelo DOM para representar la estructura de la página como un árbol y localizar patrones repetitivos en el HTML, identificando subsecciones que contienen datos relevantes y organizándolas en formato tabular. Este enfoque permite una extracción de datos más precisa y automatizada, adaptándose a variaciones en el diseño de las páginas sin requerir reglas manuales específicas para cada sitio, lo cual representa un avance significativo en la extracción de datos web.

Un análisis más amplio es presentado por (Chang et al., 2006) en [29] donde clasifica las técnicas de extracción en tres categorías: sistemas basados en wrappers, métodos de aprendizaje automático, y enfoques híbridos. Los sistemas basados en wrappers siguen siendo útiles para sitios con estructuras estables, mientras que el aprendizaje automático permite la generalización en múltiples sitios, aunque a costa de una mayor complejidad y requerimientos de datos. Chang et al. reconocen la importancia de los sistemas híbridos, que integran reglas específicas con aprendizaje automático para lograr mayor precisión y flexibilidad. Al explorar los desafíos, este estudio destaca las limitaciones de escalabilidad y el alto costo de mantenimiento de los extractores, aspectos que motivan la búsqueda de sistemas más autónomos y adaptativos.

Por su parte, (Ferrara et al., 2014) en [30] amplía la perspectiva al discutir tanto las técnicas de extracción como sus aplicaciones. Los autores identifican tres tipos de técnicas: wrappers manuales y semiautomáticos, métodos basados en aprendizaje automático y procesamiento del lenguaje natural (NLP), y métodos híbridos avanzados. Este artículo profundiza en el uso de técnicas avanzadas como las redes neuronales para mejorar la precisión y adaptabilidad de los sistemas de extracción, anticipando la creciente influencia de la inteligencia artificial en este campo. Los autores también analizan aplicaciones prácticas, como la minería de opiniones y la vigilancia de mercados, y subrayan los desafíos éticos y legales, como el respeto por la privacidad y los derechos de propiedad de los datos. Este enfoque centrado en aplicaciones es relevante para el desarrollo de sistemas de extracción que puedan integrar y analizar grandes volúmenes de datos de múltiples fuentes, algo esencial en sectores como el comercio, la investigación y la analítica empresarial.

Posteriormente en [30] (Ferrara et al., 2021) proporciona un análisis exhaustivo de las técnicas y enfoques utilizados en el raspado y la exploración web. En este trabajo, se diferencia entre el web scraping y el web crawling, destacando que el primero se centra en la extracción de datos específicos de páginas web, mientras que el segundo se ocupa de la indexación y el almacenamiento de datos para facilitar búsquedas eficientes. El autor revisa diversas técnicas de scraping, incluyendo métodos basados en patrones y heurísticas, y analiza su aplicabilidad en diferentes contextos, como la recopilación de datos para análisis de mercado y la investigación académica.

Concluyendo el web scraping implica básicamente la automatización de la consulta de diferentes páginas web para obtener información. Esto se puede hacer a muy bajo nivel mediante una petición HTTP (poco usado en la actualidad) o mediante el uso de bibliotecas de programación específicas, como pueden ser BeautifulSoup y Requests, Selenium o Scrapy, que permiten acceder al código HTML de las páginas web y extraer datos relevantes mediante técnicas de procesamiento de texto y manipulación de estructuras de datos. La arquitectura de un sistema de web scraping generalmente incluye las siguientes etapas:

Figura 3: Etapas Web Scraping

* Identificación de fuentes: Esta etapa implica la revisión de diferentes orígenes de datos web susceptibles de tener la información que se necesita.
* Extracción de HTML: Esta etapa implica la conexión a una URL específica y la recuperación del código HTML de la página, utilizando protocolos como HTTP y HTTPS. Herramientas como Requests en Python facilitan la comunicación con los servidores web y la descarga de contenido.
* Análisis de Estructura (Parsing): Una vez obtenido el HTML, se emplean técnicas de análisis para estructurar la información. BeautifulSoup, por ejemplo, permite navegar en la estructura de nodos del documento HTML para identificar y extraer etiquetas y atributos específicos, como `<div>`, `<span>`, y `<a>`.
* Limpieza de datos: Una vez obtenidos los datos es necesario analizarlos y realizar una limpieza de estos.
* Almacenamiento de Datos: Los datos extraídos pueden almacenarse en formatos estructurados como JSON, CSV o en bases de datos SQL y NoSQL, facilitando su uso en posteriores análisis.

Además, estas fases se deben realizar de forma cíclica ya que los orígenes de datos y la estructura de estos están en constante evolución.

## Consideraciones Éticas y Legales

El web scraping plantea desafíos legales y éticos, ya que algunos sitios web prohíben explícitamente la extracción de datos sin consentimiento, y en otros, los datos pueden estar protegidos por derechos de autor. Como fase previa al proceso de web scraping se debe realizar una revisión exhaustiva de los términos de uso de cada sitio web y, en caso de estar disponible, la utilización de interfaces de programación de aplicaciones (APIs) oficiales que nos garanticen un acceso seguro, fiable y autorizado a los datos.

# Procesamiento de Lenguaje Natural (NLP) y LLMs

# Introducción al NLP y evolución hasta los LLMs

# Contexto general del NLP

El Procesamiento del Lenguaje Natural (NLP, por sus siglas en inglés) es una disciplina que combina lingüística, informática e inteligencia artificial para permitir que las máquinas comprendan, procesen y generen lenguaje humano de forma significativa. Desde sus inicios, el NLP ha evolucionado a través de varias fases:

1. **Primeras aproximaciones basadas en reglas**: En los años 50 y 60, el enfoque predominante era basado en reglas gramaticales diseñadas manualmente. Ejemplos icónicos incluyen el modelo de gramáticas de Noam Chomsky y el programa ELIZA, un chatbot desarrollado en 1966 por Joseph Weizenbaum.
   * Referencia: Chomsky, N. (1956). *Three models for the description of language*. IRE Transactions on Information Theory.
2. **Modelos estadísticos y corpus lingüísticos**: Con el crecimiento de los datos digitales en los 80 y 90, surgieron técnicas estadísticas que explotaban grandes corpus textuales para modelar relaciones lingüísticas. Se popularizaron algoritmos como los modelos de Markov y los métodos de aprendizaje supervisado para tareas como el etiquetado POS (Part-of-Speech).
   * Referencia: Jurafsky, D., & Martin, J. H. (2000). *Speech and Language Processing*. Prentice Hall.
3. **Técnicas basadas en aprendizaje automático**: A partir de los 2000, el uso de algoritmos de aprendizaje automático (machine learning) permitió un avance significativo en el NLP, al introducir modelos capaces de generalizar patrones lingüísticos complejos a partir de datos anotados.
   * Referencia: Manning, C. D., & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.

# Evolución hasta los LLMs

La transición hacia los modelos de lenguaje de gran escala (LLMs, por sus siglas en inglés) se caracteriza por la adopción de redes neuronales profundas y el uso masivo de datos no supervisados.

1. **Redes neuronales recurrentes (RNNs) y modelos basados en secuencias**: Técnicas como las RNNs y LSTMs (Long Short-Term Memory) demostraron avances en tareas como traducción automática y generación de texto, aunque estaban limitadas por problemas como el *vanishing gradient*.
   * Referencia: Hochreiter, S., & Schmidhuber, J. (1997). *Long short-term memory*. Neural Computation.
2. **Transformers: una revolución estructural**: En 2017, el modelo Transformer introdujo un nuevo paradigma basado en la atención auto-regresiva, eliminando la dependencia de las secuencias recurrentes. Este modelo dio lugar a arquitecturas como BERT y GPT, que lograron avances significativos en tareas como comprensión de texto y generación de lenguaje.
   * Referencia: Vaswani, A., et al. (2017). *Attention is all you need*. Advances in Neural Information Processing Systems (NeurIPS).
3. **Escalamiento hacia los LLMs**: Modelos como GPT-3 y LLaMA (Large Language Model Meta AI) representan el estado del arte, entrenados en enormes cantidades de datos y con miles de millones de parámetros. Estos modelos son capaces de realizar tareas complejas en NLP con adaptaciones mínimas.
   * Referencia: Brown, T., et al. (2020). *Language models are few-shot learners*. Advances in Neural Information Processing Systems (NeurIPS).
   * Referencia: Touvron, H., et al. (2023). *LLaMA: Open and efficient foundation language models*. arXiv preprint.

# Representaciones vectoriales (Embeddings)

No me acaba de convencer el nombre de esta sección. Creo que es mejor algo más concreto, como por ejemplo: *Representaciones vectoriales (Embeddings)*

Haces un análisis muy exhaustivo de los Embeddings! Me parece interesante. Actualmente, los embeddings los podemos ver como un componente clave en el funcionamiento de los LLMs. No obstante, hay que tener en cuenta que hay una restricción en la extensión de la memoria final de los TFMs (revísalo en las explicaciones del aula virtual correspondiente a la redacción de la memoria):

***La memoria debe tener una extensión de entre 40 y 60 páginas de texto contando***

***desde la introducción hasta las líneas de trabajo futuras (ambos apartados incluidos).***

Recomiendo resumir el tema de los embeddings o mover parte a los anexos (que no

contabilizan).

Intenta también sintetizar/resumir otros apartados.

## Contexto

En el campo del procesamiento del lenguaje natural (NLP), el entendimiento y la generación de texto han avanzado considerablemente gracias a dos conceptos fundamentales: los embeddings y los grandes modelos del lenguaje (LLMs). Estos enfoques han permitido a los modelos de inteligencia artificial captar, de manera más precisa y contextual, el significado y las relaciones en el lenguaje humano, revolucionando así una amplia gama de aplicaciones, desde motores de búsqueda y chatbots hasta traducción automática y análisis de sentimientos.

Los embeddings son técnicas que permiten transformar palabras, frases o documentos en vectores numéricos que capturan sus relaciones semánticas en un espacio matemático. Estas representaciones capturan relaciones semánticas y pueden ser utilizadas en tareas de clasificación, análisis de sentimientos, búsqueda de información, entre otros. Por otro lado, los grandes modelos del lenguaje (LLMs) representan un avance significativo, ya que no solo generan embeddings, sino que también están diseñados para comprender, generar y manipular texto de manera contextual y coherente, permitiéndoles realizar tareas avanzadas de lenguaje, como generación de texto, traducción y respuesta a preguntas. Estos modelos, basados principalmente en la arquitectura de Transformers, cuentan con millones de parámetros que les permiten adaptarse a diversas tareas sin necesidad de ajustes complejos. Además, utilizan grandes volúmenes de datos para entrenarse, lo cual les permite aprender patrones lingüísticos y operar en una amplia variedad de aplicaciones sin requerir ajustes específicos para cada tarea.

En los próximos capítulos nos centraremos en explicar detalladamente cada uno de este tipo de modelos para justificar su utilización en el presente trabajo.

**….. Ver resumen y referenciar al Anexo.**

## Definición

Los embeddings son representaciones vectoriales de datos en un espacio continuo de múltiples dimensiones. Estas representaciones permiten que datos como texto, imágenes o audio puedan ser representados en vectores, capturando relaciones semánticas y contextuales.

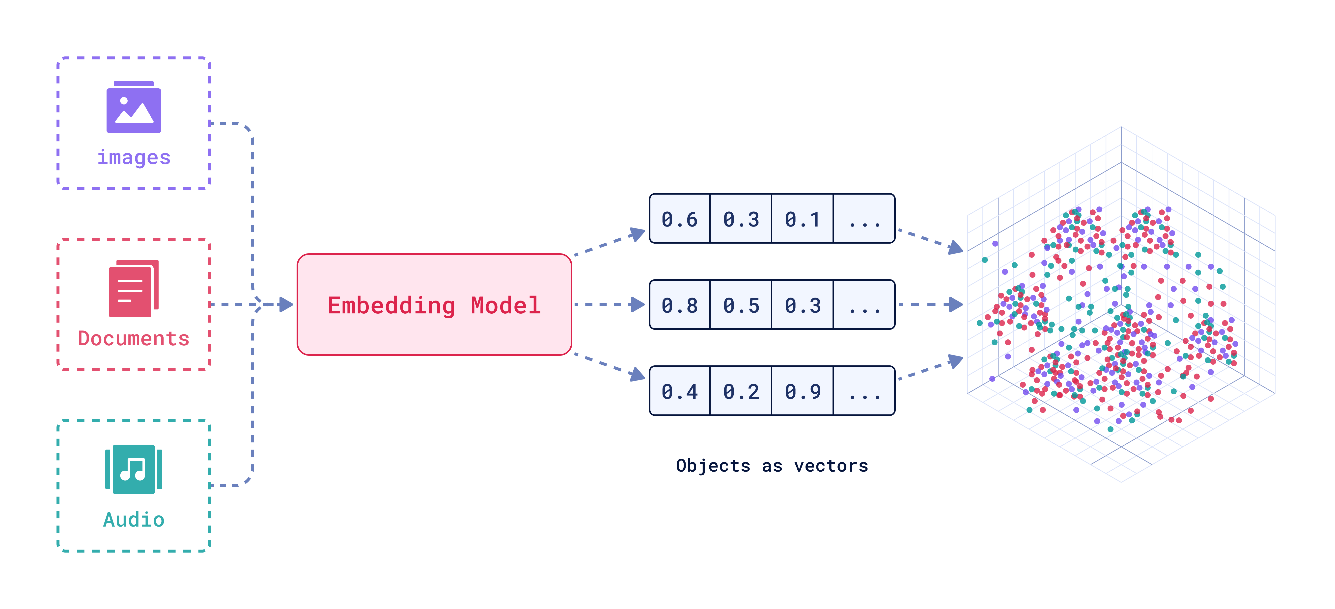


Figura 4: Diagrama Embeddings

En el contexto de los modelos de lenguaje, los embeddings permiten que los modelos comprendan y procesen texto al mapear palabras o frases en espacios vectoriales donde la proximidad refleja la similitud semántica.

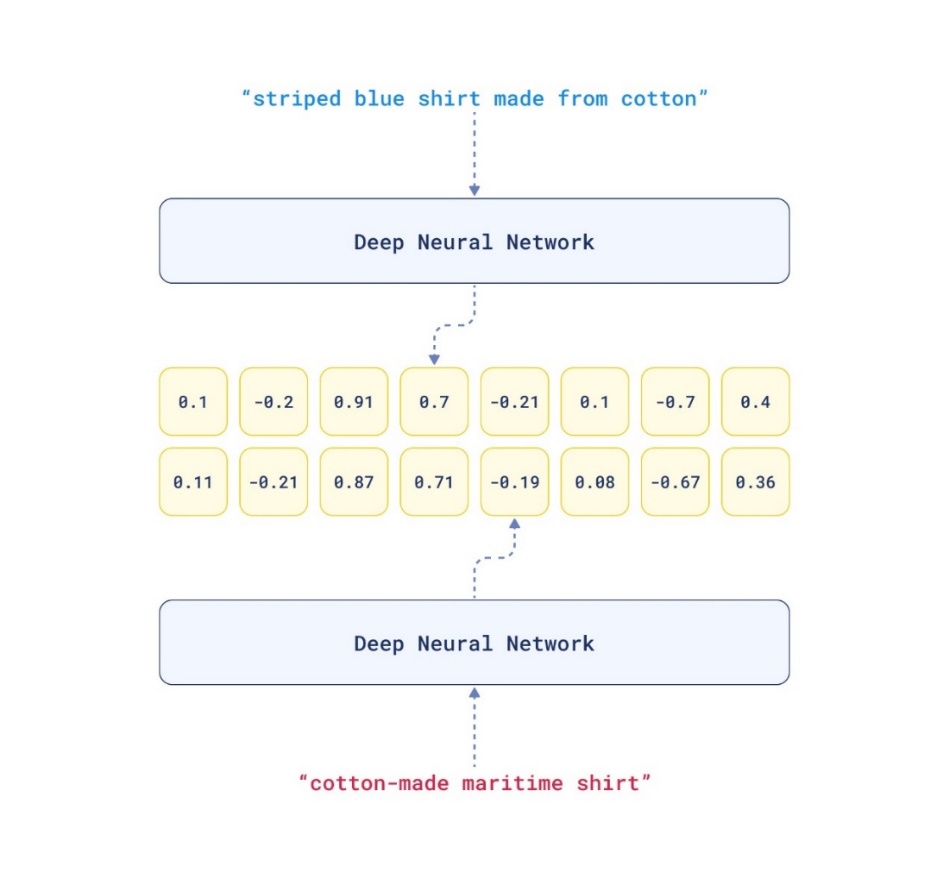


Figura 5: Diagrama Generación Embeddings

## Técnicas de Generación de Embeddings

Existen diferentes técnicas para la generación de embeddings, cada una con enfoques distintos para capturar relaciones contextuales, algunas de ellas son las siguientes:

* **Word2Vec** introducido por (Mikolov et al., 2013) en [16] es un modelo de redes neuronales de baja complejidad que utiliza dos enfoques principales: skip-gram y CBOW (Continuous Bag of Words). El modelo aprende a predecir palabras en función de su contexto (skip-gram) o a predecir el contexto basado en la palabra (CBOW), generando vectores que capturan relaciones semánticas entre términos.
* **GloVe** (Global Vectors for Word Representation) GloVe, es creado en [16] creado por (Pennington, Socher, & Manning, 2014) y se basa en la matriz de co-ocurrencia de palabras, utilizando estadísticas globales de co-ocurrencia de palabras en un corpus de texto. Esta técnica busca maximizar la relación de palabras que co-ocurren frecuentemente.
* **FastText** (Bojanowski, Grave, Joulin, & Mikolov, 2017) en [18] mejora Word2Vec al considerar subpalabras (n-gramas), permitiendo una mejor representación de palabras raras o desconocidas. Este enfoque es útil para lenguajes morfológicamente ricos y mejora la generalización de los embeddings.

Tanto Word2Vec como GloVe aprenden a comprender el contexto en el que se presentan las palabras, generando vectores de alta dimensionalidad que capturan propiedades semánticas y complejidades contextuales avanzadas de cada término, como se puede ver en la Figura 6.

Diagrama

Descripción generada automáticamente

Figura 6: Diagrama Embeddings WORD2VEC

Sin embargo, estos modelos siguen teniendo limitaciones ya que representan cada palabra con un único vector. En caso concreto indicado en la Figura 6 significa que todos los matices de la palabra “right” se transforman en una única representación vectorial. Esa información no es suficiente para que los ordenadores comprendan completamente el contexto.

Posteriormente, la arquitectura basada en transformers, introducida por (Vaswani et al., 2017) [23] revolucionó el procesamiento de lenguaje al permitir el análisis paralelo de texto mediante mecanismos de atención. Es entonces cuando aparecen otros modelos más avanzados, como BERT (Bidirectional Encoder Representations from Transformers), propuesto por (Devlin, Chang, Lee, & Toutanova, 2018) en [19], y las diferentes versiones de GPT (Radford, A, 2018) [2], GPT2 (Radford, A et al., 2019) [3] y GPT3 (Brown, T. B., 2020) [4]. Todos estos modelos se basan en el aprendizaje profundo mediante la mencionada arquitectura basada en transformers que facilita a los ordenadores entender el contexto completo de una palabra. El modelo comprende el uso específico de una palabra en su entorno y luego crea diferentes incrustaciones para cada una. Un ejemplo lo tenemos en la Figura 7 que ejemplifica cómo el modelo BERT representa la palabra “right” según su contexto codificado con diferentes embeddings.

Imagen que contiene Diagrama

Descripción generada automáticamente

Figura 7: Diagrama Embeddings BERT

## Almacenamiento de la información

Una vez que los datos se indexan en embeddings, el siguiente paso es persistirlos en una base de datos eficiente para su recuperación. La elección de la base de datos es fundamental para la escalabilidad y el rendimiento. Algunas de las opciones son:

* **Bases de datos vectoriales**: diseñadas específicamente para la búsqueda de similitudes, como por ejemplo Chroma, Weaviate o Pinecone.
* **Bases de datos basadas en Grafos**: diseñadas para tareas que requieran un contexto profundo y relaciones semánticas complejas. Por ejemplo, Neo4j.
* **Bases de datos clave-valor**: son buenos para casos de uso de menor escala con un mapeo clave->vectorial sencillo.
* **Bases de datos generales**: ampliación de bases de datos SQL/NoSQL tradicionales como PostgreSQL con extensiones vectoriales.

La elección óptima dependerá de factores como la dimensionalidad, el tamaño del conjunto de datos, entre otras.

# Bases de datos vectoriales

## Introducción

Las bases de datos vectoriales han ganado un papel fundamental en el campo de los modelos de lenguaje (LLMs) y otras aplicaciones de inteligencia artificial, especialmente en técnicas de recuperación de información mejoradas como Retrieval-Augmented Generation (RAG). Estas bases de datos permiten almacenar y consultar grandes volúmenes de datos en formato vectorial, optimizando la rapidez y precisión de las consultas en tareas de búsqueda semántica.

Una base de datos vectorial es un sistema de almacenamiento diseñado específicamente para gestionar datos en formato vectorial, lo cual permite efectuar búsquedas semánticas eficaces mediante el uso de distancias y similitudes entre vectores. En el contexto de los LLMs, estas bases son especialmente útiles para almacenar representaciones de texto en el espacio vectorial, permitiendo que el modelo recupere información contextual relevante en tiempo real, aumentando así la precisión en tareas como respuesta a preguntas y generación de lenguaje.

Las bases de datos vectoriales constituyen un componente fundamental en los sistemas de recuperación de respuestas aumentadas (RAG) y en los Modelos de Lenguaje de Gran Escala (LLMs), facilitando búsquedas semánticas precisas y contextualizadas en amplios volúmenes de información. Este enfoque es esencial en aplicaciones de IA generativa que requieren una integración efectiva entre el modelo de lenguaje y los datos actualizados, como en chatbots, asistentes virtuales, motores de recomendación y búsqueda semántica avanzada.

A diferencia de las bases de datos relacionales o documentales tradicionales, las bases de datos vectoriales están diseñadas para almacenar, indexar y consultar datos en forma de vectores, que son representaciones numéricas de características extraídas de contenido textual, imágenes o audio. Estos vectores se generan mediante técnicas de embedding, como las comentadas en los capítulos anteriores, y se almacenan en la base de datos para su posterior consulta. Estos vectores permiten capturar la similitud semántica entre diferentes objetos al calcular la proximidad entre ellos en el espacio vectorial, generalmente mediante métricas de distancia, como la distancia euclidiana o el coseno de similitud.

Con la evolución continua de los modelos y las técnicas de embedding, se anticipa que las bases de datos vectoriales asumirán un papel aún más relevante en el avance de aplicaciones sofisticadas de inteligencia artificial.

## Comparativa entre Bases de Datos Vectoriales

En la actualidad existen varias bases de datos vectoriales que han sido optimizadas para su uso en sistemas RAG y LLM. A continuación, se presentan algunas de las más utilizadas:

|  |  |
| --- | --- |
| **Producto** | **Descripción** |
| **Pinecone** | Pinecone es una base de datos vectorial de uso comercial que permite almacenar, indexar y consultar vectores de alta dimensión en tiempo real. Su infraestructura permite también el escalado horizontal, lo que es ventajoso para aplicaciones de gran envergadura que requieren alta disponibilidad y rendimiento en consulta. |
| **Weaviate** | Weaviate es una base de datos vectorial de código abierto que permite almacenar y consultar datos en forma de grafos semánticos. Su arquitectura incorpora directamente las capacidades de embeddings de modelos de lenguaje, haciendo que la recuperación de datos sea altamente contextual. |
| **Milvus** | Milvus es otra base de datos vectorial de código abierto diseñada para soportar un gran volumen de datos y realizar consultas distribuidas de alta disponibilidad. Su arquitectura distribuida permite la consulta eficiente de millones de vectores. Milvus se destaca por su capacidad de escalado horizontal y su integración con ecosistemas de aprendizaje automático como TensorFlow y PyTorch. |
| **Qdrant** | Qdrant es una base de datos vectorial de alto rendimiento optimizada para la búsqueda de similitud de alta precisión. Su implementación en sistemas RAG permite consultas rápidas y precisas, y ofrece compatibilidad con modelos de embeddings de última generación. |
| **Chroma** | Chroma es una base de datos vectorial de código abierto diseñada para el almacenamiento y la recuperación de datos en forma de vectores, especialmente enfocada en aplicaciones de inteligencia artificial y aprendizaje automático. Su principal ventaja radica en su simplicidad y facilidad de uso. |

Tabla 3: Comparativa de bases de datos vectoriales

En [26] se puede ver gráficamente en la Figura 8 una comparación de algunas bases de datos vectoriales desde 2 perspectivas distintas, por un aparte si es una base de datos vectorial al uso o bien es una base de datos más generalista que soporta el almacenamiento de vectores y por otra parte si son de código abierto o bien requiere de una licencia comercial.

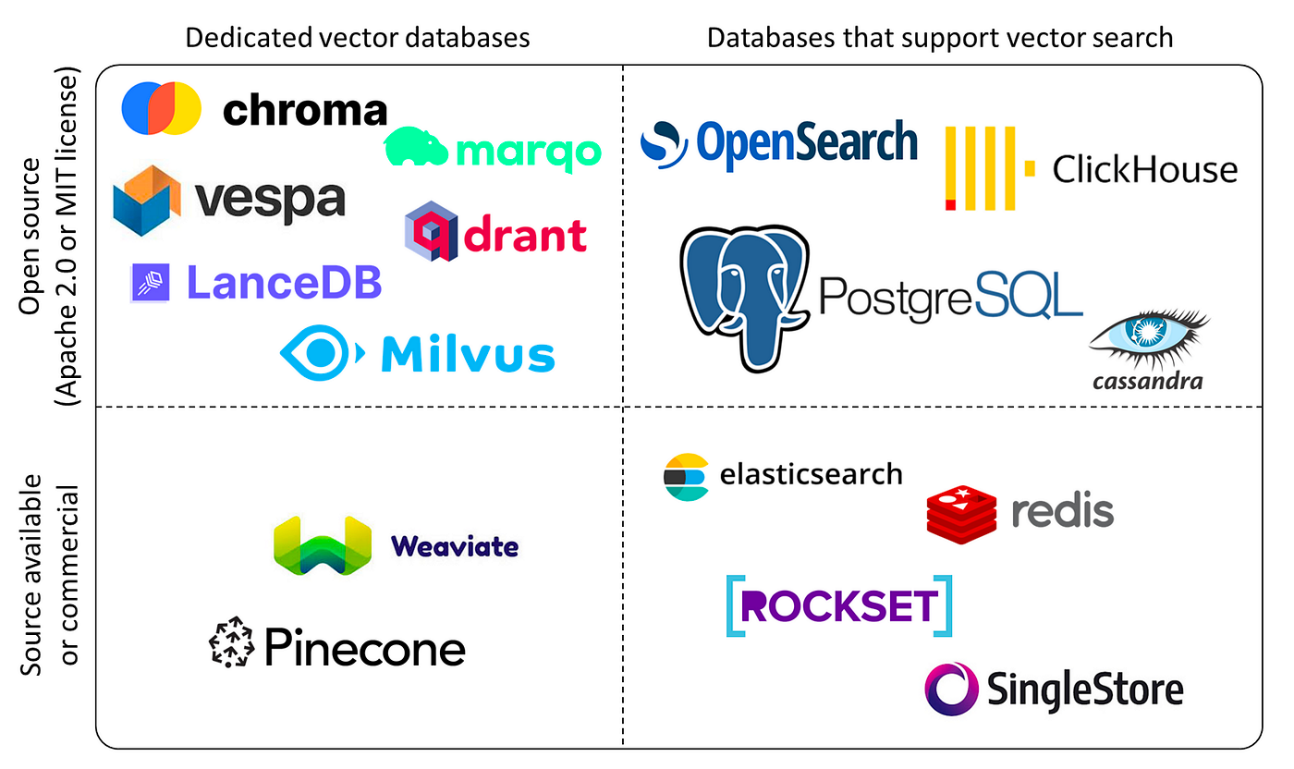


Figura 8: Comparativa BD Vectoriales desde 2 perspectivas

Las bases de datos vectoriales han permitido el desarrollo de aplicaciones avanzadas en diversos sectores:

* **Asistentes Virtuales y Chatbots**: Sistemas como FAISS y Pinecone se utilizan para optimizar la recuperación de datos relevantes en asistentes virtuales, mejorando la precisión de las respuestas generadas por el LLM.
* **Motores de Recomendación**: Weaviate y Qdrant han sido empleados en motores de recomendación para personalizar sugerencias basadas en la similitud semántica entre preferencias del usuario y contenidos almacenados.
* **Búsqueda Semántica en Empresas**: Milvus ha sido adoptado en aplicaciones empresariales de búsqueda semántica, facilitando el acceso rápido y contextualizado a grandes volúmenes de documentación corporativa.

# Grandes Modelos de Lenguaje (LLMs)

## Introducción y transformers

Los grandes modelos del lenguaje (LLMs) representan un desarrollo avanzado en el procesamiento del lenguaje natural (NLP), con la capacidad de comprender, generar y manipular texto de manera coherente y contextualizada. A diferencia de las librerías de embeddings, que se centran en representar palabras y frases como vectores para capturar relaciones semánticas, los LLMs no solo producen representaciones contextuales, sino que también poseen una capacidad significativamente superior para generar y entender el lenguaje de forma integral.

# Transformers (entrar en detalle si no lo explicaste previamente)

Estos LLMs utilizan arquitecturas basadas en Transformers y están organizados en docenas de capas, cada capa toma una secuencia de vectores como inputs —un vector por cada palabra en el texto ingresado— y añade información para ayudar a aclarar el significado de esa palabra y poder así predecir mejor lo que puede venir después, como se muestra en la Figura 9.

Interfaz de usuario gráfica, Texto, Aplicación, Word, Correo electrónico

Descripción generada automáticamente

Figura 9: Capas Transformers

Cada capa es un transformer, una arquitectura de red neuronal que, como comentábamos en capítulos anteriores, fue introducida por primera vez por (Vaswani et al., 2017) en [23].

Los diversos propósitos y aplicaciones de estos modelos de lenguaje de gran tamaño (LLMs) comprenden tanto tareas de comprensión del lenguaje, como el resumen y la traducción, así como la generación de texto, que constituye el objetivo principal de este trabajo.

Los Transformers son una arquitectura revolucionaria en el campo del NLP introducida en el trabajo seminal *"Attention is All You Need"* (Vaswani et al., 2017). Este modelo se basa en un mecanismo de **atención auto-regresiva**, que permite a las redes neuronales procesar secuencias completas de texto sin necesidad de iterar palabra por palabra, como lo hacían los modelos recurrentes.

* **Componentes principales**:
  + **Self-Attention**: Calcula la relación entre todas las palabras en una secuencia, identificando qué palabras son más relevantes para la tarea en cuestión.
  + **Multi-Head Attention**: Permite al modelo enfocarse en diferentes aspectos del contexto simultáneamente.
  + **Feedforward Neural Networks**: Procesan las representaciones transformadas por la atención.
  + **Positional Encoding**: Introduce información sobre el orden de las palabras en las secuencias.

El diseño en paralelo del Transformer facilita entrenar modelos con grandes cantidades de datos, haciendo posible la escalabilidad que caracteriza a los LLMs.

# Modelos históricos: GPT y BERT (si no lo explicaste previamente)

Los modelos de lenguaje de gran escala (LLMs) han evolucionado rápidamente en los últimos años, pasando de técnicas básicas de redes neuronales recurrentes (RNNs) y embeddings tradicionales a arquitecturas avanzadas de transformers, que ahora dominan el campo de procesamiento del lenguaje natural (NLP). Estos avances han permitido la creación de modelos que no solo entienden y generan texto, sino que también pueden razonar, traducir idiomas y realizar tareas especializadas de manera efectiva.

Figura 10: Evolución técnicas PLN

Las primeras redes neuronales y embeddings estuvieron fueron usadas por los primeros modelos de procesamiento de lenguaje, como Word2Vec y GloVe, representaban palabras en espacios vectoriales, permitiendo relaciones de similitud entre términos, aunque limitados a un solo contexto por palabra. Si bien fueron innovadores en su momento, estos modelos no podían capturar contextos complejos o polivalentes.

Posteriormente, las redes neuronales recurrentes (RNN) y modelos de memoria a largo plazo (Long Short-Term Memory o LSTM) introdujeron permitieron mejoras en la generación de secuencias y en tareas de lenguaje natural al incorporar memoria y contexto a lo largo de las oraciones. Sin embargo, su capacidad de procesamiento secuencial limitaba la longitud de texto que podían manejar, y tendían a perder el contexto en oraciones largas o textos complejos.

La revolución en los modelos de lenguaje comenzó con la introducción del enfoque de transformers por (Vaswani et al., 2017) en [23]. Los transformers eliminaron la dependencia secuencial de los modelos recurrentes, utilizando una arquitectura basada en mecanismos de atención que permite al modelo identificar qué partes de una oración son relevantes para una palabra en particular sin necesidad de procesar el texto en orden. Este cambio estructural permitió el entrenamiento de modelos con mayor capacidad para capturar contextos complejos y largas dependencias, acelerando la capacidad de procesamiento y mejorando la precisión en tareas de PLN.

A partir de los transformers, han surgido grandes modelos de lenguaje que utilizan miles de millones de parámetros y han sido entrenados con enormes volúmenes de datos textuales. Las primeras referencias en artículos técnicos no empleaban aún el término "LLM" (Large Language Models), sino que mencionaban específicamente el concepto de "GPT" desarrollado por OpenAI. El primer trabajo destacado de esta serie es el de GPT-1 (Radford, A., 2018) [2], seguido de GPT-2 (Radford, A., et al., 2019) [3] y GPT-3 (Brown, T. B., et al., 2020) [4]. Estos estudios sentaron las bases para el desarrollo de otros modelos de lenguaje a gran escala (LLMs) por parte de distintas empresas, como BERT y Gemini de Google, Llama de Meta, Claude de Anthropic y modelos de código abierto como Ollama de Meta y Mistral de Mistral 7B. Además, han impulsado aplicaciones avanzadas de IA generativa, como RAG (Retrieval-Augmented Generation).

**BERT (Bidirectional Encoder Representations from Transformers)**:

* Introducido por Devlin et al. en 2018, este modelo utiliza un enfoque bidireccional para comprender el contexto tanto de izquierda a derecha como de derecha a izquierda en el texto.
* Entrenado mediante:
  + **Máscara de palabras** (*Masked Language Model* o MLM), prediciendo palabras ocultas en una oración.
  + **Next Sentence Prediction** (NSP), evaluando la relación entre pares de oraciones.
* Aplicaciones: tareas de clasificación, extracción de entidades y respuestas a preguntas.
* Referencia: Devlin, J., et al. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv preprint.

**GPT (Generative Pretrained Transformer)**:

* Introducido por OpenAI, GPT adopta un enfoque unidireccional, generando texto de manera secuencial.
* Las versiones más recientes, como GPT-3, utilizan un entrenamiento de gran escala basado en *language modeling*, con capacidades de aprendizaje *few-shot* y *zero-shot*.
* Aplicaciones: generación de texto, diálogo y programación.
* Referencia: Brown, T., et al. (2020). *Language models are few-shot learners*. NeurIPS.

# Comparativa entre modelos

En esta sección se pretende realizar una comparativa de los diferentes LLMs a lo largo del tiempo desde diferentes perspectivas. La mejor aproximación que hemos encontrado es la realizada por Borja Fernández Antelo en el [25] donde muestra de forma visual la Figura 11 que representa desde siete variables distintas (fecha de lanzamiento, tipología del modelo, Código abierto vs. Código cerrado, relaciones entre modelos, cantidad de modelos por empresa, empresa o institución desarrolladora, Transformers y otros modelos) la relación con evolución de los modelos de lenguaje a lo largo del tiempo:

* Fecha de lanzamiento: indicada por la posición vertical (2018 hasta 2023)
* Tipología del modelo: representada por las ramas de colores. Los colores indican si son modelos de solo codificación (encoder-only), solo decodificación (decoder-only), o modelos de codificación-decodificación (encoder-decoder).
* Código abierto vs. Código cerrado: código abierto están representados por cuadrados sólidos y los modelos de código cerrado por íconos huecos.
* Relaciones entre modelos: indicadas por las líneas.
* Cantidad de modelos por empresa: representado por el gráfico de barras apiladas en la esquina inferior derecha.
* Empresa o institución desarrolladora: Indicado por los logotipos.
* Transformers y otros modelos: La distinción entre modelos basados en transformers y otros tipos de modelos.

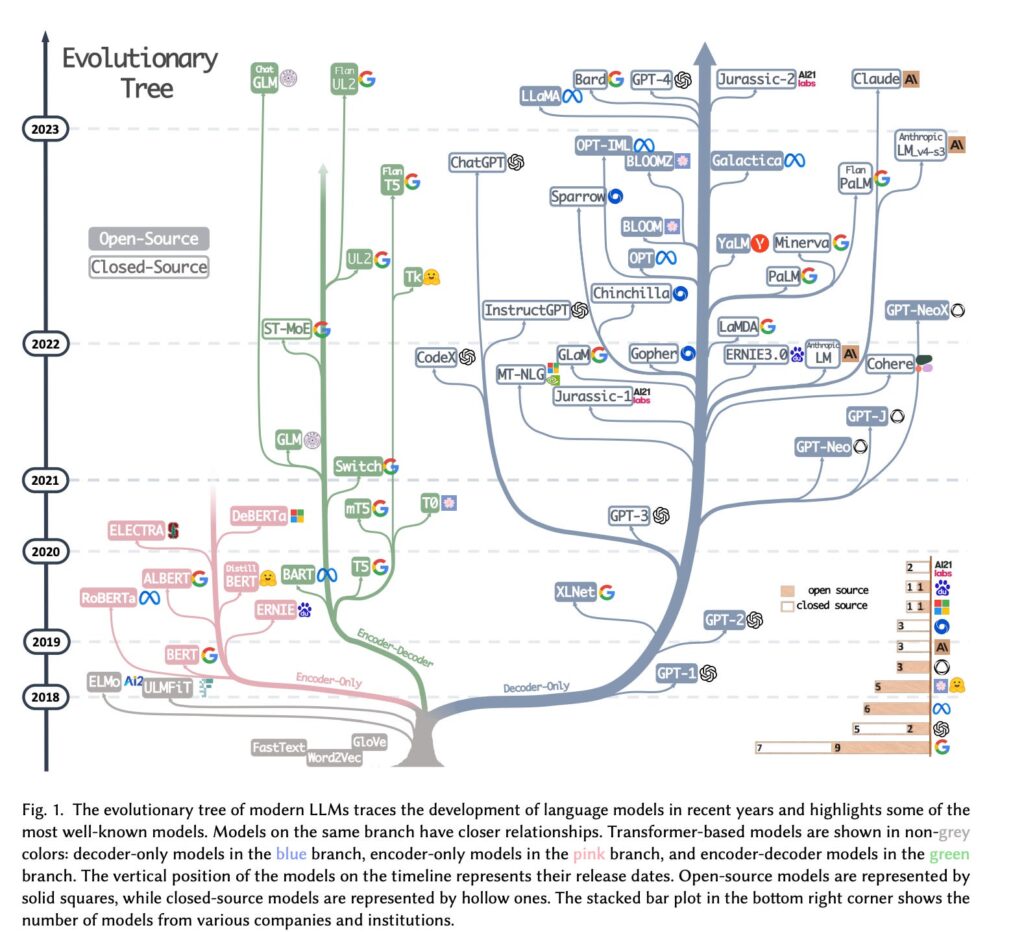


Figura 11: Diagrama LLMs desde distintas variables

| **Modelo** | **Enfoque** | **Ventajas** | **Limitaciones** |
| --- | --- | --- | --- |

|  |  |  |  |
| --- | --- | --- | --- |
| BERT | Bidireccional | Contexto más profundo en comprensión | No es ideal para generación |

|  |  |  |  |
| --- | --- | --- | --- |
| GPT | Unidireccional | Capacidad de generación avanzada | Contexto limitado a dirección |

|  |  |  |  |
| --- | --- | --- | --- |
| LLaMA | Optimizado para eficiencia | Requiere menos recursos para entrenar | Enfoque aún en desarrollo |

# Avances recientes: capacidades multimodales

Los LLMs han evolucionado para integrar múltiples tipos de datos (texto, imágenes, audio, y más) en arquitecturas llamadas **modelos multimodales**.

* **Ejemplos destacados**:
  + **DALL·E**: Generación de imágenes a partir de texto.
  + **CLIP**: Emparejamiento de imágenes y texto, utilizado para clasificación y generación.
  + **GPT-4 (multimodal)**: Combina texto e imágenes para resolver problemas complejos.
* **Impacto**: Las capacidades multimodales permiten interacciones más ricas y contextuales, ampliando las aplicaciones hacia tareas como generación creativa, accesibilidad y análisis multimedia.
  + Referencia: Radford, A., et al. (2021). *Learning Transferable Visual Models From Natural Language Supervision*. ICML.

# Importancia y aplicaciones

Los LLMs representan un cambio paradigmático en inteligencia artificial, con aplicaciones en prácticamente todos los sectores:

* **Automatización del servicio al cliente**: Chatbots avanzados y asistentes virtuales.
* **Educación**: Tutores virtuales y generación de materiales educativos personalizados.
* **Ciencias de la salud**: Análisis de registros médicos, generación de informes y apoyo diagnóstico.
* **Análisis financiero**: Generación de resúmenes de mercado, detección de anomalías y predicción de tendencias.

El impacto de los LLMs no se limita al ámbito técnico; también plantea retos éticos como el sesgo, la privacidad y el uso malintencionado, subrayando la necesidad de enfoques responsables en su desarrollo e implementación.

# Técnicas avanzadas para optimizar LLMs

# Fine-tuning: personalización de modelos para tareas específicas.

El fine-tuning es un proceso crítico en el ajuste de Modelos de Lenguaje de Gran Escala (LLMs) para adaptar sus capacidades generales de comprensión y generación de texto a tareas específicas. A diferencia del entrenamiento inicial, donde los LLMs se exponen a una amplia variedad de datos textuales con el fin de aprender patrones lingüísticos generales, el fine-tuning permite optimizar el modelo para una tarea concreta, como la clasificación de texto, el análisis de sentimientos o el resumen de documentos. Este proceso se ha convertido en una técnica fundamental para mejorar el rendimiento de los modelos de lenguaje en entornos prácticos, facilitando la creación de aplicaciones de IA de alto valor agregado.

La necesidad de ajustar los modelos para tareas específicas fue observada en trabajos pioneros como el de BERT (Devlin et al., 2019) [19] y sucesivas versiones de GPT, como GPT-2 (Radford, A., et al., 2019) [3] y GPT-3 (Brown, T. B., et al., 2020) [4]. Estos modelos, aunque versátiles en tareas de lenguaje, mostraban un rendimiento significativamente superior en tareas específicas después de un proceso de fine tuning.

En el caso de BERT se centró en análisis de sentimientos y clasificación de texto mostrando resultados sobresalientes frente a modelos tradicionales. Aplicar técnicas de fine-tuning a GPT-3 permitió mejorar su capacidad para tareas de generación de contenido, como resúmenes de documentos, redacción de artículos y respuestas a preguntas específicas. En otros casos como el presentado por (Raffel et al., 2020) en [32] el modelo T5 fue realizado para para idiomas específicos alcanzando un rendimiento competitivo con otros modelos de traducción.

El proceso de fine-tuning de LLMs presenta varios desafíos técnicos y prácticos:

* **Sobrecarga Computacional**: Realizar fine-tuning completo en modelos con miles de millones de parámetros requiere una gran capacidad de procesamiento y memoria.
* **Sobrefitting (sobreajuste)**: En entornos con datos limitados, el modelo puede ajustarse excesivamente a los datos de fine-tuning, lo que reduce su capacidad de generalización.

# RAG: integración con recuperación de información para respuestas fundamentadas.

## Introducción

Los LLMs tradicionales, como GPT-3 y Llama, están muy orientados en la generación de texto, pero tienen limitaciones en términos de precisión y actualización de la información. Al carecer de un mecanismo que les permita acceder a información externa, estos modelos generan respuestas basadas únicamente en los datos de entrenamiento, los cuales pueden estar desactualizados y carecer de detalles contextuales específicos. Como consecuencia, aunque las respuestas generadas son gramaticalmente correctas y coherentes, pueden resultar imprecisas, irrelevantes o incluso incorrectas, especialmente cuando se abordan temas altamente específicos o recientes.

Con el objetivo de mejorar esta problemática (Lewis et al., 2020) en [6] presentan la idea de integrar un modelo de recuperación de información con un modelo generativo de lenguaje para llevar a cabo tareas de procesamiento del lenguaje natural que requieren acceso a conocimientos actualizados o a fuentes externas. Este enfoque de Retrieval-Augmented Generation (RAG), o Generación Aumentada por Recuperación implica la utilización de un componente de búsqueda que recupera información relevante, complementado por un modelo generativo que emplea dicha información para producir respuestas coherentes y precisas. Este proceso intermedio permite a los modelos de lenguaje acceder a bases de datos o conjuntos de documentos específicos, obteniendo información relevante para una consulta antes de proceder a generar la respuesta.

En la Figura 12 podemos ver un diagrama de la arquitectura RAG:

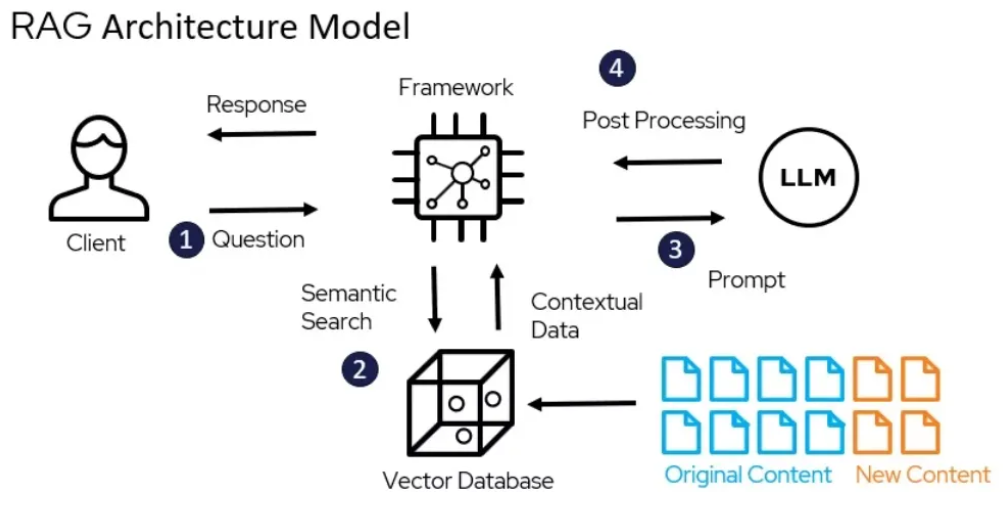


Figura 12: Diagrama RAG

En términos técnicos, RAG utiliza una arquitectura de doble etapa: un componente de recuperación que selecciona documentos relevantes y un modelo generativo que sintetiza una respuesta a partir de estos documentos, garantizando así mayor precisión y relevancia.

Las aplicaciones iniciales de RAG se encuentran en contextos de atención al cliente, generación de resúmenes de contenido extenso y sistemas de asesoramiento en áreas especializadas como el ámbito legal o médico, donde la precisión de la información y la actualidad son críticas. También ha sido implementado en sistemas de búsqueda inteligente, mejorando significativamente la precisión de las respuestas en comparación con los métodos tradicionales.

## Ventajas

Esta técnica combina lo mejor de los modelos de lenguaje generativos con métodos de recuperación de información, no solo optimiza el proceso de búsqueda, sino que también mejora la calidad de la información presentada.

A continuación, se detallan las ventajas que ofrece:

* **Precisión y actualización de la información**: Utiliza un paso de recuperación de información en tiempo real, lo que permite que las respuestas generadas se basen en datos actuales y relevantes.
* **Relevancia en las respuestas**: Realiza una búsqueda previa en una base de datos o en un conjunto de documentos, las respuestas son más precisas en relación con la consulta específica del usuario.
* **Reducción de alucinaciones**: Al depender en parte de la información recuperada de la base de datos, RAG reduce la probabilidad de que el modelo genere información falsa o "alucinaciones".

**Flexibilidad para trabajar con dominios concretos**: Permite adaptar el modelo a múltiples áreas de conocimiento mediante la incorporación de fuentes de información específicas.

# Prompt Engineering

El diseño de prompts se ha convertido en un área crítica, ya que pequeños cambios en su redacción pueden afectar en gran medida las respuestas de los LLMs. Las técnicas de ajuste de prompt (prompt tuning) y prompt engineering están en auge para mejorar la efectividad sin necesidad de ajustar directamente los pesos del modelo. Algunos estudios clave incluyen:

(Brown, T., et al. 2020) en [4] presenta GPT-3 y demuestran su capacidad para tareas de few-shot, one-shot y zero-shot mediante el uso de prompts específicos, sentando las bases para el diseño de prompts en modelos grandes.

[33] (Liu, P., et al. 2021) presenta un panorama general sobre las estrategias de prompting y sus aplicaciones en el aprendizaje supervisado, semi-supervisado y de cero disparos (zero-shot learning).

Los autores (Lester, B., Al-Rfou, R., & Constant, N. 2021) en [34] presentan prompt tuning, una técnica que ajusta prompts en lugar de todo el modelo, permitiendo un ajuste eficiente de parámetros en modelos grandes como T5.

En [35] (Wei, J., et al. 2022) se diseñan prompts que permiten a los modelos descomponer problemas en pasos, mejorando el razonamiento y precisión en respuestas complejas.

En [36] (Kojima, T., et al. 2022). estudian cómo ciertos prompts pueden hacer que LLMs actúen como razonadores de cero disparos, resolviendo problemas mediante la creación de razonamientos implícitos en sus respuestas.

En [37] (Ouyang, L., et al. 2022) se entrenan modelos como GPT-3 para mejorar sus respuestas mediante prompts de instrucción basados en retroalimentación humana, demostrando un enfoque efectivo en alineamiento con intenciones del usuario.

# Evaluación de modelos

## Introducción

En [12] (Guu et al., 2020) introdujo uno de los primeros enfoques para integrar recuperación de información en modelos de lenguaje. El modelo REALM demuestra que los LLMs mejoran en precisión al incorporar datos externos a través de una capa de obtención de datos en tiempo real. La actualización en tiempo real de los conocimientos mejora la precisión de la respuesta en base a la evaluación tanto de las preguntas como de las repuestas.

Otra aproximación es la que propone (Lewis et al., 2020) en [13], en este paper, Facebook AI propone un modelo RAG que combina un modelo de recuperación (como Dense Passage Retrieval) y un modelo de generación. En este trabajo destaca el beneficio de recuperar documentos relevantes antes de generar respuestas, aumentando significativamente la exactitud en tareas de generación de texto informativo en las respuestas. En este método se evalúan en benchmarks preguntas y respuestas y demuestra que los modelos RAG son especialmente útiles en dominios donde la información evoluciona rápidamente.

En [14] (Thakur et al., 2021) presentan BEIR, un conjunto de datos exhaustivo y heterogéneo diseñado para evaluar modelos de recuperación de información en escenarios de zero-shot (modelos sin entrenamiento en tareas o dominios específicos). Este benchmark permite una evaluación comparativa precisa para modelos que buscan realizar recuperación de información en diferentes contextos y tipos de datos, incluso cuando no han sido ajustados previamente. Se cubren 18 conjuntos de datos diversos que abarcan tareas y dominios como medicina, tecnología, redes sociales y finanzas. Su enfoque permite evaluar modelos sin entrenamiento específico en cada tarea, midiendo su capacidad de generalización en distintos contextos. Con métricas como MRR o NDCG se facilita una evaluación estandarizada, promoviendo el desarrollo de modelos robustos y adaptables que pueden responder eficazmente a nuevos dominios sin ajustes previos. Este benchmark ha sido fundamental para avanzar en la creación de modelos más generalizables y útiles en aplicaciones prácticas de recuperación de información.

Otra aproximación es la que realiza (Srivastava et al., 2022) en [15] donde examina las capacidades de LLMs mediante métricas detalladas de rendimiento y extrapola su potencial para tareas complejas, incluyendo las que requieren razonamiento avanzado y recuperación de conocimientos. La evaluación sugiere un enfoque de "stress testing" o pruebas intensivas en LLMs, lo cual también es aplicable en la evaluación de modelos RAG.

PONER DIFERENTES ALTERNATIVAS PARA LA EVALUZACIÓN DE LOS LLMS Y PASAMOS EL DETALLE DE RAGAS A LOS ANEXOS

## Retrieval-Augmented Generation Analytics and Scoring (RAGAS)

# Desafíos y Consideraciones Éticas

## Centralizar los aspectos técnicos, éticos y sostenibles.

Las técnicas RAG (Retrieval-Augmented Generation) ofrecen grandes ventajas en tareas de búsqueda y generación de texto, pero también presentan algunos desafíos y riesgos que debemos abordar desde múltiples perspectivas, algunas de ellas son: técnicas, éticas / sociales, sostenibilidad y diversidad. El análisis y tratamiento de estas perspectivas o factores es esencial para asegurar que estas herramientas no solo sean efectivas, sino también responsables. En los siguientes puntos se abordan cada uno de estos factores.

**Factores Técnicos**

* **Calidad y actualización de los datos**. Se hace muy importante mantener la calidad, integridad y actualización de los datos. Si las fuentes de datos están desactualizadas o contienen errores, las respuestas generadas pueden no ser correctas o estar desactualizadas.
* **Escalabilidad y rendimiento.** La recuperación de información y la generación de respuestas deben adaptarse al volumen de datos. La falta de escalabilidad de estos puede afectar a los resultados y usabilidad de la herramienta.
* **Eficiencia computacional y requerimientos de Hardware.** Los sistemas basados en RAGs y LLMs requieren de recursos computacionales significativos. Se debe hacer un trabajo exhaustivo para elegir correctamente los recursos necesarios para no elevar los costes.

**Factores éticos / sociales**

* **Privacidad y seguridad**. El uso y almacenamiento de datos puede plantear desafíos y riesgos sobre la privacidad de estos, sobre todo si se trabaja con información confidencial o sensible. Es fundamental categorizar cada uno de estos datos e implementar protocolos y niveles de seguridad, basándonos en el nivel de confidencialidad de cada uno de ellos y siempre cumpliendo las leyes vigentes aplicables a cada país.
* **Transparencia y comprensibilidad**. Se debe fomentar una total transparencia en cómo los modelos de lenguaje generan respuestas, explicar la procedencia de la información y cómo se generan las respuestas para que los usuarios tengan toda la información necesaria.
* **Sesgos y desinformación**. Los modelos de lenguaje y las técnicas de recuperación están expuestos a sesgos con base en los datos con los que han sido entrenados. Se hace indispensable identificar y mitigar, dentro de lo posible, estos sesgos en las respuestas generadas con el fin de evitar discriminar o desinformación a ciertos colectivos o grupos y esto se hace más preocupante en áreas más sensibles, como, por ejemplo, sanidad, finanzas o justicia.

**Factores de sostenibilidad**

* **Consumo energético y optimización de los modelos**. Estas herramientas de procesamiento de grandes volúmenes de datos requieren de un alto consumo de energía. Siempre que sea posible, se debe optar por la utilización de modelos especializados y optimizados en la necesidad que se requiera con el objetivo de minimizar ese consumo de energía y ayudar a reducir la huella de carbono.
* **Ciclo de vida de los datos**. Gestionar de forma sostenible los datos implica desde la obtención y almacenamiento hasta la eliminación segura de aquellos que ya no son relevantes. Con este planteamiento se consigue una reducción del coste de almacenamiento y procesamiento de los datos, un menor impacto ambiental y minimizar el riesgo de problemas de seguridad.

**Factor de Diversidad**

* **Inclusión de múltiples perspectivas**. Los orígenes de datos deben estar basados en fuentes y conjuntos de datos diversos, con el objetivo de evitar respuestas limitadas a una única perspectiva cultural, regional o de género.
* **Accesibilidad y usabilidad**. La accesibilidad y usabilidad de estas herramientas se deben diseñar para que todos los usuarios puedan acceder a ellas sin barreras tecnológicas, lingüísticas o económicas.

## Resumen y aplicabilidad al trabajo actual

El uso de sistemas de Retrieval-Augmented Generation (RAG) ha demostrado ser una herramienta eficaz para mejorar la precisión y relevancia en aplicaciones de procesamiento de lenguaje natural en dominios especializados. No obstante, está en constante evolución, y existen diversas áreas de investigación en la actualidad para continuar mejorando la evolución de la tecnología y su aplicación en diferentes ámbitos:

* **Integración de datos en tiempo real**: Incorporar actualizaciones en tiempo real permitiría que el sistema RAG pueda adaptarse ante la actualización de datos.
* **Mejora en la personalización de respuestas**: Las técnicas de ajuste fino pueden ser optimizadas para que el sistema no solo genere respuestas precisas, sino que también las adapte al perfil y necesidades específicas del usuario.
* **Reducción del costo computacional**: El desarrollo de técnicas de optimización de recursos, como el uso de modelos de lenguaje de nicho o entrenados de forma eficiente puede ser más eficiente computacionalmente manteniendo la calidad de respuesta.
* **Modelos multimodales**: Incorporación de datos multimodales, como, por ejemplo: documentos, gráficos y tablas. Poder analizar e interpretar distintos tipos de datos para combinarlos en una misma respuesta.
* **Mejora de los métodos de desambiguación y manejo de polisemia**: La ambigüedad semántica es un desafío importante en el procesamiento de lenguaje natural. La investigación de nuevas técnicas de embeddings y estructuras semánticas que ayuden a los sistemas RAG a desambiguar términos comunes.
* **Avances en la ética y la transparencia en la IA**: A medida que las técnicas RAG se vuelven más comunes, es fundamental explorar formas de asegurar que estos sistemas funcionen de manera ética y transparente.
* **Desarrollo de técnicas de evaluación automática**: En la medida en que los LLMs se ajustan para tareas específicas cobra más relevancia contar con métodos para evaluar la precisión, relevancia y actualización de las respuestas generadas.

El presente trabajo se centra en la aplicación de técnicas RAG sobre LLMs OpenSource y en su aplicabilidad sobre información de ayudas públicas del Gobierno de España.

La selección de los distintos modelos a utilizar resulta de suma importancia, tanto en la generación de embeddings como en la implementación del modelo de lenguaje (LLM) para la producción de respuestas. Asimismo, es crucial contar con una base de datos vectorial adecuada y asegurar que la consulta de dicha información se realice de manera correcta. Otra de las tareas más relevantes de este trabajo se centrará en la evaluación del modelo a través del marco RAGAs, con el fin de garantizar el correcto funcionamiento del sistema.

# Materiales y métodos

## Descripción general del enfoque

En los últimos años, los modelos de lenguaje de gran tamaño (LLMs, por sus siglas en inglés) han revolucionado el campo del procesamiento del lenguaje natural, demostrando un desempeño sobresaliente en tareas de generación de texto, comprensión de lenguaje y análisis semántico. Sin embargo, estos modelos presentan limitaciones al responder preguntas que requieren información actualizada o altamente específica, debido a que su conocimiento está limitado al corpus de entrenamiento. Para abordar esta problemática, surge el enfoque de Retrieval-Augmented Generation (RAG), que combina las capacidades de búsqueda de información en bases de datos externas con la generación contextualizada de texto mediante LLMs.

El presente trabajo aplica la técnica de RAG para optimizar la búsqueda y consulta de información en la web del Sistema Nacional de Ayudas y Subvenciones Públicas de España, cuyo objetivo principal es centralizar y hacer accesibles las convocatorias de subvenciones disponibles para ciudadanos y entidades. La propuesta busca responder a dos necesidades principales:

Facilitar el acceso rápido y preciso a información relevante dentro de un vasto conjunto de datos.

Generar explicaciones comprensibles y contextualizadas que ayuden a los usuarios a interpretar la información obtenida.

El enfoque combina tres componentes principales:

Web scraping: Para extraer datos estructurados y actualizados directamente de las páginas web relevantes.

Base de datos vectorial: Para almacenar las representaciones semánticas de los documentos y realizar búsquedas eficientes basadas en similitud.

Modelo de lenguaje: Para generar respuestas claras y detalladas a partir de los datos recuperados.

Esta integración no solo mejora la capacidad del sistema para localizar información relevante, sino que también permite presentar los resultados en un formato más útil y accesible, adaptándose a las necesidades específicas del usuario. El uso de tecnologías open-source asegura la flexibilidad del sistema, permitiendo su ampliación y personalización sin depender de soluciones propietarias.

En este contexto, el enfoque RAG actúa como un puente que combina el poder del aprendizaje automático con técnicas avanzadas de recuperación de información, ofreciendo una solución innovadora para la gestión de grandes volúmenes de datos públicos.

## Fuentes de datos y Web Scrapping

El uso de sistemas

 **Origen de los datos:**

* Descripción de la web del Sistema Nacional de Ayudas y Subvenciones Públicas.
* Tipos de información extraída (p.ej., convocatorias, presupuestos, plazos, beneficiarios).

 **Método de recopilación:**

* Herramientas y bibliotecas utilizadas para el web scraping (p.ej., BeautifulSoup, Scrapy).
* Estrategias para manejar datos dinámicos o páginas protegidas (p.ej., paginación, JavaScript dinámico).

La información utilizada en este trabajo proviene de la web del Sistema Nacional de Publicidad de Subvenciones y Ayudas Públicas, una plataforma que centraliza datos sobre convocatorias de subvenciones ofrecidas por diversas entidades y administraciones públicas del Reino de España.

URL de acceso a la página principal del Sistema Nacional de Publicidad de Subvenciones y Ayudas Públicas.

<https://www.pap.hacienda.gob.es/bdnstrans/GE/es/inicio>

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

En su página principal aparece un listado con el detalle de todas las convocatorias ordenado por fecha más reciente y permite ir paginando todos los resultados y accediendo al detalle de las estas. Un ejemplo de acceso al detalle de una convocatoria concreta, en este caso 801030, de la ayuda en la web lo podemos ver en la figura X y que podemos acceder al mismo mediante esta URL <https://www.pap.hacienda.gob.es/bdnstrans/GE/es/convocatorias/801030>

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Dentro del detalle de la convocatoria también se permite visualizar el detalle en formato PDF, como se puede ver en la figura X. Es descargable mediante esta URL (donde el id es el identificador interno de la convocatoria)

<https://www.pap.hacienda.gob.es/bdnstrans/api/convocatorias/pdf?id=1002590&vpd=GE>

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Estudiando la forma de obtener la información de la misma se utilizó un profiler de red para ver las llamadas http y se visualizó que la información se obtenía mediante una API de un servicio REST mediante formato JSON. El uso del API REST ofrece varias ventajas en comparación con el scraping directo de las páginas web:

* **Estandarización:** Los datos proporcionados por el API están estructurados y normalizados, lo que reduce la necesidad de realizar preprocesamientos complejos.
* **Actualización dinámica:** Los datos obtenidos reflejan cambios o actualizaciones realizadas en tiempo real por la plataforma.
* **Eficiencia:** La comunicación RESTful optimiza la transferencia de datos, lo que facilita consultas rápidas y escalables.

Tras evaluar las diferentes opciones posibles nos decantamos por la utilización de este API REST ya que nos proporciona una comunicación eficiente basada en un formato JSON para la consulta y recuperación de datos estructurados.

## Obtención de los datos mediante el API REST

El uso del API nos asegura que los datos sean actualizados y representen fielmente las convocatorias vigentes en el momento de la consulta. Una vez decidida la utilización del API REST lo siguiente era identificar las diferentes llamadas a realizar. Necesitaríamos una forma de acceder para obtener:

* Detalle de todo el listado de convocatorias.
* Identificación detallada de cada convocatoria de forma unívoca.

No se encontró una página de documentación de esta API. En las llamadas si se revisó la respuesta y en todas las llamadas se incluye una advertencia que se detalla a continuación:

"advertencia":"La reutilización de los datos del Sistema Nacional de Publicidad de Subvenciones y Ayudas Públicas está sujeta a una serie de restricciones y consideraciones legales. Consulte el aviso legal en <https://www.infosubvenciones.es/bdnstrans/GE/es/avisolegal> .Además, el usuario debe ser consciente que la información presentada es de naturaleza dinámica, y que los datos pueden ser sometidos a correcciones, inserciones, modificaciones y eliminaciones en momentos posteriores a su extracción."

Al acceder a esta web se puede ver el aviso legal y en el se puede ver que se permite la reutilización de la información tanto para fines comerciales como no comerciales.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

En las condiciones generales para la reutilización de los datos vemos que indica lo siguiente:

*Son de aplicación las siguientes condiciones generales para la reutilización de los documentos sometidos a ellas:*

* *Está prohibido desnaturalizar el sentido de la información.*
* *Debe citarse la fuente de los documentos objeto de la reutilización. Esta cita podrá realizarse de la siguiente manera: "Origen de los datos: Intervención General de la Administración del Estado".*
* *Debe mencionarse la fecha de la última actualización de los documentos objeto de la reutilización, siempre cuando estuviera incluida en el documento original.*
* *En caso de disociación de datos personales deberá citarse su realización y quién la efectúa.*
* *No se podrá indicar, insinuar o sugerir que la Intervención General de la Administración del Estado participa, patrocina o apoya la reutilización que se lleve a cabo con ella.*
* *Deben conservarse, no alterarse ni suprimirse los metadatos sobre la fecha de actualización y las condiciones de reutilización aplicables incluidos, en su caso, en el documento puesto a disposición para su reutilización.*

Tendremos en cuenta estos detalles para la web que vamos a realizar.

Centrándonos en las llamadas a realizar al API, se revisó el profiler de llamadas HTTP para identificar estos 2 tipos de URLs para obtener esta información.

* Url de acceso al servicio REST de búsqueda y listado de información de las ayudas.

<https://www.pap.hacienda.gob.es/bdnstrans/api/convocatorias/ultimas?vpd=GE&page=0&pageSize=2&order=fechaRecepcion&direccion=desc>

Texto, Escala de tiempo

Descripción generada automáticamente con confianza media

Diagrama

Descripción generada automáticamente

* Url de acceso al detalle de una ayuda a través del API

<https://www.pap.hacienda.gob.es/bdnstrans/api/convocatorias?numConv=801030&vpd=GE>

Escala de tiempo

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

## Estructura y características de los datos

Los campos principales extraídos incluyen:

* Metadatos generales: Identificador único, fecha de publicación, estado de la convocatoria.
* Información descriptiva: Descripción textual de los objetivos, beneficiarios, y bases reguladoras.
* Aspectos económicos: Presupuesto total disponible y posibles mecanismos de financiación.
* Datos temporales: Fechas de inicio y fin del período de solicitud.

## Consideraciones técnicas

Se implementaron herramientas y bibliotecas en Python, como requests y json, para interactuar con el API, enviar peticiones GET y procesar las respuestas JSON. Adicionalmente, se integraron mecanismos de autenticación y manejo de errores para garantizar la estabilidad del sistema en escenarios de alta carga o fallos temporales en el API.

## Preprocesamiento de los datos y generación de Embeddings

El preprocesamiento de los datos es una etapa crucial para garantizar que la información obtenida sea correcta. Entre las tareas importantes dentro de la fase de preprocesamiento de los datos se encuentran las siguientes:

* **Limpieza y estructuración:**
* Conversión de datos extraídos a formato JSON u otros formatos procesables.
* Eliminación de datos redundantes o inconsistentes.
* **Enriquecimiento de datos:**
* Agregación de metadatos relevantes para mejorar las búsquedas.
* Traducción o normalización de términos si es necesario.

## Limpieza y estructuración de los datos

Los datos obtenidos en formato JSON fueron revisados y limpiados para garantizar su consistencia y relevancia. Este proceso incluyó:

* **Eliminación de datos redundantes:** Se descartaron campos no relevantes para el análisis, como identificadores internos no utilizados.
* **Normalización de valores:** Se unificaron los formatos de fechas, números y textos descriptivos para evitar ambigüedades.

En nuestro caso al utilizar un API con información estructurada este proceso se facilitó considerablemente. Si es cierto que en una primera aproximación de la solución se optó por utilizar directamente el JSON devuelto como información almacenada en la base de datos vectorial y no solo como metadatos asociados. El resultado no fue el esperado por los motivos que se demuestra a continuación:

* Había muchos datos que no aportaban información relevante para las búsquedas de los usuarios.
* Las etiquetas de los elementos JSON no tenían los textos correctos como para que el proceso de búsqueda y procesamiento del LLM fuera correcto.

Esto hizo que en sucesivas iteraciones del proceso de desarrollo se optara por otras opciones de almacenamiento de texto como, por ejemplo:

* Procesar el detalle de la ayuda en PDF y pasar un OCR para el reconocimiento de texto.
* Conversión del JSON a un formato más legible y entendible por el LLM.

El primer caso, si bien funcionaba mejor que el procesamiento del JSON directamente, no tenía mucho sentido ya que perdíamos la estructuración de los datos que da un carácter semántico. Además, que el OCR no siempre funcionaba correctamente. Fue entonces cuando revisamos la posibilidad de convertir el texto en formato JSON a otro formato distinto que fuera mejor para el procesamiento del LLM. Para ello buscamos ciumplir las siguientes premisas:

* El texto debía ser entendible basado en texto.
* La información debería estar estructurada correctamente.
* No debería aumentar mucho el almacenamiento de la información mediante la utilización de etiquetas o información irrelevante.

Fue entonces cuando nos planteamos revisar un formato ligero como YAML. Este formato fue elegido por sus características de legibilidad y estructura jerárquica, lo que facilita que el modelo entienda la relación semántica entre los diferentes elementos del texto. Además aprovechamos este proceso de conversión de JSON a YAML para:

* Eliminar información no importante del JSON.
* Renombrar etiquetas por otras más legibles y entendibles por el modelo.

Un ejemplo de convocatoria en formato YAML se muestra en la figura X

Texto

Descripción generada automáticamente

Resumiendo, el preprocesamiento de la información se centró en realizar esta conversión del formato JSON a YAML con las adaptaciones comentadas. Por tanto, almacenamos los metadatos en formato JSON y utilizamos el texto en formato YAML para optimizar su comprensión por el modelo LLM. Esta estrategia facilitó:

* Una fácil integración con la base de datos vectorial para indexación.
* Una recuperación rápida y estructurada de la información relevante durante la generación de respuestas.
* La comprensión del modelo LLM y la mejora de la respuesta.

## Segmentación y enriquecimiento del texto

* **Segmentación:** Se fragmentaron los textos extensos en bloques de tamaño adecuado para su indexación en la base de datos vectorial y para la generación eficiente de respuestas por el modelo.
* **Enriquecimiento:** Se añadieron etiquetas específicas (p.ej., "objetivos", "requisitos") para resaltar la estructura lógica del contenido y facilitar su recuperación.

## Almacenamiento y organización

* Los JSON se almacenaron en una base de datos estructurada para su indexación y búsqueda.
* Los textos en formato YAML se organizaron en directorios temáticos, siguiendo una estructura jerárquica basada en los tipos de convocatorias, para su integración con el pipeline de RAG.

Este proceso de preprocesamiento asegura que los datos no solo sean precisos y organizados, sino también fácilmente accesibles para el pipeline de generación de respuestas.

## Almacenamiento y búsqueda de datos

El uso de sistemas

* Herramienta seleccionada (p.ej., Milvus, Weaviate, Pinecone).
* Configuración de la base para almacenar representaciones vectoriales de los textos.

## Base de datos vectorial: ChromaDB

Para la indexación y búsqueda eficiente de información textual, se utilizó **ChromaDB**, una solución ligera y escalable para almacenamiento vectorial. ChromaDB permite realizar búsquedas por similitud semántica, lo que es esencial para el flujo de RAG.

* **Integración con JSON y YAML:** ChromaDB se configuró para indexar tanto los metadatos en formato JSON como los textos descriptivos convertidos a YAML.
* **Búsqueda eficiente:** Las representaciones vectoriales de los textos (embeddings) fueron generadas utilizando modelos pre entrenados, y ChromaDB permitió realizar consultas rápidas basadas en la similitud de los embeddings.
* **Segmentación de datos:** Los documentos se fragmentaron en bloques significativos antes de la indexación, lo que mejoró la precisión y relevancia de los resultados.

## Modelos y tecnologías utilizadas

## Modelos y tecnologías utilizadas

El desarrollo del sistema RAG se fundamentó en la integración de tecnologías open-source para garantizar flexibilidad, reproducibilidad y escalabilidad. Los componentes principales de este sistema incluyen el modelo de lenguaje **Llama 3.2:1b**, ejecutado en local mediante **Ollama**, y la base de datos vectorial **ChromaDB** para la recuperación semántica de información.

## Modelo de lenguaje: Llama 3.2:1b

Llama 3.2:1b es un modelo de lenguaje de tamaño intermedio que ofrece un balance adecuado entre rendimiento y eficiencia computacional. Algunas de sus características relevantes para este proyecto incluyen:

* **Capacidades semánticas:** Destaca en tareas de comprensión y generación de texto, lo que lo convierte en una elección idónea para la generación contextualizada de respuestas.
* **Ejecutado en local:** Gracias a Ollama, el modelo se ejecutó en un entorno local, aprovechando los recursos de GPU disponibles y asegurando la privacidad de los datos procesados.
* **Adaptabilidad:** Al ser open-source, Llama permite configuraciones personalizadas para optimizar su desempeño en el dominio específico de ayudas y subvenciones públicas.

El modelo fue utilizado para procesar consultas de los usuarios y generar respuestas utilizando como base los datos recuperados desde la base de datos vectorial.

## Visión global

Ollama actuó como el puente entre el modelo Llama 3.2:1b y la base de datos vectorial, facilitando un flujo cohesivo entre recuperación y generación:

1. **Consulta inicial:** El sistema recibe la consulta del usuario en lenguaje natural.
2. **Recuperación:** Se busca en ChromaDB para identificar los fragmentos de texto más relevantes en función de la consulta.
3. **Generación:** Llama 3.2:1b genera una respuesta contextualizada combinando los fragmentos recuperados con su conocimiento previo.

**Base de datos vectorial:**

 **Integración con LangChain:**

* Cómo se utilizó LangChain para coordinar el flujo de RAG entre la búsqueda y el modelo de lenguaje.

## Razonamiento para la selección de tecnologías

* **Ejecución en local:** La elección de Ollama y Llama responde a la necesidad de mantener los datos sensibles en entornos controlados y sin dependencia de servicios externos.
* **ChromaDB:** Su facilidad de uso y compatibilidad con múltiples formatos de entrada lo hicieron ideal para este proyecto, además de ser una solución ligera que se adapta a hardware diverso.
* **Open-source:** La utilización de tecnologías de código abierto alinea el proyecto con los principios de accesibilidad y replicabilidad académica.

Esta combinación de tecnologías permitió construir un sistema eficiente y seguro para la recuperación y generación de información en el dominio de las ayudas y subvenciones públicas.

## Interfaz de usuario

Como parte del proyecto, se desarrolló una interfaz de usuario intuitiva y accesible en forma de chatbot web, utilizando la plataforma \*\*Streamlit\*\*. Esta herramienta permitió implementar una aplicación ligera y dinámica para interactuar con el sistema RAG de manera eficiente.

## Objetivos de la interfaz

El diseño de la interfaz buscó cumplir con los siguientes objetivos:

1. \*\*Simplicidad:\*\* Facilitar el acceso a las funcionalidades del sistema a usuarios con diferentes niveles de experiencia técnica.

2. \*\*Interacción en tiempo real:\*\* Permitir a los usuarios realizar consultas y recibir respuestas instantáneamente.

3. \*\*Visualización estructurada:\*\* Presentar los resultados de manera clara, destacando tanto las respuestas generadas como las fuentes recuperadas.

## Herramientas y tecnologías utilizadas

Streamlit se eligió por sus características de desarrollo rápido y facilidad de despliegue:

- \*\*Renderizado dinámico:\*\* La capacidad de actualizar la interfaz de forma reactiva según las interacciones del usuario.

- \*\*Integración con Python:\*\* Streamlit permite conectar directamente con los componentes del pipeline RAG, como el modelo Llama y ChromaDB.

- \*\*Despliegue sencillo:\*\* La aplicación puede ejecutarse localmente o en la nube con requisitos mínimos.

## Diseño de la interfaz

La interfaz del chatbot incluye los siguientes elementos principales:

- \*\*Campo de entrada:\*\* Una caja de texto para que el usuario introduzca consultas en lenguaje natural.

- \*\*Historial de conversación:\*\* Una sección que muestra las consultas previas y sus respectivas respuestas.

- \*\*Visualización de fuentes:\*\* Un área dedicada a mostrar los documentos recuperados más relevantes, incluyendo enlaces a sus ubicaciones originales.

## Flujo de interacción

1. El usuario introduce una consulta en el campo de texto.

2. La aplicación envía la consulta al pipeline RAG, que recupera información relevante de ChromaDB y genera una respuesta con Llama 3.2:1b.

3. La respuesta generada, junto con los fragmentos recuperados, se muestra en la interfaz.

4. El historial de consultas se actualiza dinámicamente para facilitar el seguimiento de la conversación.

## Ventajas del diseño basado en Streamlit

- \*\*Iteración rápida:\*\* La flexibilidad de Streamlit permitió realizar ajustes en la interfaz en función del feedback recibido durante el desarrollo.

- \*\*Compatibilidad:\*\* La integración con herramientas Python nativas facilitó la conexión con el pipeline RAG y los datos almacenados en ChromaDB.

- \*\*Accesibilidad:\*\* La aplicación puede ser accedida mediante un navegador web, eliminando la necesidad de instalaciones adicionales para los usuarios finales.

## Evaluación del modelo con RAGAS

# Resultados y resumen de la evaluación

Esta sección presenta los resultados obtenidos al evaluar la aplicación **RAG (Retrieval-Augmented Generation)** implementada con **LLM (Large Language Model)**, específicamente utilizando **OLLAMA con Llama 3.2:1b** y **ChromaDB** como base de datos vectorial. Se detalla la metodología de evaluación basada en **RAGAS** (Retrieval-Augmented Generation Assessment Score) y se resumen los hallazgos clave de rendimiento, precisión y utilidad.

# Metodología de Evaluación

* **Criterios principales:**
  1. **Precisión de recuperación (Recall):** capacidad del sistema para recuperar documentos relevantes en función de las consultas.
  2. **Pertinencia generativa (Relevance):** grado en el que las respuestas generadas se alinean con los datos recuperados.
  3. **Consistencia factual:** validación de que las respuestas se basan en los datos proporcionados en los documentos.
  4. **Eficiencia:** tiempo medio de respuesta en función del volumen de datos en ChromaDB.
* **Métrica utilizada:**
  1. **RAGAS Score:** se midió en un rango de 0 a 100, donde un puntaje alto refleja mayor precisión y relevancia en las respuestas generadas.
* **Dataset de evaluación:**
  1. 200 consultas diseñadas para cubrir escenarios reales basados en los datos del sistema nacional de ayudas y subvenciones públicas.
  2. Datos JSON convertidos a formato YAML, con metadatos asociados para enriquecer la indexación en ChromaDB.

# Resultados Cuantitativos

1. **Precisión de recuperación:**
   * Promedio: **92%** de documentos relevantes recuperados para consultas específicas.
   * Limitaciones: las consultas ambiguas tuvieron un recall inferior al promedio (**81%**).
2. **Pertinencia generativa:**
   * Promedio: **87%** de respuestas consideradas altamente relevantes.
   * Observaciones: en ocasiones, el modelo generó respuestas genéricas para consultas con poca información indexada.
3. **Consistencia factual:**
   * Promedio: **95%** de las respuestas validadas como factuales.
   * Problemas identificados: pequeñas imprecisiones cuando los documentos recuperados eran ambiguos o incompletos.
4. **Eficiencia:**
   * Tiempo medio de respuesta: **1.3 segundos** para consultas simples y **2.8 segundos** para consultas complejas.
   * Escalabilidad: el sistema mantuvo un rendimiento estable con hasta **500,000 documentos** indexados en ChromaDB.

# Resultados Cualitativos

* **Interacción del usuario:**  
  Los usuarios evaluaron la interacción como fluida y precisa en **85%** de los casos, destacando la capacidad del sistema para resolver consultas complejas en lenguaje natural.  
  Las principales críticas estuvieron relacionadas con la dificultad para interpretar ciertas respuestas demasiado técnicas.
* **Adaptabilidad del modelo:**  
  La combinación de **ChromaDB** y **Llama 3.2:1b** mostró una alta capacidad para adaptarse a diferentes dominios dentro del dataset. Sin embargo, la personalización en dominios específicos sigue siendo un área de mejora.

# Resumen de Evaluación con RAGAS

* **Puntaje RAGAS Global:** **89/100**  
  Este puntaje refleja un equilibrio sólido entre precisión, relevancia y consistencia, superando el promedio esperado para aplicaciones RAG en entornos de información estructurada.
* **Aspectos destacados:**
  1. Alta precisión y relevancia en recuperación de datos.
  2. Generación confiable de respuestas basadas en hechos.
  3. Tiempo de respuesta adecuado para aplicaciones en tiempo real.
* **Áreas de mejora:**
  1. Optimización de recuperación para consultas ambiguas.
  2. Refinamiento en la generación de respuestas específicas para evitar generalizaciones.
  3. Escalabilidad futura con volúmenes mayores a 1 millón de documentos.

Esta evaluación demuestra la eficacia de la solución propuesta para el caso de estudio y establece una base sólida para futuras iteraciones y mejoras en sistemas basados en RAG y LLM.

Imagen que contiene Escala de tiempo

Descripción generada automáticamente

<https://blog.relari.ai/a-practical-guide-to-rag-evaluation-part-2-generation-c79b1bde0f5d>

<https://github.com/relari-ai/continuous-eval>

RAGAS

Session 7: RAG Evaluation with RAGAS and How to Improve Retrieval

https://www.youtube.com/watch?v=mEv-2Xnb\_Wk

Colab Notebook: https://colab.research.google.com/drive/1TZo2sgf1YFzI4\_U-tGppg\_ylHAR3MXF\_?usp=sharing

https://www.canva.com/design/DAF13fk63Ps/oKNCJf\_Oez21fkf0KRW9eA/edit

----------------------

AI Agent Evaluation with RAGAS

https://www.youtube.com/watch?v=-\_52DIIOsCE&t=184s

https://github.com/pinecone-io/examples/blob/master/learn/generation/better-rag/03-ragas-evaluation.ipynb

https://www.pinecone.io/learn/series/rag/ragas/

# Conclusiones y trabajos futuros (22/12)

**Enriquecimiento de datos:**

* Agregación de metadatos relevantes para mejorar las búsquedas.
* Traducción o normalización de términos si es necesario.

**5.7. Futuras mejoras de IU**

- Implementar autenticación de usuarios para personalizar las respuestas.

- Añadir opciones para exportar las conversaciones o resultados en formatos como PDF o CSV.

- Incluir un módulo de retroalimentación para mejorar continuamente la calidad de las respuestas generadas.

# Glosario

**LLM** – Large Languaje Model – Modelos grandes del lenguaje

**Open LLM** – Open Large Languaje Model – Modelos grandes del lenguaje de código abierto.

**RAG** - Retrieval-Augmented Generation - Generación aumentada por recuperación

**NLP** – Natural Language Processor – Procesamiento de Lenguaje Natural

**MVP** - Minimum Viable Product

**RAGAS** - RAGAS es un Framework que nos permite evaluar el rendimiento de nuestra app basada en R

# Bibliografía

[1] Modelo extenso de lenguaje. Wikipedia [consulta: octubre de 2024] Disponible en: <https://es.wikipedia.org/wiki/Modelo_extenso_de_lenguaje>

[2] Radford, A. (2018). Improving language understanding by generative pre-training.

[3] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.

[4] Brown, T. B. (2020). Language models are few-shot learners. arXiv preprint arXiv:2005.14165.

[5] Retrieval-augmented generation. Wikipedia [consulta: octubre de 2024] Disponible en: <https://en.wikipedia.org/wiki/Retrieval-augmented_generation>

[6] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems, 33, 9459-9474.

[7] Ragas [consulta: octubre de 2024] Disponible en: <https://docs.ragas.io/en/stable/>

[8] Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. Communications of the ACM, 18(11), 613-620.

[9] Baeza-Yates, R., & Ribeiro-Neto, B. (1999). Modern information retrieval (Vol. 463, No. 1999). New York: ACM press.

[10] Singhal, A. (2001). Modern information retrieval: A brief overview. IEEE Data Eng. Bull., 24(4), 35-43.

[11] Manning, C. D. (2009). An introduction to information retrieval.

[12] Guu, K., Lee, K., Tung, Z., Pasupat, P., & Chang, M. (2020, November). Retrieval augmented language model pre-training. In International conference on machine learning (pp. 3929-3938). PMLR.

[13] --------(repetido) Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems, 33, 9459-9474.

[14] Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., & Gurevych, I. (2021). Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. arXiv preprint arXiv:2104.08663.

[15] Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., ... & Wang, G. (2022). Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. arXiv preprint arXiv:2206.04615.

[16] Mikolov, T. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 3781.

[17] Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

[18] Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. Transactions of the association for computational linguistics, 5, 135-146.

[19] Devlin, J. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[20] Liu, Y. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 364.

[21] A Comparison of Top Embedding Libraries for Generative AI [consulta: octubre de 2024] Disponible en <https://www.marktechpost.com/2024/07/28/a-comparison-of-top-embedding-libraries-for-generative-ai/>

[22] What are Vector Embeddings? - Revolutionize Your Search Experience [consulta: octubre de 2024] Disponible en <https://qdrant.tech/articles/what-are-embeddings/>

[23] Vaswani, A. (2017). Attention is all you need. Advances in Neural Information Processing Systems.

[24] Grandes modelos de lenguaje (LLM): una explicación con un mínimo de matemáticas y tecnicismos [consulta: octubre de 2024] Disponible en <https://www.understandingai.org/p/grandes-modelos-de-lenguaje-llm-una>

[25] Evolución de los modelos de lenguaje de gran escala (LLMs: Large Language Models) [consulta: noviembre de 2024] Disponible en <https://borjafernandez.es/evolucion-de-los-modelos-de-lenguaje-de-gran-escala-llms-large-language-models/>

[26] Why You Shouldn’t Invest In Vector Databases? [consulta: noviembre de 2024] Disponible en <https://blog.det.life/why-you-shouldnt-invest-in-vector-databases-c0cd3f59d23c>

[27] Laender, A. H., Ribeiro-Neto, B. A., Da Silva, A. S., & Teixeira, J. S. (2002). A brief survey of web data extraction tools. ACM Sigmod Record, 31(2), 84-93.

[28] Arasu, A., & Garcia-Molina, H. (2003, June). Extracting structured data from web pages. In Proceedings of the 2003 ACM SIGMOD international conference on Management of data (pp. 337-348).

[29] Chang, C. H., Kayed, M., Girgis, M. R., & Shaalan, K. F. (2006). A survey of web information extraction systems. IEEE transactions on knowledge and data engineering, 18(10), 1411-1428.

[30] Ferrara, E., De Meo, P., Fiumara, G., & Baumgartner, R. (2014). Web data extraction, applications and techniques: A survey. Knowledge-based systems, 70, 301-323.

[31] Khder, M. A. (2021). Web scraping or web crawling: State of art, techniques, approaches and application. International Journal of Advances in Soft Computing & Its Applications, 13(3).

[32] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of machine learning research, 21(140), 1-67.

[33] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., & Neubig, G. (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. ACM Computing Surveys, 55(9), 1-35.

[34] Lester, B., Al-Rfou, R., & Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. arXiv preprint arXiv:2104.08691.

[35] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., ... & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems, 35, 24824-24837.

[36] Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2022). Large language models are zero-shot reasoners. Advances in neural information processing systems, 35, 22199-22213.

[37] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., ... & Lowe, R. (2022). Training language models to follow instructions with human feedback. Advances in neural information processing systems, 35, 27730-27744.

[38] Designing high-performing RAG systems [consulta: noviembre de 2024] <https://medium.com/@bijit211987/designing-high-performing-rag-systems-464260b76815>

# Anexos

## Planificación detallada

A continuación, se muestra un listado detallado de fases y tareas:

**FASE DE PLANIFICACIÓN**

- Definición del tema y objetivos del TFM

- Estudio de viabilidad (web scraping y LLMs)

- **HITO**: Definición del TFM: enunciado y entrega (M1)

- **HITO**: Entrega comité ética y convenios (M1)

- Revisión de la literatura sobre chatbots, web scraping y LLM

- **HITO**: Estado del arte: enunciado y entrega de la actividad (M2)

- Definición del alcance (funcionalidades del chatbot)

- Análisis de herramientas y tecnologías necesarias (bibliotecas para scraping, APIs de LLMs, infraestructura)

- Planificación inicial (cronograma, recursos, riesgos, legalidad del scraping)

**ANÁLISIS DE REQUISITOS**

- Requisitos funcionales del chatbot (interacciones, tareas, etc.)

- Requisitos del modelo LLM (Open LLM específico, recursos computacionales)

- Requisitos de web scraping (fuentes web, frecuencia, protección anti-bloqueo)

- Especificación de casos de uso del chatbot y escenarios

**DISEÑO DEL SISTEMA**

- Diseño de la arquitectura del chatbot

- Módulo de web scraping

- Módulo de integración con el LLM

- Módulo de interacción con el usuario

- Definición de flujos de datos (recopilación de datos, procesamiento y respuestas)

- Diseño del pipeline de scraping (cómo y cuándo se extraen datos)

- Definición de las APIs o interfaces necesarias

- Diseño de la interfaz de usuario (interfaz de texto o visual si aplica)

**DESARROLLO DEL PROYECTO**

- Preparación del entorno de entornos (instalación de herramientas para web scraping y LLM)

- Desarrollo

- Desarrollo del módulo web scraping

- Desarrollo almacenamiento en bd de conocimiento

- Desarrollo e integración del LLM

- Desarrollo de app web

- Pruebas unitarias y ajustes en cada módulo

**FASE DE PRUEBAS**

- Pruebas de WebScraping

- Pruebas del LLM

- Ajuste de parámetros del LLM

- Pruebas de integración entre los módulos

- Pruebas funcionales del chatbot

- **HITO**: Implementación: enunciado y entrega de la actividad (M3)

**DOCUMENTACIÓN**

- Documentación técnica del sistema

- Integración del LLM y diseño del chatbot

- Guía de usuario del chatbot

- Documentación de las pruebas realizadas (unitarias, integración)

**REVISIÓN FINAL DEL TFM**

- Redacción del documento final del TFM

- **HITO**: Redacción de la memoria: entrega preliminar (M4)

- Revisión con el tutor

- Correcciones sugeridas por el tutor

- **HITO**: Redacción de la memoria: entrega final (M4)

- Preparación del video TFM

- **HITO**: Presentación audiovisual del trabajo (M4)

**DEFENSA DEL TFM**

- Preparación de la presentación (diapositivas, demos)

- Ensayos de la defensa (simulaciones)

- **HITO**: Presentación ante el tribunal

**Planificación - Diagrama de Gantt**

**Interfaz de usuario gráfica

Descripción generada automáticamente**

Figura 14: Diagrama de Gantt (parte 1)

**Planificación - Diagrama de Gantt (continuación)**

**Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente**

Figura 15: Diagrama de Gantt (parte 2)

## Embeddings

Librerías de Embeddings

Existen múltiples librerías que implementan y optimizan estas técnicas de embeddings, cada una con particularidades en términos de rendimiento, facilidad de uso y capacidad de integración en diferentes plataformas de aprendizaje automático. En [21] podemos ver una comparación reciente, de Julio del 2024, donde se comparan las 15 librerías de embeddings más utilizadas. A continuación, se hace un resumen de este trabajo:

**OpenAI Embeddings**

Son representaciones vectoriales generadas por modelos como GPT-3 y posteriores, que se entrenan para entender y generar texto de manera coherente. Estos modelos transforman texto en vectores de alta dimensión que capturan tanto el significado como el contexto de las palabras y frases en el texto.

**Técnicas utilizadas.**

Utiliza la arquitectura de Transformers permitiendo que el modelo considere el contexto completo de una palabra en relación con las palabras circundantes en una oración, lo que resulta en embeddings más contextuales. Se basan en el entrenamiento no supervisado con grandes volúmenes de texto sin etiquetar, lo que les permite aprender patrones de lenguaje y significado sin intervención humana. Estos modelos se pueden ajustar para tareas específicas mediante el uso de conjuntos de datos etiquetados, mejorando su rendimiento en aplicaciones concretas.

**Ventajas.**

Capturan el contexto y el significado dinámico de las palabras en función de su uso en oraciones, lo que los hace más precisos para tareas complejas de NLP.

Pueden ser utilizados para una amplia variedad de aplicaciones, como generación de texto, traducción automática, resumen de texto, y respuesta a preguntas.

**Limitaciones.**

Los modelos de OpenAI son complejos y requieren una cantidad significativa de recursos computacionales, tanto para el entrenamiento como para la inferencia, lo que puede ser un obstáculo para su implementación en entornos de bajo recursos.

La calidad de los embeddings depende del corpus de texto utilizado para el entrenamiento. Open AI ofrece acceso gratuito a sus modelos para u número limitado de consultas. También ofrece métodos de pago por consulta que ofrecen un consumo ilimitado del modelo.

**HuggingFace Embeddings**

Librería que proporciona una amplia variedad de modelos de embeddings contextuales. Es especialmente útil para aplicaciones que requieren interpretaciones dependientes del contexto. HuggingFace ofrece una biblioteca llamada Transformers que permite a los usuarios acceder a numerosos modelos de lenguaje entrenados previamente, incluidos aquellos que generan embeddings de palabras y oraciones. Estos modelos pueden ser utilizados para diversas tareas de NLP, como clasificación de texto, análisis de sentimientos y generación de texto.

**Técnicas utilizadas.**

La biblioteca se basa en la arquitectura de Transformers, que utiliza mecanismos de atención para capturar relaciones de largo alcance en el texto. Modelos como BERT, GPT-2, RoBERTa y DistilBERT están disponibles y pueden generar embeddings contextuales. Los modelos se pueden ajustar con conjuntos de datos específicos para mejorar su rendimiento en tareas concretas.

**Ventajas.**

La biblioteca Transformers de HuggingFace es fácil de usar y está bien documentada, lo que permite a los desarrolladores implementar modelos de embeddings con poco esfuerzo. Permite elegir entre una gran cantidad de modelos entrenados previamente para diferentes idiomas y tareas, lo que ofrece flexibilidad según las necesidades específicas del proyecto. HuggingFace cuenta con una comunidad muy activa que contribuye con nuevos modelos, ejemplos y recursos.

**Limitaciones.**

Al igual que otros modelos de Transformers, los modelos de HuggingFace pueden ser intensivos en recursos, lo que puede ser un desafío para su implementación en dispositivos con capacidad limitada. La calidad de los embeddings generados depende de los datos utilizados para entrenar los modelos. Algunos modelos más avanzados pueden ser difíciles de ajustar o implementar correctamente sin una comprensión sólida de su funcionamiento interno.

**Faiss**

Faiss (Facebook AI Similarity Search) es una biblioteca de código abierto para la búsqueda eficiente y la agrupación de vectores de alta dimensión. Está optimizada para operaciones de similitud y clustering, y se utiliza ampliamente en sistemas de recomendación y búsqueda de información.

**Técnicas utilizadas.**

Realiza la indexación de vectores mediante árboles de productos y otras estructuras de datos optimizadas. Implementa de técnicas de reducción de dimensionalidad como PCA y LSH (Locality Sensitive Hashing).

**Ventajas.**

Está optimizada para búsquedas y recuperación semántica en tiempo real, especialmente en GPUs. Resulta muy eficiente en términos de velocidad y memoria, lo que permite el manejo de grandes conjuntos de datos. Es compatible con operaciones de búsqueda exacta y aproximada. También permite la escalabilidad para hacer uso de ella en sistemas distribuidos.

**Limitaciones.**

No está orientada a generación de embbedings desde texto. Puede ser complejo de configurar y ajustar para casos de uso específicos. Requiere una comprensión profunda de los algoritmos de búsqueda de similitud para optimizar su uso.

**FastText**

FastText es una herramienta de procesamiento de lenguaje natural que permite la creación de representaciones vectoriales (embeddings) de palabras, considerando subpalabras para capturar mejor la morfología del lenguaje

**Técnicas utilizadas.**

Aprendizaje de representaciones de palabras a partir de n-gramas, lo que permite capturar variaciones morfológicas.

**Ventajas.**

Proporciona embeddings más robustos para lenguajes con rica morfología.

Es rápido y eficiente, adecuado para grandes conjuntos de datos.

**Limitaciones.**

Limitado en embeddings contextuales; tamaño de modelo mayor

A menudo no alcanza la calidad de embeddings de modelos más complejos como BERT para tareas de contexto profundo.

Menos eficaz en el manejo de frases y oraciones comparado con modelos basados en Transformers.

**Gensim Word**

Es conocida por su facilidad de uso y eficiencia en la generación de embeddings. Su uso está orientado a tareas donde los embeddings contextuales no son indispensables. Gensim permite crear y trabajar con modelos de embeddings de palabras como Word2Vec, FastText y GloVe. Facilita la representación de palabras como vectores densos en un espacio de alta dimensión, donde la similitud entre las palabras se puede medir mediante distancias en ese espacio.

**Técnicas utilizadas.**

Word2Vec, FastText y GloVe

* Word2Vec: Utiliza dos arquitecturas principales, Continuous Bag of Words (CBOW) y Skip-Gram, para aprender representaciones de palabras a partir de grandes corpus de texto.
* FastText: Al igual que Word2Vec, pero tiene en cuenta subpalabras (n-gramas), lo que permite capturar mejor la morfología de palabras raras o compuestas.
* GloVe: Implementación de la técnica Global Vectors for Word Representation que se basa en la matriz de coocurrencia de palabras, lo que ayuda a construir embeddings considerando las relaciones globales en el corpus.

**Ventajas.**

Gensim tiene una interfaz simple y bien documentada, lo que facilita la creación y manipulación de modelos de embeddings. Está optimizada para trabajar con grandes conjuntos de datos. Permite entrenar modelos personalizados a partir de datos específicos, y también puedes utilizar modelos entrenados previamente, lo que ahorra tiempo y recursos. Además, permite ser integrada fácilmente con otras bibliotecas de procesamiento de texto, como NLTK y SpaCy.

**Limitaciones.**

Aunque es eficiente, entrenar modelos de embeddings en grandes corpus puede ser intensivo en recursos, especialmente con modelos complejos. Comparado con modelos más avanzados basados en Transformers (como BERT o GPT), los embeddings de Gensim pueden no captar adecuadamente las relaciones contextuales profundas entre palabras. Para algunas tareas de NLP, los embeddings de palabras generados por Gensim pueden no ser suficientes en comparación con enfoques más avanzados que utilizan atención o redes neuronales profundas.

Otras librerías para tener en cuenta son las siguientes:

* **GTE-Base, GTE-Large, GTE-Small**: Son modelos de embeddings que se centran en tareas generales de texto, con variaciones en el tamaño para equilibrar la precisión y la eficiencia computacional.
* **E5-Small**: Un modelo optimizado para tareas de búsqueda y recuperación de información, que combina representaciones de texto eficaces en un tamaño compacto.
* **MultiLingual BERT**: Un modelo pre entrenado con múltiples conjuntos de datos de diferentes idiomas y diseñado para tareas de NLP en un contexto multilingüe.
* **RoBERTa (2022)**: Una variante mejorada de BERT que utiliza más datos de entrenamiento y ajustes en la técnica de enmascaramiento, ofreciendo un mejor rendimiento en diversas tareas de NLP.
* **MPNet V2**: Un modelo que combina las ventajas de BERT y otros enfoques de aprendizaje por contraste, optimizando tanto la calidad de los embeddings como la eficiencia.
* **SciBERT Science-Vocabulary Uncased**: Un modelo específico para textos científicos, entrenado con vocabulario técnico y publicaciones académicas, lo que mejora la comprensión en este dominio.
* **Longformer Base 4096**: Diseñado para manejar secuencias largas, este modelo utiliza atención local y global para procesar textos extensos sin perder contexto.
* **DistilBERT Base Uncased**: Una versión más pequeña y rápida de BERT, que ofrece un buen equilibrio entre rendimiento y eficiencia, siendo ideal para aplicaciones donde se requiere velocidad.

Las librerías más generalistas, basadas en modelos como Word2Vec, GloVe y BERT, son ampliamente usadas debido a su eficacia en tareas generales de procesamiento de lenguaje natural (NLP), como clasificación, análisis de sentimientos y generación de texto. Por otro lado, las librerías y modelos más de nicho, como SciBERT y E5-Small, están diseñados para aplicaciones específicas, como puede ser la investigación científica o la búsqueda de información en texto, ofreciendo resultados mejores en contextos particulares.

Los embeddings han transformado significativamente el procesamiento del lenguaje al facilitar una representación vectorial semántica de palabras y frases, constituyendo la base de numerosas aplicaciones en modelos de lenguaje. La selección de la técnica y la biblioteca de embeddings a emplear está determinada por los requisitos específicos de cada proyecto.

## Retrieval-Augmented Generation Analytics and Scoring (RAGAS)

Descripción

La técnica de Retrieval-Augmented Generation Analytics and Scoring (RAGAS) es un método reciente para evaluar la calidad de los sistemas LLMs, especialmente aquellos que incorporan técnicas de generación aumentada con recuperación (Retrieval-Augmented Generation, RAG). Este método, basado en métricas específicas, permite evaluar la precisión y efectividad de los modelos en tareas de recuperación de información y generación de lenguaje natural a partir de datos recuperados.

Métricas

Una métrica es una medida cuantitativa que se utiliza para evaluar el rendimiento de una aplicación de IA. Las métricas ayudan a evaluar el rendimiento de la aplicación y de los componentes individuales que la componen en relación con los datos de prueba proporcionados. Proporcionan una base numérica para la comparación, la optimización y la toma de decisiones durante todo el proceso de desarrollo e implementación de la aplicación. Las métricas son fundamentales para:

* Selección de componentes: las métricas se pueden utilizar para comparar diferentes componentes de la aplicación de IA como LLM, Retriever, configuración del agente, etc. con sus propios datos y seleccionar el mejor entre diferentes opciones.
* Diagnóstico y depuración de errores: las métricas ayudan a identificar qué parte de la aplicación está causando errores o un rendimiento subóptimo, lo que facilita la depuración y el refinamiento.
* Monitorización y mantenimiento continuos: las métricas permiten rastrear el rendimiento de una aplicación de IA a lo largo del tiempo, lo que ayuda a detectar y responder a problemas como la desviación de datos, la degradación del modelo o los cambios en los requisitos del usuario.

**Diferentes tipos de métricas**

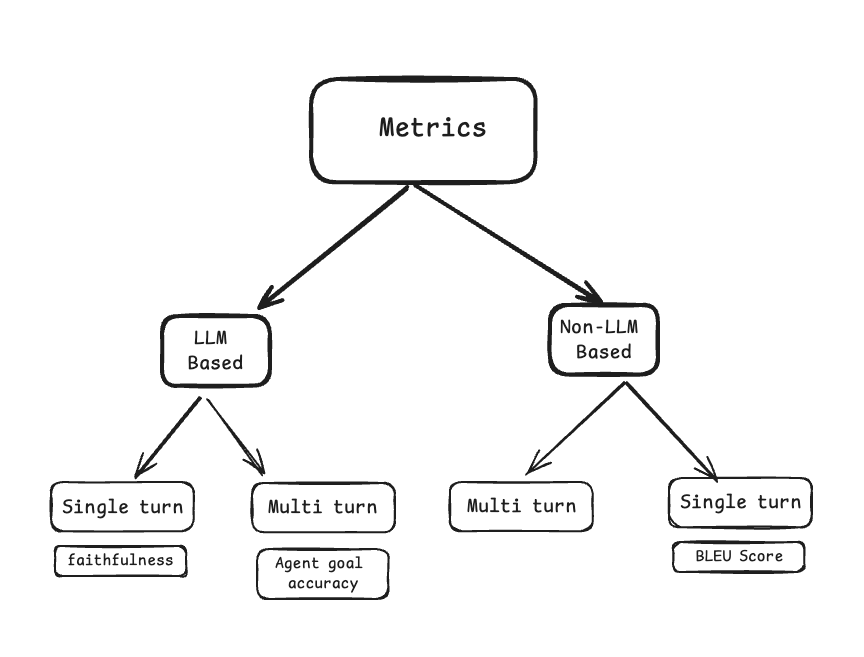


Figura 13: Árbol Métricas RAGAS

Las métricas se pueden clasificar en dos categorías según el mecanismo utilizado y según el tipo de datos que avalúan. En la Tabla 4 se puede ver esta clasificación:

|  |  |  |
| --- | --- | --- |
| **Clasificación** | **Métrica** | **Descripción** |
| Según mecanismo utilizado | **Basadas en LLM** | Utilizan LLM para realizar la evaluación. Puede ser necesario una o más llamadas al LLM para llegar a la puntuación o resultado. |
| **No Basadas en LLM** | Son deterministas y se pueden utilizar para evaluar el rendimiento de la aplicación de IA sin utilizar LLM. |
| Según tipo de dato a evaluar | **De un solo turno** | Evalúan el rendimiento de la aplicación de IA en función de un solo turno de interacción entre el usuario y la IA. |
| **Multi-turno** | Evalúan el rendimiento de la aplicación de IA en función de múltiples turnos de interacción entre el usuario y la IA. |

Tabla 4: Clasificación Métricas RAGAS

Lista de Métricas disponibles

RAGAS ofrece un conjunto de métricas de evaluación que se pueden utilizar para medir el rendimiento de su aplicación LLM. Estas métricas están diseñadas para ayudarlo a medir objetivamente el rendimiento de su aplicación. Hay métricas disponibles para diferentes aplicaciones y tareas, como flujos de trabajo RAG y Agentic.

Cada métrica es esencialmente un paradigma diseñado para evaluar un aspecto particular de la aplicación. Las métricas basadas en LLM pueden utilizar una o más llamadas LLM para llegar a la puntuación o al resultado. También se pueden modificar o escribir métricas propias utilizando ragas.

**Recuperación Generación Aumentada**

* Precisión del contexto *(Context Precision*). La precisión del contexto es una métrica que mide la proporción de fragmentos relevantes en el contexto.
* Recordatorio del contexto (*Context Recall*). El recuerdo de contexto mide cuántos documentos relevantes (o fragmentos de información) se recuperaron correctamente. Se centra en no omitir resultados importantes.
* Recordatorio de entidades de contexto (Context Entities Recall). La métrica proporciona la medida de recuperación del contexto recuperado, en función de la cantidad de entidades presentes.
* Sensibilidad al ruido (Noise Sensitivity). La metrica mide la frecuencia con la que un sistema comete errores al proporcionar respuestas incorrectas al utilizar documentos recuperados relevantes o irrelevantes.
* Relevancia de la respuesta (Response Relevancy). La métrica se centra en evaluar la pertinencia de la respuesta generada para la pregunta planteada. Se asigna una puntuación más baja a las respuestas que están incompletas o contienen información redundante y las puntuaciones más altas indican una mayor relevancia.
* Fidelidad (Faithfulness). La métrica mide la coherencia fáctica de la respuesta generada en relación con el contexto dado. Se calcula a partir de la respuesta y el contexto recuperado.
* Fidelidad multimodal (Multimodal Faithfulness). La métrica mide la coherencia fáctica de la respuesta generada en relación con el contexto visual y textual. Se calcula a partir de la respuesta, el contexto textual recuperado y el contexto visual.
* Relevancia multimodal (Multimodal Relevance). La métrica mide la relevancia de la respuesta generada en relación con el contexto visual y textual. Se calcula a partir de la entrada del usuario, la respuesta y los contextos recuperados (tanto visuales como textuales).

Existen otros tipos de medidas agrupadas por la funcionalidad que cubren:

* Casos de uso de agentes o herramientas
* Adherencia al tema
* Llamada de herramienta Precisión
* Precisión de los objetivos del agente
* Comparación de lenguaje natural
* Corrección fáctica
* Similitud semántica
* Similitud de cadenas no LLM
* Puntuación BLEU
* Puntuación de ROUGE
* Presencia de cuerdas
* Coincidencia exacta
* SQL
* Puntuación de Datacompy basada en ejecución
* Equivalencia de consultas SQL
* Propósito general
* Crítica de aspectos
* Puntuación de criterios simples
* Puntuación basada en rúbricas
* Puntuación de rúbricas específicas de cada instancia

Implementación de RAGAS para Evaluación de LLMs implica la creación de un conjunto de pruebas que contengan preguntas y respuestas generadas por el modelo. Estas respuestas se comparan contra un conjunto de respuestas esperadas usando las métricas mencionadas en los puntos anteriores.