



Universitat Oberta
de Catalunya

UNIVERSITAT OBERTA DE CATALUNYA (UOC)

MÁSTER UNIVERSITARIO EN CIENCIA DE DATOS (*DATA SCIENCE*)

TRABAJO FINAL DE MÁSTER

ÁREA 3

AgriChat: un copilot para la agricultura.

Automatización y mejora de los procesos agrícolas.

Autor: Enrique Villalobos Torregrosa

Tutor: José Luís Iglesias Allones

Profesora: Laia Subirats

Valencia, 4 de junio de 2024

Créditos/Licencia



Esta obra está sujeta a una licencia de Reconocimiento - NoComercial - SinObraDerivada
3.0 España de CreativeCommons.

Ficha

| | |
|-----------------------------------|----------------------------------|
| Título del trabajo: | AgriChat |
| Nombre del autor: | Enrique Villalobos Torregrosa |
| Nombre del colaborador/a docente: | Laia Subirats |
| Nombre del PRA: | José Luís Iglesias Allones |
| Fecha de entrega (mm/aaaa): | Valencia, 4 de junio de 2024 |
| Titulación o programa: | Máster en Ciencia de Datos. |
| Área del Trabajo Final: | El nombre de la asignatura de TF |
| Idioma del trabajo: | Español |
| Palabras clave | LLms, agricultura, RAG |

Dedicatoria

«*El éxito precisa ‘trabajo de hormiguita’: esfuerzo y sobre todo, perseverancia.*»

A toda mi familia y amigos, pero sobretodo a mis padres ya que, cuando la tormenta más oscura era con más luz brillaban.

Y por supuesto, a mí sobrino, que aun estando en el comienzo de su vida encuentre en la misma el valor intrínseco del esfuerzo y la perseverancia, único camino hacia el conocimiento.

Agradecimientos

Dar las gracias a mi tutor, José Luís Iglesias Allones, por su paciencia y compromiso con el proyecto. Las recomendaciones y los conocimientos transmitidos han sido la brújula que nos ha orientado durante todo el proyecto.

Dar la gracias también a la profesora colaboradora, Laia Subirats, por su diligencia en la dinamización de la asignatura en los foros.

Abstract

The present work is framed within the field of natural language processing. We have questioned how we can reduce the indiscriminate use of phytosanitary products using the power and scope of Large Language Models (LLMs) to improve the understanding of phytosanitary product usage by the agricultural community. Current environmental sustainability issues entail not only the contamination of our environmental heritage but also the proliferation or exacerbation of new diseases for humanity, while the agricultural community faces a continuous process of economic value loss and, consequently, incremental income degradation.

Our main objective has been reflected in the construction of a WebChat with RAG (Accelerating Retrieval-Augmented) techniques as a tool to aid and support decision-making regarding the use of phytosanitary products. To achieve this, a dataset has been compiled from official sources and used as a specific knowledge base for the model. The implementation of this tool aims to provide users, especially the agricultural community, with an interactive platform where they can obtain precise and contextualized information on the proper use of phytosanitary products, thus contributing to a more sustainable agriculture and environmental preservation.

The results obtained demonstrate the viability of the system. Unstructured information has been structured in tabular format, around 87 % of tables have been correctly identified and among them 60 % have been able to be structured in .csv format. Regarding the proposed RAG system, its general performance is around 0.56 out of 1 in its best answers and 0.31 in its worst answers.

Keywords: Natural Language Processing, RAG, LLMs

Resumen

El presente trabajo esta enmarcado dentro campo del procesamiento del lenguaje natural. Nos hemos preguntado cómo podemos reducir el uso indiscriminado de productos fitosanitarios usando la potencia y el alcance que tienen los Grandes Modelos del Lenguaje (LLMs) para la mejora de la comprensión del uso de los productos fitosanitarios por parte de la comunidad agrícola. Los actuales problemas de sostenibilidad medioambiental suponen no solo la contaminación de nuestro patrimonio medioambiental sino la proliferación o agravamiento de nuevas enfermedades para la humanidad al tiempo que la comunidad agraria se enfrenta a un proceso continuo de pérdida de valor económico y, por ende, a una degradación incremental de sus ingresos.

Nuestro objetivo principal se ha plasmado en la construcción de un WebChat con técnicas RAG (Accelerating Retrieval-Augmented) como herramienta de ayuda y apoyo a las decisiones de uso de los productos fitosanitarios. Para ello, se ha elaborado un conjunto de datos descargado de fuentes oficiales usandolo como base de conocimiento específica del modelo. La implementación de esta herramienta tiene como finalidad proporcionar a los usuarios, especialmente a la comunidad agrícola, una plataforma interactiva donde puedan obtener información precisa y contextualizada sobre el uso adecuado de productos fitosanitarios, contribuyendo así a una agricultura más sostenible y a la preservación del medio ambiente

Los resultados obtenidos demuestran la viabilidad del sistema. Se ha conseguido estructurar información no estructurada en formato tabular, alrededor del 87% de tablas se han identificado correctamente y entre ellas un 60% han podido ser estructuradas en formato .csv. En cuanto al sistema RAG planteado, su desempeño general está en torno al 0.56 de 1 en sus mejores respuestas y en 0.31 en su peores respuestas.

Palabras clave: Natural Language Processing, RAG, LLMs

Índice general

| | |
|--|----------|
| Créditos/Licencia | I |
| Ficha | II |
| Dedicatoria | III |
| Agradecimientos | IV |
| Abstract | V |
| Índice de Figuras | X |
| Índice de Tablas | XI |
| | |
| 1. Introducción | 1 |
| 1.1. Contextualización y justificación del proyecto. | 1 |
| 1.2. Objetivos del proyecto. | 4 |
| 1.2.1. Objetivos generales. | 4 |
| 1.2.2. Objetivos específicos. | 5 |
| 1.3. Metodología. | 6 |
| 1.4. Plan de investigación. | 7 |
| | |
| 2. Estado del Arte | 9 |
| 2.1. Generaciones de Modelos de Lenguaje. | 9 |
| 2.2. Modelos de Lenguaje Preentrenados (LLMs). | 12 |
| 2.2.1. ¿Qué es un LLMs? | 12 |

| | |
|---|-----------|
| 2.2.2. Revisión de los LLMs más relevantes. | 14 |
| 2.2.3. Tareas de los LLMs en NLP. | 15 |
| 2.2.4. Métodos de evaluación del desempeño. | 18 |
| 2.3. Técnicas de Ajuste del Modelo en el Procesamiento del Lenguaje Natural . . | 21 |
| 2.3.1. Introducción. | 21 |
| 2.3.2. Transfer-Learning en LLMs. | 22 |
| 2.3.3. Fine-Tuning de LLMs. | 24 |
| 2.4. RAG (Accelerating Retrieval-Augmented). | 27 |
| 2.4.1. El problema de las alucinaciones en LLMs. | 27 |
| 2.4.2. ¿Qué es RAG?, funcionamiento, tipos y arquitectura. | 28 |
| 3. Diseño e implementación. | 35 |
| 3.1. Ajuste de Table Transformer. | 35 |
| 3.1.1. Investigación previa | 35 |
| 3.1.2. Creación del conjunto de datos | 37 |
| 3.1.3. Ajuste del modelo | 38 |
| 3.1.4. Proceso de extracción de información tabular. | 41 |
| 3.2. Construcción del sistema RAG. | 43 |
| 3.2.1. Arquitectura del modelo RAG. | 43 |
| 4. Resultados y líneas de trabajo futuras. | 47 |
| 4.1. Resultados | 47 |
| 4.2. Líneas de trabajo futuras. | 51 |
| Bibliografía | 53 |

Índice de figuras

| | | |
|-------|--|----|
| 1.1. | Evolución modelos LLMs | 2 |
| 2.1. | Arquitectura Transformer en <i>Attention Is All You Need</i> | 13 |
| 2.2. | Visión General de la evolución de los LLMs | 15 |
| 2.3. | Extracción de información de PDFs | 18 |
| 2.4. | BenchMark Hugging Face | 20 |
| 2.5. | BenchMark SuperGLUE | 20 |
| 2.6. | Métricas en evaluación automatizada. | 21 |
| 2.7. | Métricas en evaluación con feedback humano. | 21 |
| 2.8. | Categorización enfoques Transfer-Learning. | 22 |
| 2.9. | Taxonomía PEFT. | 25 |
| 2.10. | Crecimiento de técnicas PEFT. | 26 |
| 2.11. | Métodos de optimización LLMs. | 29 |
| 2.12. | Naive RAG. | 30 |
| 2.13. | Advanced RAG. | 32 |
| 2.14. | Modular RAG. | 33 |
| 3.1. | Relación datos disponibles. | 36 |
| 3.2. | Arquitectura Table Transformer. | 36 |
| 3.3. | Tablas anotadas. | 38 |

| | | |
|------|---|----|
| 3.4. | Tablas totalmente anotadas. | 38 |
| 3.5. | Pipeline para la extracción de información. | 42 |
| 3.6. | Clasificación de tipos de tabla. | 43 |
| 3.7. | Arquitectura RAG empleada. | 44 |

Índice de tablas

| | | |
|------|--|----|
| 1.1. | Temporización del trabajo | 8 |
| 2.1. | LLms open source | 15 |
| 3.1. | Hiperparámetros detección tablas | 39 |
| 3.2. | Hiperparámetros detección de la estructura de tablas | 39 |
| 4.1. | Tablas identificadas. | 47 |
| 4.2. | Reconocimiento estructura de Tablas tipo A | 48 |

Capítulo 1

Introducción

1.1. Contextualización y justificación del proyecto.

No hay duda que los avances en el diseño e implantación de la Inteligencia Artificial (IA) ha supuesto todo un reto para las sociedades modernas técnificadas. Desde que en 1943 con el trabajo seminal de Mcculloch, Warren and Pitts, Walter [23] y su modelización matemática de una red neuronal hasta hoy en día la IA ha demostrado, aunque con cierta variabilidad, su capacidad disruptiva produciendo una transformación profunda de todos los sistemas sociales (jurídico, económico, cultural..) y por ende ha definido una nueva forma de hacer las cosas y de relacionarse con nuestro entorno.

Hoy en día, nos encontramos en sociedades altamente conectadas o también denominadas **sociedad red** inmersas en procesos de transformación digital donde cada vez hay mejores redes de comunicación y posibilidad de cálculo. Paradigmas como el Internet of Things (IoT), las smart-cities, la agricultura de precisión junto a los nuevos valores de calidad medioambiental están actuando como catalizadores para la implantación de la IA en toda la sociedad.

Cuando hablamos de IA lo hacemos de forma general, como si fuese una unidad estructurada lo que limita muchas veces el poder centrarse en aspectos más profundos. Sabemos que el propósito de la IA en general es emular las capacidades humanas de razonamiento mediante la adquisición de datos por distintas vías y su posterior procesamiento pero lo que se nos escapa es que al igual que a un humano hay que enseñarle a hablar, escribir y a razonar la IA también debe de aprender estas facultades.

Dentro de la IA en general nos encontramos con el campo del *Natural Language Processing* (NLP) cuya actividad científica está orientada a que los modelos de IA aprendan a comunicarse y razonar como humanos. El campo del NLP ha pasado por diversas transformaciones, en un principio tendríamos el uso de *Redes Neuronales Recurrentes* (RNN) usadas en los modelos *Long Short-term memory* (LSTM) el cual fue desarrollado por Hochreiter [13] hasta la llegada del modelo *Transformer* [38] en 2017 lo que ha supuesto un avance significativo en el campo del NLP.

Nos encontramos en una edad dorada para el progreso del NLP debido principalmente a un incremento de la inversión económica, tanto pública como privada y al incremento en la investigación y desarrollo de modelos cada vez más grandes al amparo de la arquitectura *Transformer* [38].

Tal y como refleja el siguiente gráfico extraido de *Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond* [48] vemos como la arquitectura *Transformer* ha potenciado la mejora continua de los LLMs.

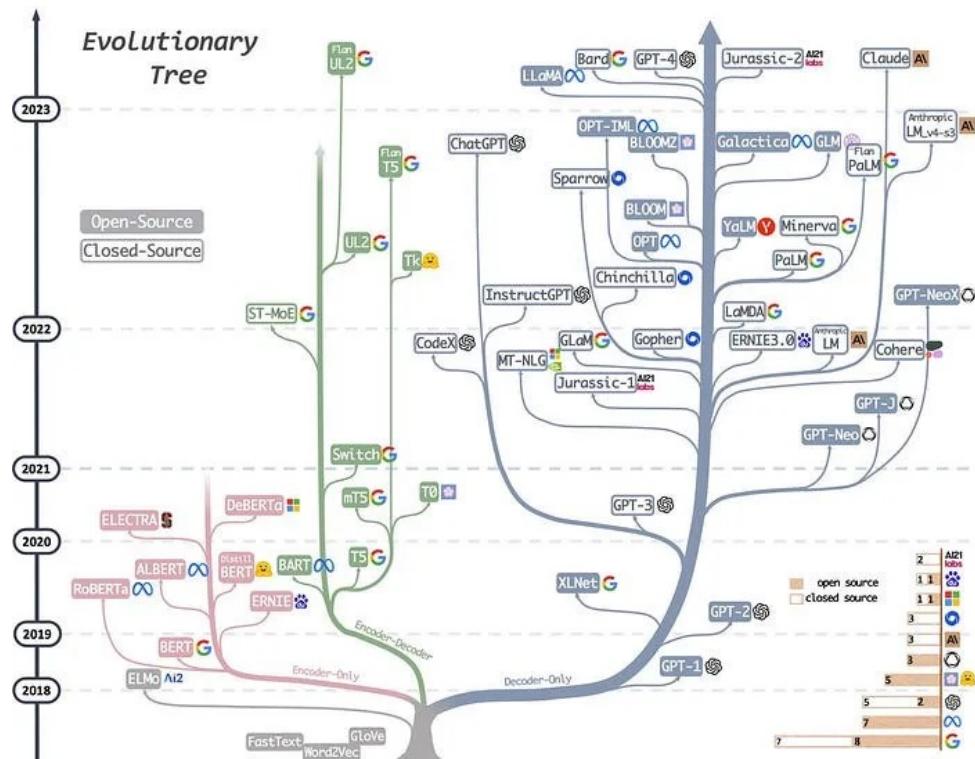


Figura 1.1: Evolución modelos LLMs

Por otra parte, el cambio en los valores medioambientales ha supuesto un nuevo campo de

actuación para la IA en temas relacionados con la gestión y uso de productos fitosanitarios. Durante la Segunda Guerra Mundial el continente Europeo quedó devastado en materia de calidad y producción alimentaria y una vez finalizada la misma se apostó por un modelo de agricultura intensivo en el uso de productos fitosanitarios con el fin de incrementar la producción de alimentos sacrificando la calidad de los mismos por su cantidad. Son tiempos donde se minimizan los efectos que tiene sobre el medioambiente y la salud el uso indiscriminado de los mismos.

Hoy en día, el uso de los productos fitosanitarios no es baladí y un uso responsable de los mismos es una pieza fundamental para la consecución de los principios de sostenibilidad medioambiental definidos en la Directiva de uso sostenible de los productos fitosanitarios aprobado por la Unión Europea (*Directiva 2009/128/CE del Parlamento Europeo y del Consejo, de 21 de octubre de 2009*).

Al amparo de la expansión y mejora de los grandes modelos de lenguaje (LLMs) ha surgido todo un ecosistema integrado de soluciones que cubren todas las fases para la creación de modelos, desde su diseño hasta la implementación. En el ámbito del desarrollo encontramos frameworks como LLamaIndex¹ o LangChain² que han permitido acelerar los procesos de mejora e implementación de modelos de lenguaje adaptados a tareas específicas así como el impulso del movimiento open-source para democratizar el acceso a los modelos de lenguaje (GPT-2, BERT, XLNet, T5, DistilledBERT, ROBERTA...), la creación de hubs de investigadores en torno al NLP y la construcción de comunidades de NLP como HuggingFace³.

El entrenamiento de un modelo de lenguaje tipo LLMs implica el uso de muchos recursos computacionales, se trata de hubs de ordenadores con hardware muy especializado, lo que implica una ingente cantidad de recursos económicos casi prohibitivos y al alcance de pocos agentes. Sin embargo, se han desarrollado nuevas técnicas como FineTuning [14], LORA [16], QLORA [7], RAG [49] que evitan reentrenar todos los billones de parámetros del modelo base (LLMs) mejorando su precisión en dominios específicos y dando acceso a

¹<https://www.llamaindex.ai/>

²<https://www.langchain.com/>

³<https://huggingface.co/>

la experimentación. En este trabajo se explotará la técnica de RAG.

Desde esta ventana de oportunidades que estamos viviendo este trabajo se centrará en explorar las capacidades que tienen los LLMs en su interacción con archivos PDFs que contienen información relativa a los productos fitosanitarios. Se buscará crear una aplicación web tipo Chat donde los usuarios podrán subir sus documentos e interactuar con ellos mediante el Chat.

La motivación del trabajo actual se origina al saber que uno de los factores que incide en el mal uso de los productos fitosanitarios es la falta de comprensión de las indicaciones de uso y de la regulación adyacente, es por ello, que pensamos que un sistema de Chat con información útil y veraz sobre información relativa por ejemplo de los detalles del producto, cultivos que puede tratar, plagas que tratan y otra series de cualidades del mismo sería una buena herramienta tecnológica en manos del campo de la agricultura profesional que reduciría el uso indiscriminado de productos fitosanitarios, mejorando la gestión de los mismos y por ende el rendimiento económico de la explotación dado el peso que los productos fitosanitarios tienen en la estructura de costes de las explotaciones agrarias. Consideramos que un buen uso de la herramienta puede suponer una ventaja competitiva para la agricultura en general y para las explotaciones agrícolas en particular.

1.2. Objetivos del proyecto.

1.2.1. Objetivos generales.

Tal y como hemos definido en la contextualización, el principal objetivo será el diseño y despliegue de una aplicación web de tipo Chat con Inteligencia Artificial (IA) donde la comunidad profesional agrícola, estudiantes e investigadores podrán cargar al sistema de chat un conjunto de PDFs que contienen información sobre productos fitosanitarios en diversos formatos e idiomas para poder interactuar con los mismos mediante el uso del lenguaje natural.

1.2.2. Objetivos específicos.

Para la consecución del objetivo general será necesario la consecución de hitos específicos, como son:

1. Los grandes modelos de lenguaje (LLMs) tienen un ámbito de actuación generalista y acotado en el tiempo (entrenamiento). Dado que nuestro objetivo principal pertenece a un dominio específico (el uso de fitosanitarios en la agricultura) un primer hito es **la captación, estudio y elaboración de un conjunto de datos para el uso de técnicas RAG** (Accelerating Retrieval-Augmented) [49]. Para ello, se realizará un estudio de la web del Ministerio de Agricultura, Pesca y Alimentación para extraer datos relevantes para el proyecto.
2. Dada la diversidad de modelos preentrenados el segundo hito será **la búsqueda y evaluación de diferentes modelos LLMs** que permitan interactuar con PDFs. Tras su selección, éste será el modelo base. En base a la metodología Transformer [38] buscaremos modelos de lenguaje open-source y estudiaremos sus características. Tras la selección los evaluaremos mediante pruebas usando Notebook Jupyter.
3. Ensamblar un **modelo de lenguaje usando el framework LLamaIndex**. Esta tarea incluye desde la ingesta de datos y la pregunta a responder hasta la creación de base de datos vectoriales, ventanas de contexto y respuesta del modelo al usuario. Se trabajará como prioridad el carácter de la seguridad de los datos, dando en todo momento las referencias que se han usado para elaborar la respuesta.
4. **Desarrollo e implementación de una AppWeb** como herramienta de apoyo a la toma de decisiones ofreciendo accesibilidad mediante un navegador o móvil a la comunidad agraria, gestores, investigadores y estudiantes así como la posibilidad de automatización, en el sentido que mejora los procesos de toma de decisiones al reducir la carga de trabajo manual que implica la lectura y comprensión de la documentación asociada a los usos de los productos fitosanitarios.
5. **Evaluación del rendimiento de la arquitectura RAG:** para ello se usará el

framework RAGAS⁴

1.3. Metodología.

Dado el carácter profesionalizador de este trabajo y una vez focalizado el tema de investigación mediante la revisión de la literatura se procederá a emplear una metodología ágil para definir los entregables propuestos en base a las siguientes fases de carácter cíclico:

1. **Requerimientos del proyecto:** se definirán con claridad las características de la app Web (funcionalidades) y los casos de uso (cómo se usará). Se consideran entregables el prototipado web y el esquema de los casos de uso.
2. **Adquisición de datos:** una vez que reducimos la ambigüedad de los requerimientos y los traducimos a casos de uso nos centraremos en la construcción de un conjunto de datos que se alinearán con los casos de uso definidos. Se tratará de la búsqueda y selección de documentos PDFs que serán preprocesados para obtener información relevante. Se estudiará la técnica RAG así como los desafíos y oportunidades que ofrece a nuestro proyecto. Se considera entregable el propio conjunto de datos preprocesado usando el framework Distilabel⁵ y Argilla⁶ para verificar y mejorar la calidad del conjunto de datos.
3. **Selección del modelo de lenguaje (LLMs) más adecuado.** Se búscara y seleccionará aquellos modelos de lenguaje (LLMs) que mejor se adapte al dominio agrícola, específicamente al uso de productos fitosanitarios. Se busca realizar una comparativa entre modelos y se propone como entregable un Notebook Jupyter con la evaluación de los mismos.
4. **Diseño del modelo LLMs mediante el framework LLamaIndex.** se elaborará todo el flujo necesario para ajustar el modelo base elegido en el punto anterior aplicando técnicas RAG al proceso de inferencia. Se propone como entregable un

⁴ <https://docs.ragas.io/en/stable/concepts/index.html>

⁵<https://distilabel.argilla.io/latest/>

⁶<https://argilla.io/>

Jupyter Notebook donde acepte como input preguntas y el outputs sea las respuestas del modelo. Se usará como IDE Visual Studio Code.

5. **Desarrollo e implementación de AgriChat:** se valorará el uso de frameworks como Flask o Streamlit para la implementacion de las funcionalidades y los casos de uso. Como mínimo la app Web AgriChat podrá responder a preguntas específicas sobre los PDFs sobre los que se les pregunte y dará la posibilidad de extraer información no estructurada (tablas en el pdf) en un formato estructurado como Json o Csv. Se propone como entregable un proyecto web en local o desplegado en la nube con el modelo diseñado en funcionamiento. Se trabajará con el IDE Visual Studio Code.
6. **Evaluación del modelo desarrollado.** Dado el crecimiento cuantitativo de los modelos LLMs se han desarrollado pruebas estandarizadas para su evaluación, en este trabajo aplicaremos el framework RAGAS para la evaluación de toda la arquitectura RAG planteada.

1.4. Plan de investigación.

Para la planificación temporal del trabajo hemos tenido en cuenta la temporización del plan docente de la asignatura el cual está dividido en 5 módulos temporales. Cada módulo define un espacio temporal y es ahí donde nosotros introduciremos todas las tareas propuestas que emanan de los objetivos definidos en el apartado anterior.

El proyecto tiene una duración de aproximadamente 114 días de trabajo con una jornada laboral de Lunes a Viernes con unas 25 horas semanales. Tal y como se observa en la tabla de temporalización el 48 % del trabajo tendrá un carácter empírico, el 30 % se invertirá en investigación y el resto, un 22 % a tareas de redacción y creación de entregables.

Tabla 1.1: Temporización del trabajo

| Esquema | Días |
|---|------|
| Definición y planificación (28/02 a 12/03) | |
| Definición y justificación | 3 |
| Motivación | 2 |
| Definición de objetivos | 3 |
| Temporalización | 3 |
| Estado del arte (13/03 a 26/03) | |
| Busqueda/selección | 10 |
| Redacción | 4 |
| Diseño e implementación (27/03 a 21/05) | |
| Definición de caso de uso | 2 |
| Estudio de fuentes de datos | 2 |
| Construcción del dataset | 5 |
| Investigación y selección LLMs | 10 |
| Diseño customizado del LLMs | 21 |
| Desarrollo e implementación Web | 12 |
| Evaluación de modelos | 3 |
| Redacción de la memoria (22/05 a 18/06) | |
| Desarrollo de la memoria | 6 |
| Interpretación y evaluación de datos | 10 |
| Presentación en formato audiovisual | 12 |
| Defensa del proyecto (19/06 a 07/07) | |
| Presentación-resumen del trabajo | 4 |
| Aprendizaje Blackboard Collaborate | 1 |
| Defensa del trabajo | 1 |

¹ La duración se muestra en días.

Capítulo 2

Estado del Arte

2.1. Generaciones de Modelos de Lenguaje.

La inteligencia artificial como disciplina científica está sometida a procesos de cambio a través de la refutación de sus teorías y mejora continua de sus paradigmas en base a la evidencia obtenida de la experiencia y la experimentación, este hecho, común en las disciplinas científicas, crea estadios a través de los cuales se puede observar su evolución. En el área del NLP, podemos establecer una evolución temporal que nos enseña el esfuerzo colectivo por hacer que las máquinas aprendan a comunicarse y hasta cierto punto, razonar como la especie humana.

1. *Statistical Language Models (SLM)*: se trata de un estadio seminal donde se busca modelos estadísticos para predecir la siguiente palabra dada una secuencia de texto, se centrán en el cálculo de probabilidades de ocurrencia, concepto que emerge de la Teoría de la Información y los avances en los estudios de probabilidad condicionada. Encontramos trabajos seminales teóricos *A mathematical theory of communication*[35] que sientan las bases teóricas para modelar el lenguaje. Fruto del conocimiento adquirido se elaboran los modelos N-gramas donde destacamos el artículo *A statistical approach to language translation*[4] que desarrolla un sistema de traducción basado en un modelo de N-gramas, así como el trabajo en la implementación de las cadenas de Markov con el trabajo de Jelinek, Frederick and Mercer, Robert L. [18] y su suavizado de Laplace (garantiza que ninguna probabilidad sea 0). Los modelos N-gramas, son pues un primer intento de modelización de lenguaje que presenta

problemas y limitaciones como son la explosión del espacio de búsqueda debido al crecimiento exponencial del tamaño del vocabulario y por ende del espacio de búsqueda o la escasez de datos (frecuencia casi 0 en la aparición de algunos n-gramas) que dan como resultado una modelización del lenguaje que no es capaz de capturar la diversidad y variabilidad del lenguaje natural.

2. *Neural Language Models (NLM)*: en esta etapa se pasa de un modelado de predicción de palabras a otro enfoque donde se espera que la máquina aprenda y capture representaciones sintácticas y semánticas relevantes del lenguaje mediante el uso de redes neuronales artificiales (ANN) de distinto tipo como son las recurrentes (RNN) y las convolucionales (CNN) con el fin de que las representaciones (vector denso o embedding) capture la relación temporal y espacial implícita en el corpus de entrenamiento. Al usar redes neuronales se amplian las tareas del NLP, ahora se traduce mejor, se genera texto más coherente y articulado, se resumen documentos, se crea una mini memoria para sistemas de preguntas-respuestas (Q&A), etc.. Uno de los trabajos pioneros que usan redes recurrentes es *Recurrent Neural Network based Language Model* [25] el cual demostró reducir un 50 % la perplejidad del modelo y dio paso a más investigación. Otro avance importante fue el propuesto en el artículo *Natural Language processing (almost) from scratch* [6] donde se propone un enfoque novedoso para en el procesamiento del lenguaje natural (NLP), ya que no se depende del conocimiento lingüístico previo muy condicionado por la tarea a resolver sino que, el modelo aprende de los datos disponibles de tal forma que el sistema o modelo es capaz de descubrir nuevos patrones y representaciones. Nacen así los modelos *from scratch* “desde cero” para señalar la no existencia de suposiciones predefinidas. Destacar por último el desarrollo del algoritmo word2vec publicado en *Distributed Representations of Words and Phrases and their Compositionality* [27] y su idea principal, el vector denso para la representación de palabras. Desde entonces, los vectores densos han capturado las relaciones semánticas y de significado mediante la similaridad (similaridad coseno) en el espacio vectorial donde viven dando lugar a poder realizar operaciones aritméticas con los mismos.

3. *Pre-trained language models (PLM)*: en esta nueva etapa, el foco de trabajo se centra en ir incrementando el corpus en tamaño y diversidad a la vez que se empiezan a entrenar a modelos para más de una tarea. Tenemos modelos como ELMo[29] (*Embeddings from Language Models*) entrenado en una red tipo LSTM (*Long Short Term Memory*) y BERT [8] (*Bidirectional Encoder Representation Transformer*) que ya hace uso de la arquitectura transformer[38]. Se empieza a construir modelos generalistas para posteriormente mejorar su desempeño con técnicas de Fine Tuning.
4. *Large Language Models (LLMs)*: estos modelos muestran comportamientos diferentes a los PLM más pequeños y muestran habilidades no vistas en la resolución de tareas complejas, conocidas como habilidades emergentes[43]. Estas son definidas como “*We consider an ability to be emergent if it is not present in smaller models but is present in larger models.*”[43] de tal forma que algo ocurre en la red neuronal durante su entrenamiento y que no solo depende del grado de escalamiento de la red neuronal o de la calidad de sus datos. Tres son las habilidades emergentes que se han descubierto que ocurren en los LLMs y no están presente en los PLN.
 1. *In-context Learning (ICL)*: se trata de una forma de entrenamiento basada en actualizar continuamente el modelo con nueva información del contexto en el que esta trabajando. De esta forma, el modelo se adapta rápidamente a cambios en el entorno manteniendo la relevancia de sus respuestas. Destacar el artículo *The Whole is Better than the Sum: Using Aggregated Demonstrations in In-Context Learning for Sequential Recommendation*[41] donde se hace uso del ICL para la creación de un sistema de recomendación.
 2. *Instruction Following*: se trata que un modelo comprenda las órdenes o solicitudes expresadas en lenguaje humano (voz, texto) y luego tome medidas en consecuencia para realizar las instrucciones que componen la tarea. Así pues, los LLMs pueden seguir las instrucciones para hacer tareas sin usar ejemplos explícitos de como se realiza la tarea lo que mejora su capacidad de generalización. En el artículo *Finetuned Language Models Are Zero-Shot Learners*[42] se experimentó con modelos con instrucciones y sin ellos observándose que LaMDA-PT,

ajustado con instrucciones, mejoró el desempeño del modelo en tareas no vistas en el entrenamiento al superar los 68 mil millones de parámetros.

3. *Step by Step Learning*: esta habilidad surge al descomponer una tarea compleja en subtareas más simples. Cada subtarea es un aprendizaje incremental y el modelo solo puede pasar a la siguiente cuando la ha hecho bien. Experimentos publicados en *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*[44] han demostrado mejoras en tareas de aritmética, sentido común y simbólico.

2.2. Modelos de Lenguaje Preentrenados (LLMs).

2.2.1. ¿Qué es un LLMs?.

En el apartado anterior hemos visto una pequeña evolución del NLP para generar lenguaje humano coherente y comprensivo que nos ha llevado hasta el concepto más actual, el LLMs (*Large Language Models*), siendo pues hora de conocerlos, para ello, presentaremos sus elementos fundamentales apoyándonos en la arquitectura más empleada, los *Transformers*[38].

Una vez recolectado el corpus, éste se procesa mediante una fase de **tokenización** para dividir el conjunto de datos en tokens, que son unidades significativas más pequeñas (palabras, subpalabras o caracteres) que el modelo procesará. Los esquemas de tokenización más usados suelen ser:

- *WordPiece*: presentado en el artículo *Neural Machine Translation of Rare Words with Subword Units*[34] y su concepto de *Subwords Units*. En este caso, el algoritmo divide el texto en unidades más pequeñas que más tarde fusiona con las palabras originales creando un vocabulario extendido que posteriormente se usará en la fase de entrenamiento e inferencia. En el trabajo de Mike Schuster y Kaisuke Nakajima[33] se uso esta técnica para la traducción entre el Japones y el Koreano.
- *Byte Pair Encoding (BPE)*: en este caso el BPE tiene un enfoque basado en la compresión del vocabulario al fusionar pares de bytes más frecuentes. Esta técnica

permite una mejor representación de la morfología y estructura del lenguaje.

- *UnigramLM*: presentado en el artículo de Taku Kudo[20] en 2018, se basa en un modelo Unígrafo que es capaz de generar múltiples segmentaciones de subpalabras con probabilidades “... which is capable of outputting multiple subword segmentations with probabilities”. Se trata de asignar probabilidades según la frecuencia de aparición en el corpus, entendiendo cada palabra como una unidad independiente.

La siguiente fase será el **embedding** que convertirá los tokens en representaciones vectoriales que captura las características semánticas y sintácticas de las palabras dentro de un espacio vectorial dado. En 2013 se publicó *Distributed Representations of Words and Phrases and their Compositionality*[26] presentando a Word2vec. En base al mismo, aparecieron nuevos embeddings como GLOVE[28] en 2014 y FastText[3] en 2017.

Una vez se han construido los embeddings, vamos a presentar sucintamente los elementos de la arquitectura transformer[38] tan importantes hoy en día en la construcción de los LLMs. Veamos su arquitectura en la siguiente imagen.

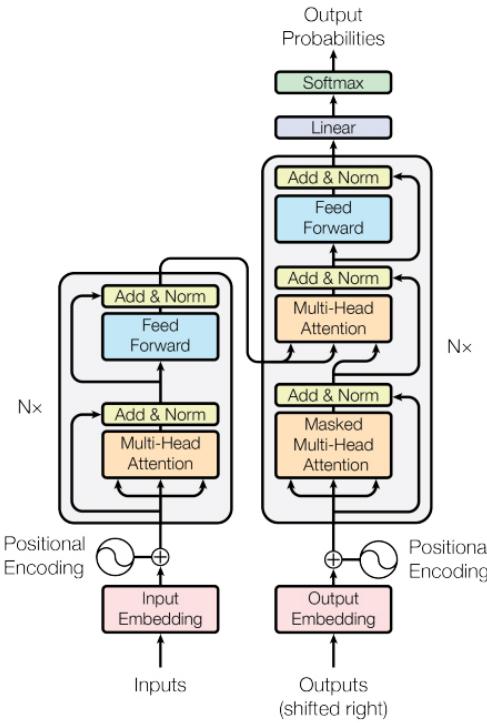


Figure 1: The Transformer - model architecture.

Figura 2.1: Arquitectura Transformer en *Attention Is All You Need*

Podemos observar que la primera pieza que encontramos es un **encoding positional**

cuyo papel es clave debido a que en una secuencia de texto la posición de las palabras es importante para capturar las relaciones espacio-temporales de las mismas. Los mecanismos de atención (Multi Head Attention) no son capaces de obtener información sobre el orden (temporal y espacial) en la secuencia de entrada, siendo el objetivo del encoding positional proporcionar dicha información a los mismos. Existen dos formas de asignar posicionamiento, de manera absoluta y de manera relativa con técnicas como ALiBi[30] (*Attention with Linear Biases*) y RoPE[36] (*Rotated Positional Encoding*).

Posteriormente tenemos dos componentes fundamentales, **el encoder y el decoder**. Primeramente el encoder producirá representaciones de alta calidad, lo que significa que los embeddings procesados en el mismo dispondrán de información semántica, sintáctica y espacio-temporal mediante mecanismos de atención para traspasar esas representaciones de alta calidad al decodificador el cual se encargará específicamente de predecir la probabilidad de ocurrencia de tokens sobre todo el diccionario con el cual el modelo ha sido entrenado.

Para la comprensión de la arquitectura *Transformer* es necesario una mayor explicación no sólo de todos sus componentes sino también de los procesos de transformación internos pero ese nivel de alcance queda fuera de la envergadura de éste proyecto. Sin embargo, hemos considerado importante mostrar sucintamente su arquitectura así como las demás fases necesarias para el diseño y modelización de los LLMs, ya que, éstas serán parte del proyecto.

2.2.2. Revisión de los LLMs más relevantes.

En los últimos años y sobre todo tras la aparición de la arquitectura *Transformer*[38] en 2017 han surgido un variado número de iniciativas encaminadas a generar LLMs tanto de ámbito abierto (open-source) como privativos basados en el pago de uso según los tokens enviados a la API del proveedor del modelo. Dado que uno de nuestros objetivos implica la búsqueda de modelos preentrenados de carácter abierto, en éstos vamos a centrar este apartado.

Para tener una visión general acerca del panorama actual en el despliegue de LLMs, vamos a comentar el timeline del artículo *A survey of Large Language Models*[50] que se

observa en la siguiente imagen.

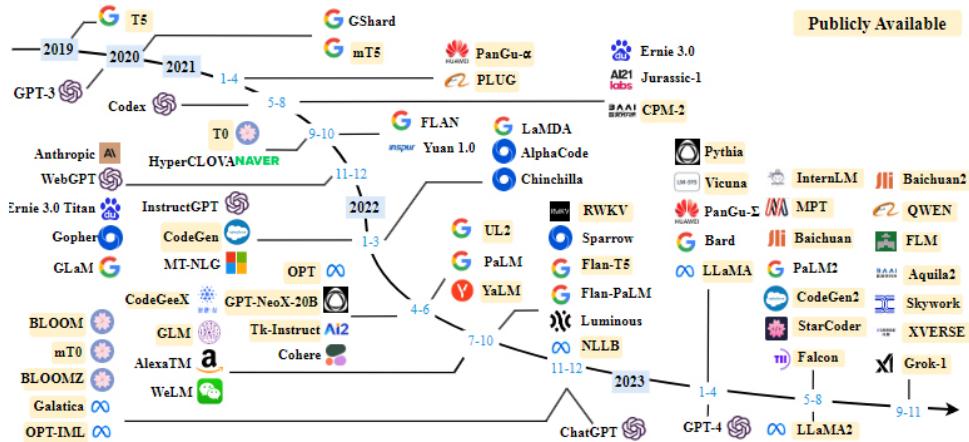


Figura 2.2: Visión General de la evolución de los LLMs

En términos generales se observa como cada vez más el área de los LLMs se mueve hacia modelos de tipo open-source y el uso más intensivo de los componentes encoder-decoder (Figura 1.1) de la arquitectura *Transformer*.

Dado el crecimiento y multiplicación de modelos LLMs, es difícil establecer una descripción pormenorizada del panorama actual, es por ello que vamos a presentar los modelos más recientes que son open-source. Para ello, se ha elaborado la siguiente tabla que resume las características de los modelos.

Tabla 2.1: LLms open source

| Modelo | Lanzamiento | Licencia | Compañía | Propósito | Num..of.Params | Tokens |
|-----------|-------------|------------|------------|-----------|----------------|--------|
| T5 | Oct-2019 | Apache 2.0 | Google | General | 11B | 1T |
| mT5 | Oct-2020 | Apache 2.0 | Google | General | 13B | 1T |
| CodeGen | Mar-2022 | Apache 2.0 | SalesForce | Código | 16B | 577B |
| BLOOM | 2022 | Rail 1.0 | BigScience | General | 176B | 366B |
| Galactica | Nov-2022 | Apache 2.0 | Meta | Ciencia | 120B | 106B |
| LLama | Feb-2023 | – | Meta | General | 65B | 1.4T |
| LLama 2 | Jul-2023 | LLama 2.0 | Meta | General | 70B | 2T |

2.2.3. Tareas de los LLMs en NLP.

Los LLMs están pre-entrenado normalmente para desarrollar tareas muy generales donde la comprensión de la misma queda relegada a generar respuestas sintáctica y semánticamente coherentes. A medida que los LLMs han ido mejorándose se han descubierto

ciertas tareas donde destacan posibilitando todo un nuevo campo de estudio para la automatización de tareas. Vamos a destacar las siguientes tareas:

- **Tareas de traducción:** aunque la traducción fue de las primeras tareas, incluso antes de la aparición de la arquitectura *Transformer*, fue tras ésta cuando empezó a generar resultados más allá de los esperados. En 2019, Roee Aharoni, Melvin Johnson, Orhan Firat[1] demostraron no solo la potencia de los LLMs en tareas de traducción sino la posibilidad de traducir desde varios idiomas de origen a varios idiomas de destino. Posteriormente, en 2023 en el estudio *Multilingual Machine Translation with Large Language Models: Empirical Results and Analysis* [51] acentuan y validan la capacidad de los LLMs en tareas de traducción simultanea.
- **Tareas de Generación de Texto:** esta tarea consiste en crear secuencialmente texto coherente con respecto a la entrada proporcionada al modelo del lenguaje. Es una tarea muy transversal debido a su uso en diferentes tipos de aplicaciones y productos como el resumen de un texto, la redacción automática de texto, la generación de diálogos de ChatBots, etc... En 2019 se presenta GPT-2[31] que verifica que los LLMs ya no necesitan un aprendizaje supervisado, sino que el propio modelo empieza a aprender a predecir texto correctamente conforme se aumentan los parámetros del modelo, pasando de miles a miles de millones al tiempo que incrementando el corpus de pre-entrenamiento. En esa misma época se presentaría BERT[8] (*Bidirectional Encoder Representation Transformer*) que permite el tuneado del mismo mediante la modificación de sus capas superiores para una amplia gama de tareas que supongan una inferencia lingüistica. Sin embargo, BERT es un modelo bidireccional (acepta todo el contexto disponible en la entrada) que solo usa el encoder de la arquitectura *Transformer* por lo que está mejor optimizado para tareas de comprension de texto con la generacion de los llamados embeddings de alta calidad.
- **Tareas de Extracción y Clasificación de información:** esta es la tarea que emprenderemos en este proyecto. Se basa en la capacidad que tienen los LLMs para acceder a datos no estructurados y poder extraer conocimientos de los mismos. Son

casos donde el LLMs interactua con documentos sin estructura interna y debe de aprender a identificar las entidades y sus relaciones, así como los diversos tipos de objetos (tablas, imágenes, etc..) con el fin de responder a cuestiones planteadas en base a la información extraida. Tras el despliegue de las diversas técnicas de ajuste o alineación del modelo pre-entrenado a tareas más específicas este campo ha emergido con mucha potencia debido a su impacto en la estructura de conocimiento de las organizaciones, que les ha permitido automatizar tareas de clasificación y análisis de la misma. En tareas de clasificación destacamos el proyecto Deep Learning for Technical Document Classification[19] que demostró la capacidad de las RNN para automatizar los procesos de clasificación documental dentro de las organizaciones. Por otra parte, el avance en extracción de la información, sobretodo en formato PDF, ha mejorado considerablemente y capa vez esta adquiriendo mayor alcance, tal y como vemos en la siguiente tabla del artículo *A Benchmark of PDF Information Extraction Tools using a Multi-Task and Multi-Domain Evaluation Framework for Academic Documents*[24] que aunque se centre en extracción de información de textos académicos es una buena medida de la importancia de esta área.

Table 1: Publications on information extraction from PDF documents.

| Publication ¹ | Year | Task ² | Method | Training Dataset ³ |
|--------------------------|------|-------------------|-----------------------|---|
| Palermo [44] | 1999 | M, ToC | Rules | 100 documents |
| Klink [27] | 2000 | M | Rules | 979 pages |
| Giuffrida [18] | 2000 | M | Rules | 1,000 documents |
| Aiello [2] | 2002 | RO, Title | Rules | 1,000 pages |
| Mao [37] | 2004 | M | OCR, Rules | 309 documents |
| Peng [45] | 2004 | M, R | CRF | CORA (500 refs.) |
| Day [14] | 2007 | M, R | Template | 160,000 citations |
| Hetzner [23] | 2008 | R | HMM | CORA (500 refs.) |
| Councill [12] | 2008 | R | CRF | CORA (200 refs.), CiteSeer (200 refs.) |
| Lopez [36] | 2009 | B, M, R | CRF, DL | None |
| Cui [13] | 2010 | M | HMM | 400 documents |
| Ojokoh [42] | 2010 | M | HMM | CORA (500 refs.), ManCreat FLUX-CiM (300 refs.), |
| Kern [25] | 2012 | M | HMM | E-prints, Mendeley, PubMed (19K entries) |
| Bast [5] | 2013 | B, M, R | Rules | DBLP (690 docs.), PubMed (500 docs.) |
| Souza [53] | 2014 | M | CRF | 100 documents |
| Anzaroot [3] | 2014 | R | CRF | UMASS (1,800 refs.) |
| Vilnis [57] | 2015 | R | CRF | UMASS (1,800 refs.) |
| Tkaczyk [55] | 2015 | B, M, R | CRF, Rules, SVM | CiteSeer (4,000 refs.), CORA (500 refs.), GROTOAP, PMC (53K docs.) |
| Bhardwaj [7] | 2017 | R | FCN | 5,090 references |
| Rodrigues [49] | 2018 | R | BiLSTM | 40,000 references |
| Prasad [46] | 2018 | M, R | CRF, DL | FLUX-CiM (300 refs.), CiteSeer (4,000 refs.) |
| Jahongir [4] | 2018 | M | Rules | 10,000 documents |
| Torre [15] | 2018 | B, M | Rules | 300 documents |
| Rizvi [47] | 2020 | R | R-CNN | 40,000 references |
| Hashmi [22] | 2020 | M | Rules | 45 documents |
| Ahmed [1] | 2020 | M | Rules | 150 documents |
| Nikolaos [33] | 2021 | B, M, R | Attention, BiLSTM | 3,000 documents |

Figura 2.3: Extracción de información de PDFs

2.2.4. Métodos de evaluación del desempeño.

Toda construcción humana en cualquier contexto y ámbito de actuación en algún momento debe pasar por la prueba de la evaluación. Este hecho, no debería verse como un descrédito hacia el trabajo de la comunidad científica sino como el mejor instrumento teórico-práctico de separar la ciencia de la opinión y como forma de mejora continua del rendimiento. En el área de los LLMs existen varias formas de evaluar un modelo pero lo más importante radica en seleccionar el método de evaluación que mejor se adapte a la tarea a desempeñar y a las características de los datos.

Los LLMs pueden evaluarse de distintas formas como son:

- Lo bien que realizan las tareas (Task): se trata de evaluar al modelo en una variedad de tareas para observar su rendimiento e identificar zonas de mejora. Se trata de las tareas más conocidas como son:
 - Análisis de sentimiento y clasificación de texto donde estudios como *Holistic Evaluation of Language Models (HEML)*[5] demostraron que los LLMs presentan puntuaciones altas en la comprensión de la inclinación emotiva de los datos y en la clasificación de los documentos.
 - Inferencia del lenguaje natural (NLI): son tareas donde se propone una hipótesis y ésta es resultado de las premisas dadas al modelo. El modelo debe de ser capaz de desarrollar un pensamiento lógico. Estudios como *Can Large Language Models Capture Dissenting Human Voices?*[21] muestran resultados deficientes en tareas de NLI señalando un campo de mejora posible.
 - Conocimiento semántico: son tareas donde se espera que el modelo aprenda el lenguaje no solo a un alto nivel (conceptos, relaciones, interpretación de palabras,...) sino comprender el significado subyacente (lo que se insinua) y la intencionalidad del emisor. Zhengwei Tao[37] y otros autores señalan deficiencia en este tipos de tareas donde los LLMs siguen puntuando más bajo que los humanos. Técnicas como el RLHF (Reinforcement Learning Human Feedback) han dado buen resultado para enseñar a los LLMs a comprender el significado semántico.

Existen una multitud de tareas más que podrían mencionarse pero lo importante es comprender que una evaluación está muy estrechamente relacionada con las tareas a desarrollar por el LLM.

La siguiente área donde podemos evaluar los LLMs son con el uso de conjuntos de datos para probar el rendimiento de los modelos en diferentes tareas, por ejemplo con conjuntos de datos como GLUE[40] o su versión ampliada SuperGLUE[39]. Existe una multitud de

comparadores (Benchmark) estandarizados destacamos el proyecto de Hugging Face *Open LLM LeaderBoard* y las iniciativa *SuperGLUE*.

The screenshot shows the LLM Benchmark interface. At the top, there are several filter options: Average, ARC, Hellaswag, MMU, TruthfulQA, Winogrande, SSBM, Type, Architecture, Precision, Merged, Hub License, #Params(B), Hub, Model sha, Hide models, Private or deleted, Contains a merge/moerge, Flagged, and Mo. Below these are sections for Model types (pretrained, continuously pretrained, fine-tuned on domain-specific datasets), chat models (RLHF, DPO, IFT, ...), base merges and moerges, and precision (float16, bfloat16, fp16, fp4, GPTQ). There are also filters for Model sizes (in billions of parameters) ranging from -? to 70+. The main table lists models with columns for Average, ARC, Hellaswag, MMU, TruthfulQA, Winogrande, and SSBM. One model, "cloudyu/TomGrc_FusionNet_34Bx2_MoE_v0.1_DPO_116", is highlighted with an orange border.

| Model | Average | ARC | Hellaswag | MMU | TruthfulQA | Winogrande | SSBM |
|---|---------|-------|-----------|-------|------------|------------|-------|
| dav1dIn/Beex-72b-v0.5 | 81.22 | 79.79 | 91.15 | 77.95 | 74.5 | 87.85 | 76.12 |
| SF-Foundation/EIn-72b-v0.11 | 80.81 | 76.79 | 89.02 | 77.2 | 79.82 | 84.86 | 78.77 |
| SF-Foundation/EIn-72b-v0.13 | 80.79 | 76.19 | 89.44 | 77.07 | 77.82 | 84.93 | 79.3 |
| SF-Foundation/EIn-72b-v0.12 | 80.72 | 76.19 | 89.46 | 77.17 | 77.78 | 84.45 | 79.23 |
| abacusai/Swag-72B-v0.1 | 80.48 | 76.82 | 89.27 | 77.15 | 76.67 | 85.08 | 78.7 |
| libvlibv/alpaca-dragon-72b-v1 | 79.3 | 73.89 | 88.16 | 77.4 | 72.69 | 86.93 | 77.63 |
| mzech/MoE-72B-Loce-1.8.7-DPO | 78.55 | 70.82 | 85.96 | 77.13 | 74.71 | 84.86 | 78.62 |
| cloudyu/TomGrc_FusionNet_34Bx2_MoE_v0.1_DPO_116 | 77.91 | 74.06 | 86.74 | 76.65 | 72.24 | 83.35 | 74.45 |
| saltlux/Luxia-21.4b-alignment-v1.0 | 77.74 | 77.47 | 91.88 | 68.1 | 79.17 | 87.45 | 62.4 |
| cloudyu/TomGrc_FusionNet_34Bx2_MoE_v0.1_Full_linear_DPO | 77.52 | 74.86 | 86.67 | 76.69 | 71.32 | 83.43 | 72.93 |
| zhengz/HuTAO-7Bx2-MoE-v0.1 | 77.5 | 73.81 | 89.22 | 64.92 | 79.57 | 87.37 | 71.11 |
| yunconglong/Tzuthf01_DPO_TomGrc_FusionNet_7Bx2_MoE_13B | 77.44 | 74.91 | 89.3 | 64.67 | 78.82 | 88.24 | 69.52 |

Figura 2.4: BenchMark Hugging Face

The screenshot shows the SuperGLUE benchmark interface. At the top, there are links for GLUE and SuperGLUE, and a navigation bar with Paper, Code, Tasks, Leaderboard, FAQ, Diagnostics, Submit, and Login. The main area displays a table of models ranked by score. The columns include Rank Name, Model, URL Score, CoLA, SST-2, MRPC, STS-B, QQP, MNLI-m, MNLI-mm, QNLU, RTE, WNLI, and AX. The table lists 13 models, with the top one being "Turing ULR v6".

| Rank Name | Model | URL Score | CoLA | SST-2 | MRPC | STS-B | QQP | MNLI-m | MNLI-mm | QNLU | RTE | WNLI | AX | |
|-----------|-------------------------------------|----------------------|------|-------|------|-----------|-----------|-----------|---------|------|------|------|------|------|
| 1 | Microsoft Alexander v-team | Turing ULR v6 | 91.3 | 73.3 | 97.5 | 94.2/92.3 | 93.5/93.1 | 76.4/90.9 | 92.5 | 92.1 | 96.7 | 93.6 | 97.9 | 55.4 |
| 2 | JDExplore d-team | Vega v1 | 91.3 | 73.8 | 97.9 | 94.5/92.6 | 93.5/93.1 | 76.7/91.1 | 92.1 | 91.9 | 96.7 | 92.4 | 97.9 | 51.4 |
| 3 | Microsoft Alexander v-team | Turing NLR v6 | 91.2 | 72.6 | 97.6 | 93.8/91.7 | 93.7/93.3 | 76.4/91.1 | 92.6 | 92.4 | 97.9 | 94.1 | 95.9 | 57.0 |
| 4 | DIRL Team | DeBERTa + CLEVER | 91.1 | 74.7 | 97.6 | 93.3/91.1 | 93.4/93.1 | 76.5/91.0 | 92.1 | 91.8 | 96.7 | 93.2 | 96.6 | 53.3 |
| 5 | ERNIE Team - Baidu | ERNIE | 91.1 | 75.5 | 97.8 | 93.9/91.8 | 93.0/92.6 | 75.2/90.9 | 92.3 | 91.7 | 97.3 | 92.6 | 95.9 | 51.7 |
| 6 | Allcelind & DIRL | StructBERT + CLEVER | 91.0 | 75.3 | 97.7 | 93.9/91.9 | 93.5/93.1 | 75.6/90.8 | 91.7 | 91.5 | 97.4 | 92.5 | 95.2 | 49.1 |
| 7 | DeBERTa Team - Microsoft | DeBERTa / TuringNLv4 | 90.8 | 71.5 | 97.5 | 94.0/92.0 | 92.9/92.6 | 76.2/90.8 | 91.9 | 91.6 | 99.2 | 93.2 | 94.5 | 53.2 |
| 8 | HF-EVTEK | McALBERT + DkM | 90.7 | 74.0 | 97.0 | 94.5/92.6 | 93.8/92.5 | 74.7/90.6 | 91.3 | 91.1 | 97.8 | 92.0 | 94.5 | 52.6 |
| 9 | PING-AN Omni-Sistic | ALBERT + DAAF + NAS | 90.6 | 73.5 | 97.2 | 94.0/92.0 | 93.0/92.4 | 76.1/91.0 | 91.6 | 91.3 | 97.5 | 91.7 | 94.5 | 51.2 |
| 10 | T5 Team - Google | T5 | 90.3 | 71.6 | 97.5 | 92.8/90.4 | 93.3/92.6 | 75.1/90.6 | 92.2 | 91.9 | 96.5 | 92.8 | 94.5 | 53.1 |
| 11 | Microsoft D365 AI & MSR AI & GATECH | MT-DNN-SMART | 89.9 | 69.5 | 97.5 | 93.7/91.6 | 92.9/92.5 | 73.9/90.2 | 91.0 | 90.8 | 99.2 | 89.7 | 94.5 | 50.2 |
| 12 | Huawei Noah's Ark Lab | NEZHA-Large | 89.8 | 71.7 | 97.3 | 93.3/91.0 | 94.2/91.9 | 75.2/90.7 | 91.5 | 91.3 | 96.2 | 90.3 | 94.5 | 47.9 |
| 13 | LG AI Research | ANNA | 89.8 | 68.7 | 97.0 | 92.7/90.1 | 93.0/92.8 | 75.3/90.5 | 91.8 | 91.6 | 96.0 | 91.8 | 95.9 | 51.8 |

Figura 2.5: BenchMark SuperGLUE

Por último, la evaluación puede desarrollarse de forma automática o apoyado con intervención humana. Esta es un cuestión difícil y dependerá mucho de la naturaleza de la tarea a la que se enfrenta el LLMs como a los objetivos de evaluación que se hayan definido. En una evaluación automatizada se aplican métricas estandizadas, tal y como podemos ver en la siguiente tabla extraída de *A Survey on Evaluation of Large Language Models*[5].

En base al concepto que nos interesa medir tenemos métricas diversas. Por ejemplo, para la precisión del modelo (*Accuracy*) osea, cómo de bien el modelo está realizando la tarea tenemos métricas como Coincidencia Exacta (Exact Match), el F1 Score y ROUGE.

Table 9. Key metrics of automatic evaluation.

| General metrics | Metrics |
|-----------------|---|
| Accuracy | Exact match, Quasi-exact match, F1 score, ROUGE score [118] |
| Calibrations | Expected calibration error [60], Area under the curve [54] |
| Fairness | Demographic parity difference [242], Equalized odds difference [64] |
| Robustness | Attack success rate [203], Performance drop rate [264] |

Figura 2.6: Métricas en evaluación automatizada.

Cuando en la evaluación hay un feedback humano de expertos en la tarea el pre-entrenamiento del modelo recibe retroalimentación más completa y ajustada a la tarea real. La tabla siguiente muestra algunas métricas relevantes para una evaluación con feedback humano.

Table 10. Summary of key factors in human evaluation

| Evaluation Criteria | Key Factor |
|-----------------------------|--|
| Number of evaluators | Adequate representation [7], Statistical significance |
| Evaluation rubrics | Accuracy [178], Relevance [261], Fluency [196], Transparency, Safety [85], Human alignment |
| Evaluator's expertise level | Relevant domain expertise [144], Task familiarity, Methodological training |

Figura 2.7: Métricas en evaluación con feedback humano.

En definitiva, la evaluación de los LLMs es una mezcla de métricas tradicionales como la perplejidad que nos indica la confianza, en términos de probabilidad, que el modelo tiene en sus predicciones y otras técnicas más recientes como pueden ser BLEU, ROUGE, METEOR, etc...

2.3. Técnicas de Ajuste del Modelo en el Procesamiento del Lenguaje Natural

2.3.1. Introducción.

Tal y como hemos indicado anteriormente, para el pre-entrenamiento de un LLMs es necesario una gran cantidad de recursos de computación lo que limita su posibilidad a grandes institutos de investigación u organizaciones. Para alinear y mejorar el LLMs a tareas específicas se han desarrollado una serie de técnicas que evitan un pre-entrenamiento del modelo.

2.3.2. Transfer-Learning en LLMs.

Durante el pre-entrenamiento de un LLMs el modelo ajusta sus pesos según la función de pérdida. Estos pesos, que es el conocimiento adquirido por el modelo, tienen un carácter generalista, es decir, es un conocimiento común a muchas tareas donde se procese lenguaje natural. Este enfoque es útil cuando la tarea que intentamos resolver no tiene un dominio específico, ya que, entonces el modelo degrada su rendimiento debido a que maneja un conocimiento generalista.

La base del transfer-learning trata de transferir conocimiento de una tarea general a otra tarea que de alguna forma tiene relación con su base de conocimientos, sus pesos. En un proceso de transfer-learning siempre tendremos un dominio fuente (los pesos aprendidos) y un dominio objetivo. Aunque esta área se ha desarrollado muy rápidamente, vamos a intentar mostrar sus principales avances.

En la siguiente imagen se observa la categorización de *A Comprehensive Survey on Transfer Learning*[52] y los énfoques más empleados.

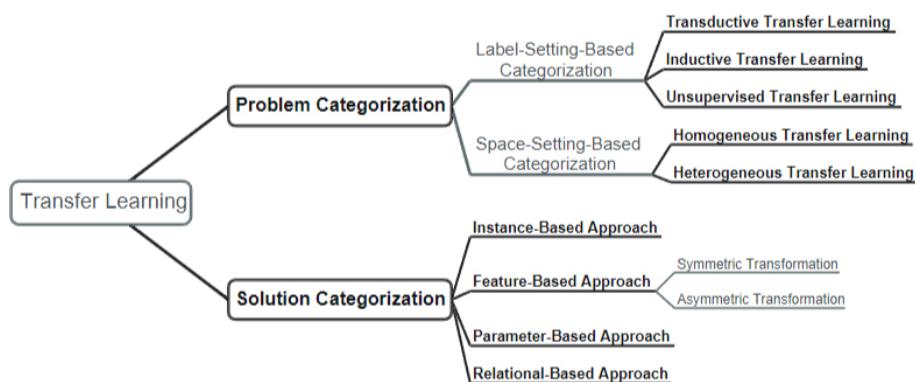


Figura 2.8: Categorización enfoques Transfer-Learning.

Vemos que la transferencia de aprendizaje se centra en:

- **Categorizando los problemas:** dependiendo de las etiquetas del conjunto de datos.

Se trataría de si proporcionamos al modelo un conjunto de datos etiquetado o no etiquetado, podemos ver 3 métodos:

- Transductive Transfer-Learning: la información de etiquetas solo proviene del dominio fuente y transfiere el conocimiento manteniendo una distribución de

datos parecida en ambos dominios, el fuente y el objetivo. Su objetivo es mejorar el rendimiento del modelo en un dominio específico y no tanto en su capacidad de generalización. Es útil cuando no se dispone de muchos datos en el dominio objetivo o cuando el conocimiento del dominio fuente está relacionado con el dominio objetivo.

- Inductive Transfer-Learning: en este caso se dispone de información de etiquetas para las instancias del dominio objetivo. El modelo utilizará los datos y las etiquetas del dominio fuente para aprender características generales y patrones que pueden ser útiles en el dominio objetivo. Se busca crear un modelo generalizado que pueda adaptarse diferentes dominios y tareas.
- Unsupervised Transfer-Learning: en estos casos no se dispone de información de etiquetas para ninguna de las instancias en los dominios fuente y objetivo. Este enfoque es útil cuando el etiquetado de datos en el dominio objetivo es caro o difícil de obtener, ya que permite aprovechar la información disponible en el dominio fuente para mejorar el rendimiento en el dominio objetivo sin requerir etiquetas adicionales.

Otro aspecto a tener en cuenta es la relación entre las características de entrada y las etiquetas de salida del LLMs. Se trataría de saber si son homogéneas (las características de entrada y las etiquetas de salida son consistentes entre el dominio fuente y el dominio objetivo) o heterogéneas (las características de entrada y las etiquetas de salida pueden ser diferentes entre los dominios).

- **Categorizando la solución:** las diferentes metodologías y enfoques que se usan en el Transfer-Learning.
 - *Instance-Based Approach:* se basa en utilizar ejemplos específicos o instancias del dominio fuente para mejorar el rendimiento en el dominio objetivo. En este enfoque, en lugar de transferir conocimientos en términos de modelos o representaciones, se transfieren ejemplos individuales o casos específicos que son relevantes para la tarea en el dominio objetivo. Este enfoque es útil cuando

encontramos instancias en el dominio fuente que coinciden con el dominio objetivo.

- *Feature-Based Approach*: se basa en transferir conocimientos en forma de características o atributos del dominio fuente al dominio objetivo. En este enfoque, en lugar de transferir ejemplos individuales o instancias, se extraen características relevantes del dominio fuente y se utilizan para mejorar el aprendizaje en el dominio objetivo.
- *Parameter-Based Approach*: se basa en transferir los parámetros aprendidos de un modelo en el dominio fuente a un modelo en el dominio objetivo, se transfieren los pesos de los parámetros del modelo preentrenado en el dominio fuente al modelo en el dominio objetivo. Posteriormente con técnicas de fine-tunning se alinea el los pesos del modelo al dominio objetivo.
- *Relational-Based Approach*: se basa en la transferencia de conocimiento sobre las relaciones entre las entidades, como son las relaciones semánticas o estructurales relevantes para la tarea en el dominio objetivo.

2.3.3. Fine-Tuning de LLMs.

En el apartado anterior hemos visto como el transfer-learning tiene la capacidad de aprovechar el conocimiento adquirido de una tarea (dominio fuente) para mejorar el dominio en una nueva tarea (dominio objetivo). Ahora veamos como podemos ajustar mejor ambos dominios al entrenar adicionalmente con datos de la tarea específica un modelo pre-entrenado.

En un mundo donde la eficiencia y rendimiento de los LLMs están estrechamente relacionadas con los costes de explotación y la huella de carbono de los LLMs se han buscado nuevas técnicas para implementar modelos LLMs mejorando la eficiencia y rendimiento de los mismos.

Una técnica de ajuste fino fue propuesta por Jeremy Howard and Sebastian Ruder y conocida como ULMFiT[@howard2018universal] en 2018. Introdujeron la técnica *discriminative fine-tuning* que consiste en ajustar cada capa con tasas de aprendizaje diferentes

para mejorar el ajuste. Se trataría de asignar una tasa de aprendizaje específica a cada conjunto de parámetros (cada capa del modelo), lo que permite ajustar cada capa con mayor precisión. Ha demostrado ser efectiva en tareas o dominios típicos del NLP. Destacamos el trabajo de Dou Hou[15] y otros autores para detectar el lenguaje paternalista y condescendiente usando varios modelos pre-entrenados (BERT, ERNIE 2.0, ROBERTA y BERT-PLC).

En 2020 el artículo *Don't Stop Pretraining: Adapt Language Models to Domains and Tasks*[11] muestra las ventajas de los métodos de ajuste fino y como han mejorado la eficiencia en una multitud de tareas y dominios.

La práctica más común usada es PEFT (*Parameters Efficient Fine Tuning*)[12] que trata de ajustar un número mínimo de parámetros para conseguir un mejor rendimiento en tareas posteriores en comparación con el ajuste de todo el modelo pre-entrenado. En el artículo Parameter-Efficient Fine-Tuning for Large Models[12] se desarrolla el algoritmo PEFT, en él se presenta una taxonomía que nos ayuda a comprender su funcionamiento y la forma como aplicarlos.

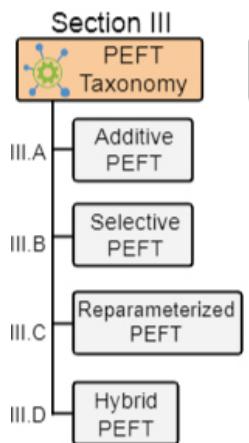


Figura 2.9: Taxonomía PEFT.

Vemos en la taxonomía:

- **PEFT Aditivo** : que modifica la arquitectura del modelo al introducir nuevos módulos o parámetros entrenables.
- **PEFT Selectivo**: donde solo un subconjunto de parámetros es entrenable durante el ajuste fino.

- **PEFT Reparametrizado:** reparametriza el modelo en un espacio de baja dimensionalidad con los parámetros originales del modelo para el entrenamiento, y luego lo transforma de manera equivalente para la inferencia.
- **PEFT Híbrido:** Combina diferentes métodos de PEFT.

En la siguiente imagen extraída de *Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment*[46] vemos como las técnicas PEFT han crecido en importancia y alcance al tiempo que los modelos de lenguaje incrementaban el número de parámetros.

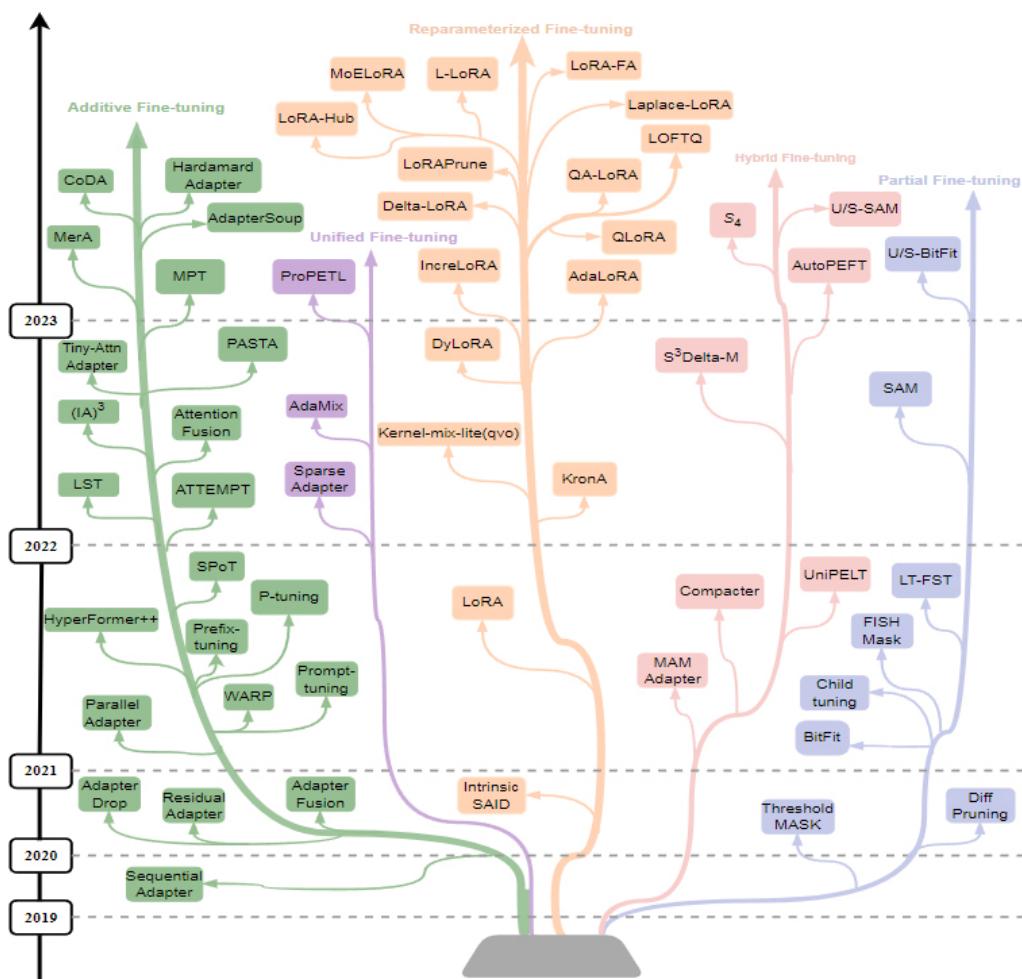


Figura 2.10: Crecimiento de técnicas PEFT.

Las técnicas de fine-tuning han visto mejorar su eficiencia sobre todo en el uso de la memoria necesaria en los procesos de fine-tuning mediante el uso de métodos como QLORA[7] donde se ha demostrado como se reduce drásticamente el uso de la memoria

necesaria para realizar fine-tuning. En el paper ajustan un modelo de 65B de parámetros en una GPU de 48GB.

2.4. RAG (Accelerating Retrieval-Augmented).

2.4.1. El problema de las alucinaciones en LLMs.

Desde la aparición de los LLMs han surgido, como hemos visto en los puntos anteriores, toda una serie de técnicas centradas en mejorar el desempeño de los LLMs a la hora de realizar nuevas tareas ajustando para ello el conocimiento del modelo. Pero los LLMs a pesar de los ajustes presentan ciertas deficiencias: fabrican hechos que no corresponden con la realidad y tienen lo que se ha llamado alucinaciones (*hallucinations*). En el estudio *A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions*[17] se describe toda una taxonomía sobre las alucinaciones. Los autores exponen la siguiente taxonomía:

- **Alucinaciones de Factualidad:** hacen referencia a la discrepancia entre la respuesta del modelo y los hechos verificables en el mundo real. Según los autores se manifiesta de dos formas:
 - *Inconsistencia Factual:* la salida del LLMs no es ficticia pero tiene contradicciones con el mundo real (datos erróneos de un evento histórico).
 - *Fabricación de hechos:* la salida del LLMs incluye hechos que no pueden verificarse en el mundo real.
- **Alucinaciones de Fidelidad:** se produce cuando el contenido generado por el LLM se desvía de las instrucciones del usuario o del contexto dado, lo que puede afectar a la utilidad y la fiabilidad de las respuestas generadas.

Diversos estudios ya han demostrado que las alucinaciones son un problema para los LLMs. En 2024 se publica *Hallucination is Inevitable: An Innate Limitation of Large Language Models*[47] donde se realizan la siguiente pregunta *can hallucination be completely*

eliminated? concluyendo que las alucinaciones siempre estarán implícitas en los LLMs debido a la imposibilidad de que los modelos aprendan todas las funciones computables y por tanto siempre alucinarán. Una de las técnicas que se han propuesto para reducir las alucinaciones ha sido RAG.

2.4.2. ¿Qué es RAG?, funcionamiento, tipos y arquitectura.

En 2020 se publica *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*[22] donde se propone una nueva metodología llamada RAG especialmente destinada a tareas de NLP intensivas en conocimiento. El flujo de trabajo de RAG se puede resumir en:

1. Recuperar información relevante de un vasto corpus de documentos.
2. Utilizar la información recuperada para generar respuestas o texto, mejorando así la calidad de las predicciones.

El funcionamiento de este método de trabajo evita que haya que reentrenar todo el LLMs para cada tarea específica, sino que se adjunta una base de conocimientos actualizada para mejorar la calidad de las respuestas del modelo. Se distingue dos etapas:

1. **Fase de codificación:** se usan modelos de codificación para recuperar documentos relevantes de la base de conocimientos basados en preguntas, como BM25, DPR, ColBERT.
2. **Fase de Generación:** Utilizando el contexto recuperado como condición, éste es enviado al LLMs el cual generará las inferencias.

El uso de RAG siempre está asociado con tareas intensivas en conocimiento, sobretodo actualizado, y no es una técnica para la comprensión de dominios grandes, como puede ser la traducción de texto, formatos o estilos de los mismos.

RAG se concibe como una técnica de optimización de LLMs al igual que lo es el fine-tunning o el prompt engineer. En la siguiente gráfica[10] vemos donde se posiciona RAG con

respecto a dos variables, el conocimiento externo requerido y la adaptación del modelo requerida.

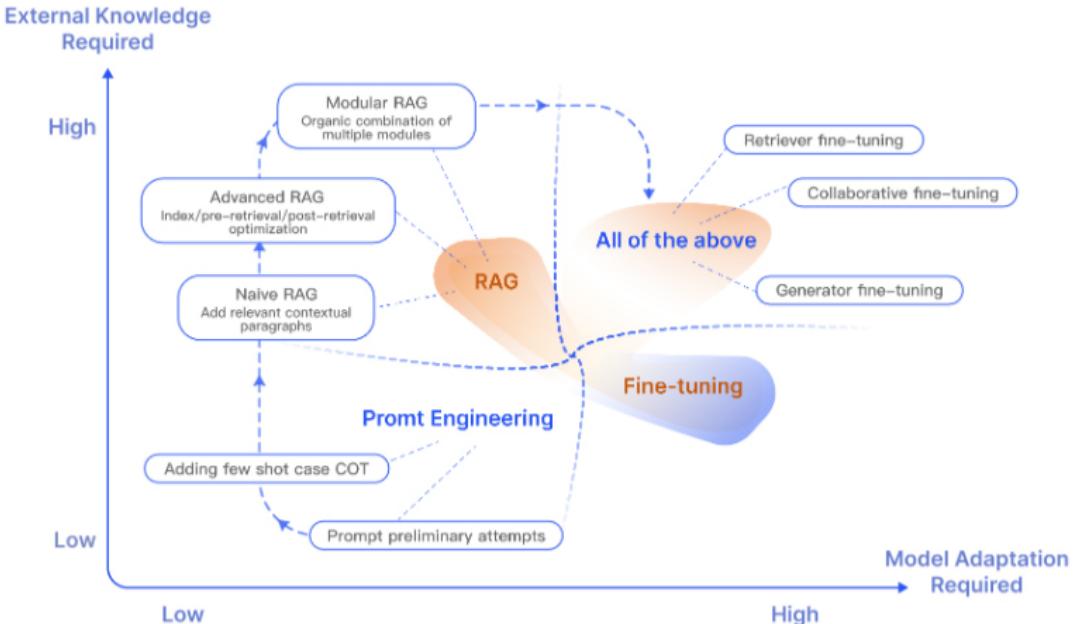


Figure 2: RAG compared with other model optimization methods

Figura 2.11: Métodos de optimización LLMs.

Podemos observar como la técnica de RAG se sitúa en el cuadrante superior izquierdo que son aquellas tareas más demandantes de conocimientos y menos de adaptación del modelo. También podemos observar como la técnica RAG tiene distintas formas de implementarse, siendo estas el RAG Básico, el RAG Avanzado y el RAG modular. Veamos sucintamente su arquitectura y características.

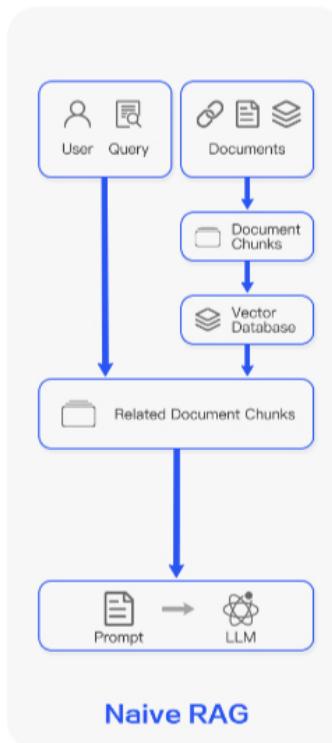


Figura 2.12: Naive RAG.

El Naive RAG o RAG básico esta compuesto por tres elementos:

- **Un indexador de información:** cuya función es buscar y recuperar información indexada, normalmente mediante la función de similaridad coseno. Tiene 3 etapas:
 1. **Indexación de datos:** se limpian y extraen los datos originales, y se convierten en texto plano diferentes formatos de archivo (PDF, HTML), ya que es imprescindible que la información este en un formato legible y procesable para su posterior indexación.
 2. **División en fragmentos (Chunking):** Consiste en dividir el texto cargado en fragmentos más pequeños, ya que los LLMs tienen limitaciones en la cantidad de contexto que pueden manejar (ventana de contexto).
 3. **Embedding y creación de índice:** el texto se codifica en vectores que se utilizan en el proceso de recuperación para calcular la similitud entre el vector y el vector de la consulta. Posteriormente se crea un índice que almacena los fragmentos originales del corpus y los embeddings en forma de pares clave-valor para facilitar búsquedas rápidas y frecuentes en el futuro.

- **Un proceso de recuperación (*Retriever*):** sirve para obtener la información relevante necesaria para responder a la consulta del usuario y consta de 3 fases:
 1. **Entrada del usuario:** al ingresar una consulta o pregunta, se utiliza el mismo modelo de codificación (embedding) que en la primera etapa para convertir la consulta en un vector. Este vector representa la representación numérica de la consulta en un espacio semántico.
 2. **Cálculo de similitud:** se calcula la similitud entre el embedding de la pregunta y los embeddings de los bloques de documentos en el corpus.
 3. **Selección de los mejores documentos:** Basándose en el nivel de similitud calculado, se eligen los K bloques de documentos principales que se consideran como la información de contexto ampliada que se enviará en junto con la consulta al LLMs.
- **Un proceso de Generacion (*Generation*):** se trata de crear respuestas coherentes y relevantes a partir de la información recuperada en las etapas anteriores. Consta de 4 fases:
 1. **Combinación de la pregunta y documentos relacionados:** La pregunta formulada por el usuario y los documentos relevantes recuperados en la etapa de recuperación se combinan para formar un nuevo prompt que actua como un contexto ampliado.
 2. **Generación de respuestas:** El LLMs se encarga de responder a la pregunta basándose en la información proporcionada en el prompt.
 3. **Uso del conocimiento del modelo:** Dependiendo de los requisitos específicos de la tarea, se puede decidir si permitir que el LLM utilice su conocimiento previo para complementar la respuesta generada.
 4. **Incorporación de información de diálogo histórico:** Si existe información de diálogo histórico relevante, esta también puede ser fusionada en el prompt para permitir la generación de respuestas en diálogos de múltiples rondas (ChatBots).

El RAG basico presenta ciertas deficiencias sobre todo cuando se manejan volúmenes de datos muy grandes tiende a ser ineficiente en términos de velocidad de recuperación de contexto. Asimismo, no es apto para Chatbots debido a que es difícil mantener la coherencia y la continuidad en las conversaciones.

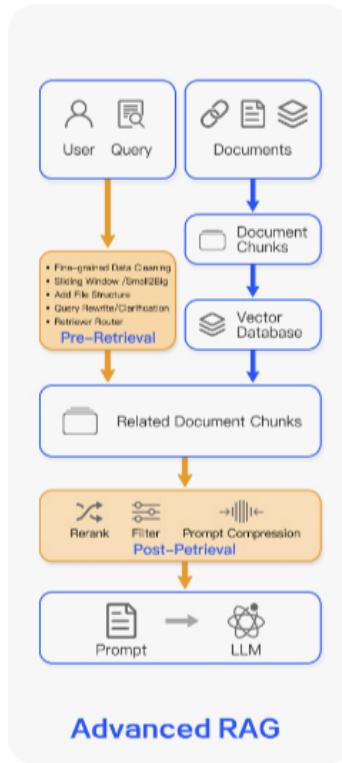


Figura 2.13: Advanced RAG.

El RAG avanzado es una mejora del RAG básico que incluye:

- **Mejoras en la generación de recuperación:** ha integrado métodos tanto de pre-recuperación como de post-recuperación para mejorar la calidad de la generación de recuperación.
- **Optimización del indexado:** Para abordar los problemas de indexación se ha implementado estrategias de optimización del indexado que incluyen el uso de técnicas como ventanas deslizantes, segmentación detallada y metadatos para mejorar la organización y accesibilidad de la información en el corpus de documentos.
- **Optimización del proceso de recuperación:** se ha introducido diversas metodologías para optimizar el proceso de recuperación de información buscando mejorar la

eficiencia y precisión en la selección de documentos relevantes.

- **Implementación específica:** tiene mayor flexibilidad en su implementación, permitiendo ajustes tanto a través de un pipeline, como de manera integral mejorando su adaptación a escenarios diversos.

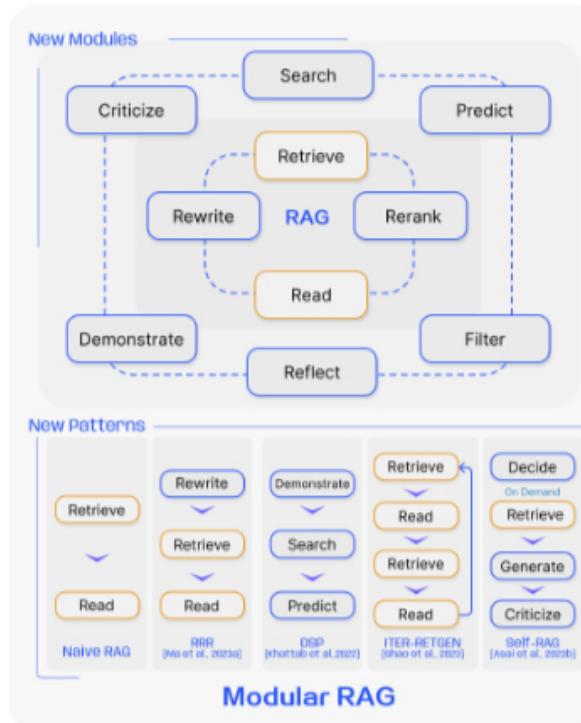


Figura 2.14: Modular RAG.

Para finalizar, vamos a presentar las características principales del modular RAG el cual esta siendo el paradigma principal que más se está usando en la implementación de RAG.

1. **Integración de métodos diversos:** Modular RAG integra una variedad de métodos para expandir los módulos funcionales del sistema mejorando su eficacia y precisión.
2. **Estructura reestructurada de módulos RAG:** se trata de la adaptación y evolución de los componentes del sistema para abordar problemas específicos identificados durante la implementación de enfoques anteriores.
3. **Enfoque iterativo:** dispone de un enfoque iterativo lo que proporciona al sistema la capacidad para realizar ajustes y mejoras de manera continua a lo largo del proceso.

Una vez conocemos las tres formas de implementar la técnica del RAG, vamos a señalar los medios para su evaluación. La evaluación proporciona información crucial sobre la eficacia, precisión y capacidad de respuesta del sistema, lo que permite identificar áreas de mejora y optimización. Aunque existen métricas específicas de evaluación como tasas de acierto, tasa de éxito, NDCG (Normalized Discounted Cumulative Gain), precisión.. nosotros vamos a hablar sobre 2 frameworks de evaluación: RAGAS[9] y ARES[32].

Por una parte, RAGAS se enfoca en evaluar la capacidad de los sistemas de RAG para identificar y utilizar párrafos de contexto relevantes, generar respuestas fieles a ese contexto y producir respuestas de alta calidad en general. Utiliza métricas como la fidelidad de la respuesta, la relevancia de la respuesta y la relevancia del contexto. Por otra parte, ARES utiliza datos anotados manualmente y datos sintéticos para entrenar modelos de lenguaje ligeros que evalúan los mismos items que RAGAS. La novedad es que ARES utiliza el enfoque de Predictive-Driven Reasoning (PDR) para proporcionar intervalos de confianza estadísticos y mejorar la precisión de la evaluación. En este proyecto se usará como framework de evaluación RAGAS.

Capítulo 3

Diseño e implementación.

Durante el desarrollo de AgriChat hemos trabajado en dos grandes áreas de desarrollo, la primera para ajustar el modelo Table Transformer de Microsoft, cuyo objetivo es la extracción de información no estructurada y su posterior transformación en información estructurada en formato .csv. La segunda área incluye todos aquellos métodos y herramientas empleadas para el desarrollo de un sistema RAG.

3.1. Ajuste de Table Transformer.

3.1.1. Investigación previa

En un principio, iniciamos un proceso de investigación sobre modelos ya entrenados para la identificación y extracción de tablas en documentos .pdf, fruto de este proceso seleccionamos como posibles candidatos a los modelos cascadeTabNet, TableFormer y Table Transformer de Microsoft.

De los tres modelos identificados y tras realizar los primeros ensayos ninguno de los tres era capaz de identificar correctamente los elementos que eran tablas de datos debido principalmente a la naturaleza de los datos con los que trabajamos que no siguen una estructura típica de datos tabulares.

Tras un proceso de ensayo el modelo que ofrecía mejores resultados fue Table Transformer de Microsoft ya que, aunque no identificaba correctamente la estructura de la tabla si que era capaz, aunque con mucha variabilidad, de identificar que es una tabla dentro del contexto no estructurado de los pdfs.

3.1. AJUSTE DE TABLE TRANSFORMER.

| Dataset | Model | Precision | Recall | F1 |
|---------|-------------|--------------|--------------|--------------|
| Both | ResNeXt-101 | 05.93 | 90.44 | 93.11 |
| | ResNeXt-152 | 06.72 | 88.95 | 92.67 |
| | Ours | 02.09 | 95.71 | 94.33 |
| Latex | ResNeXt-101 | 87.44 | 95.12 | 91.12 |
| | ResNeXt-152 | 87.20 | 96.24 | 91.49 |
| | Ours | 05.92 | 97.28 | 96.60 |
| Word | ResNeXt-101 | 95.77 | 76.10 | 84.81 |
| | ResNeXt-152 | 96.50 | 80.32 | 87.67 |
| | Ours | 94.35 | 95.49 | 94.92 |

Datos de entrenamiento

Datos disponible

Figura 3.1: Relación datos disponibles.

El modelo que finalmente seleccionamos fue el Table Transformer el cual usariamos para la extracción y el reconocimiento de la estructura de la tablas y se procedió a investigar más profundamente el modelo con el fin de construir una estrategia de acción que nos permitiera mejorar los resultados.

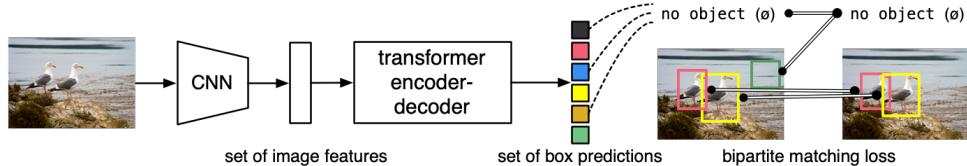


Figura 3.2: Arquitectura Table Transformer.

Como se observa en la imagen superior el modelo tiene como entrada una imagen la cual es procesada por los dos grandes componentes del modelo, la parte convolucional y la parte transformer. Una vez el modelo ha ingerido los datos en forma de imagen ésta es procesada por la parte convolucional que es la encargada de identificar aquellos elementos que son una tabla para posteriormente enviar esa predicción al componente transformer el cual identifica las características de las tablas.

Así pues, el modelo Table Transformer esta dividido en dos componentes que son a su vez modelos neuronales que hay que ajustar para obtener un flujo de identificación y reconocimiento de las tablas coherentes con nuestros objetivos. Ambos modelos se encuentran alojados en la plataforma de HuggingFace y son accesibles públicamente.

- Detección tablas: <https://huggingface.co/microsoft/table-transformer-detection>
- Estructura: <https://huggingface.co/microsoft/table-transformer-structure-recognition>

Una vez identificado el modelo que usaríamos, investigado su arquitectura y comprendido como nos puede ayudar a la consecución de nuestros objetivos planteamos como plan de acción un ajuste de ambos modelos.

3.1.2. Creación del conjunto de datos

Dado que teníamos que ajustar el modelo fue necesario la construcción de dos conjuntos de datos, el primero anotando las tablas de nuestro datos con el fin de señalar que son las tablas y el segundo conjunto de datos para señalar cómo está estructurada nuestras tablas, osea, que es una fila, una columna, un título, etc. Este proceso de anotación de datos con intervención humana es un proceso largo y tedioso por lo que procedimos a buscar herramientas de automatización del mismo que nos permitiera un equilibrio considerable entre el tiempo de experimentación disponible y la anotación del mayor número de instancias posibles.

La herramienta de anotación de imágenes elegida fue Roboflow, el cual es un servicio que abarca todo el proceso de entrenamiento de modelos de visión por computación, desde la construcción del dataset en diversos formatos hasta el entrenamiento y despliegue del modelo.

Para el primer dataset, y tras el análisis de los datos de los que disponíamos, construimos un dataset en formato COCO con 1.183 instancias de las cuales 837 (70 %) instancias fueron para entrenamiento, 237 (20 %) instancias para validación y 118 (10 %) para test. La imagen siguiente muestra nuestra conceptualización de lo que es una tabla.

3.1. AJUSTE DE TABLE TRANSFORMER.

The figure shows three separate tables, each with a 'Tabla' label to its right:

- Plazos de Seguridad (Protección del Consumidor)**: A table showing usage periods (Uso) and corresponding safety periods (P.S. [días]). It includes rows for Tomate, Jitomate, Cebolla, etc., with values like 14, 21, 28, and NP.
- Fabricante**: A table listing the manufacturer's information: KKV ITALIA S.R.L., Via Carlo Della Rocca, 7, 20021 Rescaldina (MI), ITALIA.
- Composición**: A table showing the composition of the product, specifically SULFATO CUPROCALCICO 20% (EXPR. EN CIU) [MP] P/P.

Figura 3.3: Tablas anotadas.

El segundo dataset se construyo en formato COCO con 818 instancias anotadas de las cuales 634 (78 %) instancias son para entrenamiento, 100 (22 %) instancias son para validación y 84 () para test. En esta fase se anotaron menos instancias debido a la dificultad en términos de tiempo de anotar toda la estructura que compone una tabla. En la imagen mostramos un ejemplo de tablas totalmente anotadas.

The figure shows three tables with extensive annotations using red boxes:

- Plazos de Seguridad (Protección del Consumidor)**: The entire table is highlighted with a large red box. Specific cells are also highlighted: 'Uso' (highlighted in red), 'P.S. (días)' (highlighted in red), and individual row entries like 'Tomate' (highlighted in red).
- Condiciones Generales de Uso**: A section containing general usage conditions, with the entire box highlighted in red.
- Clase de Usuario**: A section indicating the user class, with the entire box highlighted in red.

Figura 3.4: Tablas totalmente anotadas.

3.1.3. Ajuste del modelo

A continuación, entramos en la fase de ajuste de los modelos y dado que ajustar modelos implica el uso de recursos computacionales potentes se contrató 200 unidades informática en Google Colab usando para el entrenamiento de ambos modelos una L4 GPU 22.5 GB de VRAM y con 62GB de RAM.

Para ambos modelos, tanto para entrenamiento como en la validación, usamos las siguientes métricas de evaluación:

- **Loss Bbox:** cuantifica la precisión de las predicciones de las coordenadas de las cajas delimitadoras durante el entrenamiento.
- **Loss Giou:** es una medida de superposición entre las cajas delimitadoras predichas y las cajas delimitadoras verdaderas.
- **Training_loss y Evaluating_loss:** proporciona una visión general del rendimiento general del modelo en términos de la optimización de los parámetros durante el entrenamiento o la validación.

En primer lugar, empezamos ajustando el modelo de identificación de tablas para lo cual probamos diferentes configuraciones para los hiperparámetros, vemos un resumen en siguiente tabla:

Tabla 3.1: Hiperparámetros detección tablas

| Ver | Epoch | Batch | Lr | Lr_CNN | Optimizador | Loss Bbox | | Loss Giou | | Loss | |
|----------|-----------|----------|--------------|--------------|--------------|----------------|---------------|----------------|---------------|---------------|--------------|
| | | | | | | bbox.Train | bbox.Eval | Giou.Train | Giou.Eval | Train | Eval |
| 0 | 10 | 2 | 1e-04 | 1e-05 | AdamW | 0.01259 | 0.01776 | 0.09506 | 0.103 | 0.2559 | 0.3336 |
| 1 | 20 | 2 | 1e-04 | 1e-05 | AdamW | 0.01932 | 0.01644 | 0.0804 | 0.0881 | 0.2489 | 0.2706 |
| 2 | 20 | 6 | 1e-04 | 1e-05 | AdamW | 0.0091113 | 0.01591 | 0.05579 | 0.09181 | 0.158 | 0.2916 |
| 3 | 40 | 2 | 1e-04 | 1e-05 | AdamW | 0.08463 | 0.0171 | 0.08463 | 0.1015 | 0.2822 | 0.332 |
| 4 | 40 | 6 | 1e-04 | 1e-05 | AdamW | 0.01026 | 0.01523 | 0.05213 | 0.08224 | 0.219 | 0.2586 |
| 5 | 80 | 2 | 1e-04 | 1e-05 | AdamW | 0.01379 | 0.01844 | 0.07676 | 0.1137 | 0.2147 | 0.3588 |

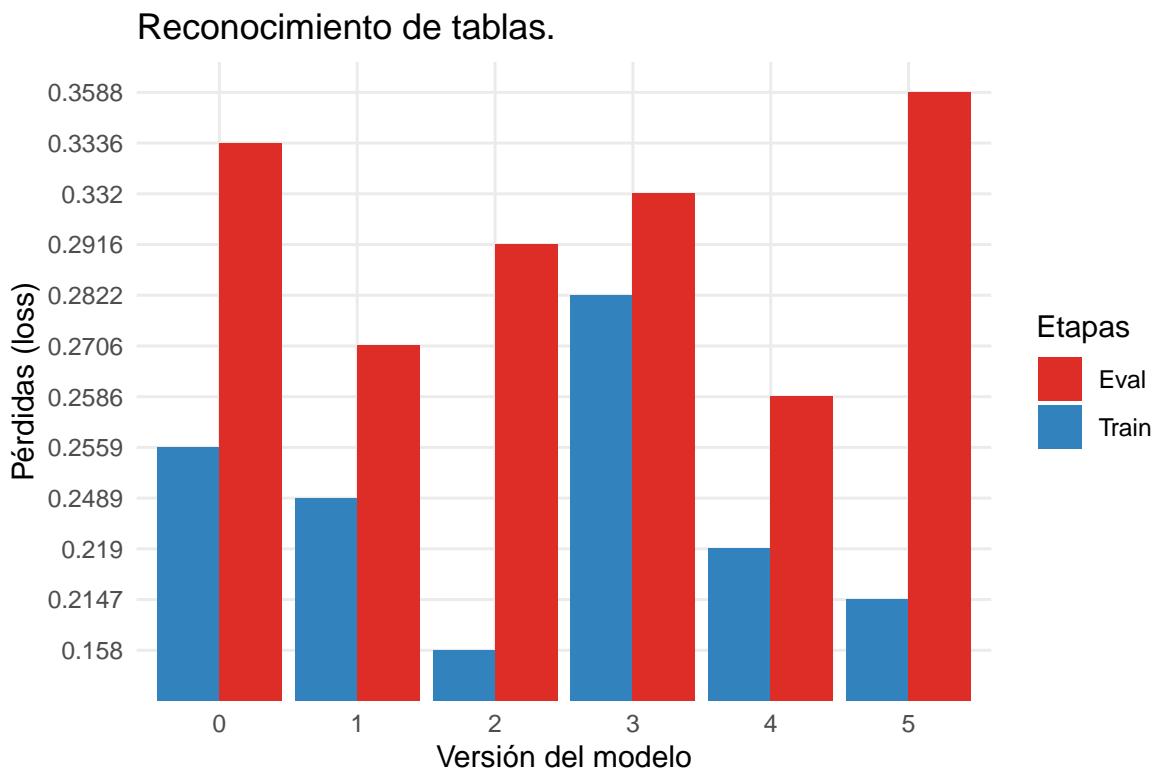
Tal y como observamos en la tabla y dado que nuestro objetivo es mejorar la identificación de las tablas en términos de la mejor predicción de la caja delimitadora (bbox) finalmente seleccionamos la configuración del modelo número 3.

El siguiente paso fue el ajuste del modelo identificador de la estructura de la tabla para lo cual procedimos como en el apartado anterior. Los experimentos (ajuste de hipérparámetros) se pueden observar en la siguiente tabla.

Tabla 3.2: Hiperparámetros detección de la estructura de tablas

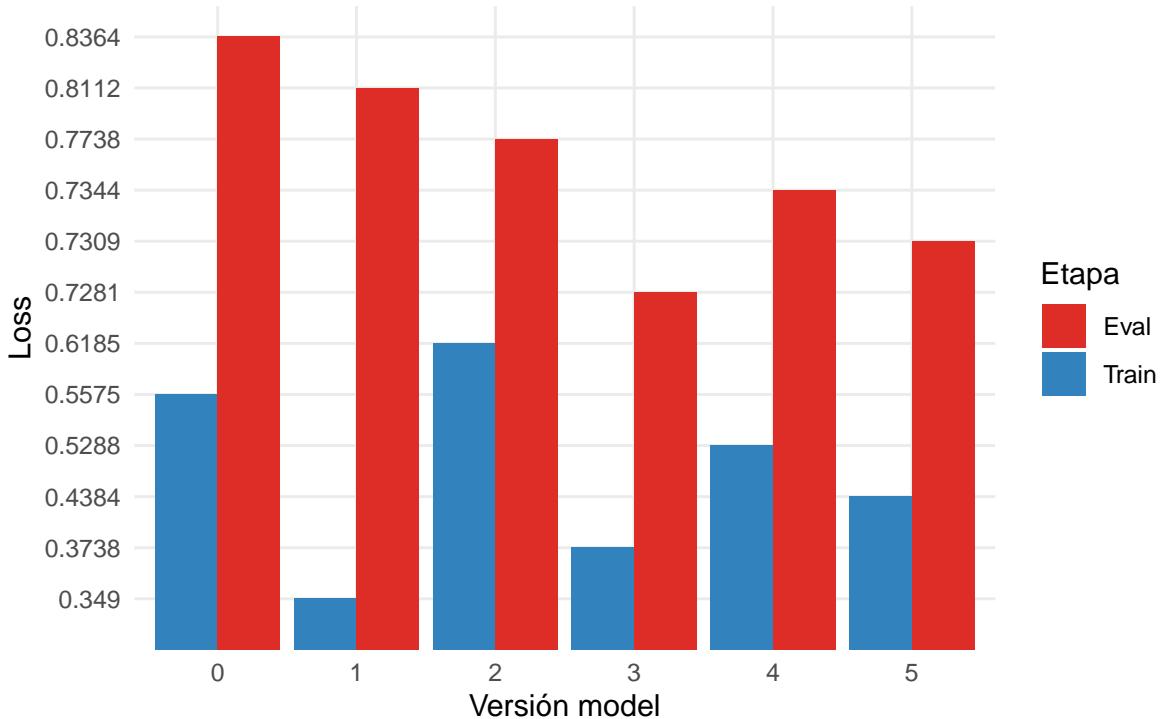
| Ver | Epoch | Batch | Lr | Lr_CNN | Optimizador | Loss Bbox | | Loss Giou | | Loss | |
|----------|-----------|----------|--------------|--------------|--------------|----------------|----------------|---------------|---------------|---------------|---------------|
| | | | | | | bbox.Train | bbox.Eval | Giou.Train | Giou.Eval | Train | Eval |
| 0 | 10 | 2 | 1e-04 | 1e-05 | AdamW | 0.03994 | 0.05074 | 0.1368 | 0.2105 | 0.5575 | 0.8364 |
| 1 | 20 | 2 | 1e-04 | 1e-05 | AdamW | 0.0219 | 0.04625 | 0.09954 | 0.2091 | 0.349 | 0.8112 |
| 2 | 20 | 6 | 1e-04 | 1e-05 | AdamW | 0.02685 | 0.04304 | 0.1943 | 0.2014 | 0.6185 | 0.7738 |
| 3 | 40 | 6 | 1e-04 | 1e-05 | AdamW | 0.02718 | 0.03966 | 0.1058 | 0.2013 | 0.3738 | 0.7281 |
| 4 | 40 | 2 | 1e-04 | 1e-05 | AdamW | 0.01925 | 0.04418 | 0.1608 | 0.1944 | 0.5288 | 0.7344 |
| 5 | 60 | 2 | 1e-04 | 1e-05 | AdamW | 0.02875 | 0.04318 | 0.1447 | 0.1935 | 0.4384 | 0.7309 |

Una vez que ambos modelos estaban ajustados, tuvimos que tomar una decisión sobre que versión del modelo subiríamos al hub de HuggingFace para su inferencia. Para ello, usamos como métrica de desempeño el training_loss y evaluating_loss para cada modelo ya que ambas nos dan una idea del ajuste de los modelos. En el caso del identificador de tablas obtuvimos el siguiente gráfico:



Al comparar las versiones vemos cierto grado de sobreajuste, sobretodo en las versiones 0, 2, 4 y 5 ya que la pérdida en entrenamiento es mucho más bajo que en validación lo que nos señala cierto sobreajuste y en consecuencia falta de generalización. La versión del modelo seleccionado fue la 3 debido a que la diferencia entre entrenamiento y validación no es muy grande con respecto a las diferencias de las demás versiones, aun así su puntuación de pérdida es elevada.

Estructura de tablas.



Por último, para el modelo de reconocimiento de la estructura de tablas obtuvimos un panorama parecido al anterior. Podemos observar problemas de sobreajuste en las versiones 0, 1, 3 y 5. Finalmente seleccionamos la versión 4 ya que, tiene las puntuaciones más bajas de todas las versiones. Debemos de considerar que ambos modelos han sido entrenados con pocas instancias, tal y como explicamos anteriormente.

3.1.4. Proceso de extracción de información tabular.

Tal y como hemos explicitado, uno de nuestros objetivos era explorar soluciones open-source que nos permitiera la extracción de información tabular desestructurada y su posterior codificación en información estructurada. Para la consecución del mismo, construimos un pipeline de procesamiento tal y como muestra la siguiente imagen.

El proceso inicia con la adquisición de datos, en nuestro caso, una muestra de fichas informativas sobre productos fitosanitarios que emite el Ministerio de Agricultura, Pesca y Alimentación en formato pdf. Tras su lectura, cada página de cada documento pdf es transformada con la biblioteca pdf2img a un formato de imagen y guardadas en directorios independientes cuyo nombre es el del documento original.

3.1. AJUSTE DE TABLE TRANSFORMER.

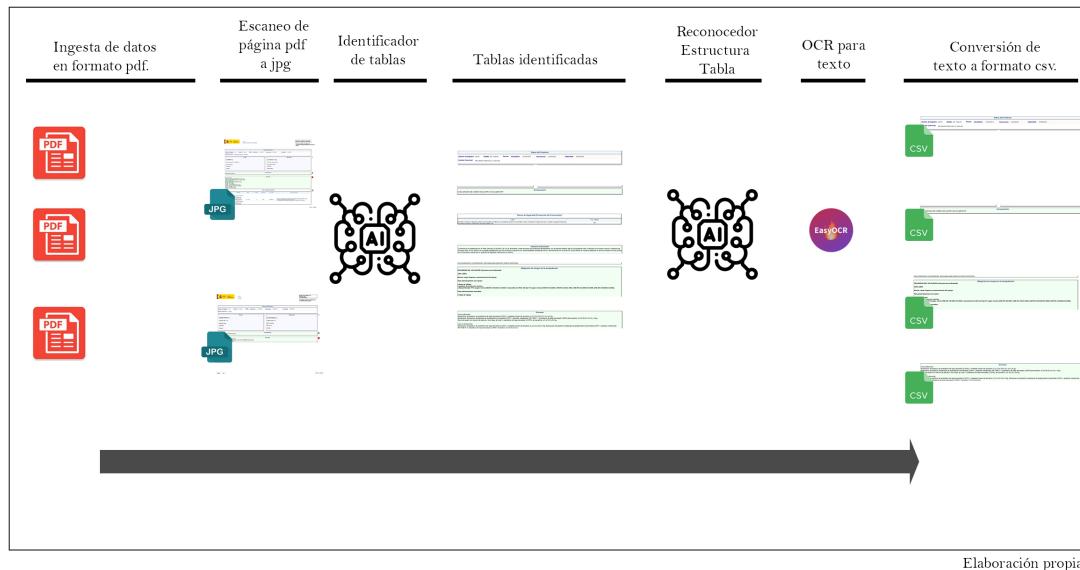


Figura 3.5: Pipeline para la extracción de información.

Una vez que la información la tenemos en un formato de imagen es preprocesada ajustando sus dimensiones según las especificaciones que el modelo de identificación de tablas nos señala. Posteriormente la imagen es procesada por el modelo que es capaz de predecir los bbox entorno a las tablas identificadas. En este momento disponemos de un conjunto de bbox los cuales usamos para recortar de la imagen original las tablas. El resultado es un conjunto de pequeñas imágenes que representan tablas.

El conjunto de imágenes son enviadas al modelo de reconocimiento de la estructura de la tabla el cual predice los bbox correspondientes a cada elemento de una tabla (row, column, header, . . .).

Debido principalmente a las pocas muestras que el modelo ha visto en su entrenamiento fue necesario realizar un trabajo de filtrado y clasificación de tipos de tabla, de tal forma que pudieramos centrarnos en aquellas predicciones que se ajustaban correctamente a las tablas para posteriormente cuando el modelo se reentrenara y mejorara sus predicciones pescindir de este mecanismo de seguridad.

Las tablas fueron clasificadas según su título en la siguiente estructura:

```

def clear_tables(tables):
    ^  format_tables = {
        'DATOS DEL PRODUCTO': 'A',
        'TITULAR': 'A',
        'FABRICANTE': 'A',
        'COMPOSICION': 'A',
        'ENVASES': 'A',
        'CONDICIONES GENERALES DE USO': 'A',
        'CLASE DE USUARIO': 'A',
        'MITIGACION DE RIESGOS EN LA MANIPULACION': 'A',
        'MITIGACION DE RIESGOS AMBIENTALES': 'A',
        'ELIMINACION DEL PRODUCTO Y/O CALDO': 'A',
        'GESTION DE ENVASES': 'A',
        'OTRAS INDICACIONES REGLAMENTARIAS': 'A',
        'CONDICIONES DE ALMACENAMIENTO': 'A',
        'USOS Y DOSIS AUTORIZADOS': 'B',
        'PLAZOS DE SEGURIDAD (PROTECCION DEL CONSUMIDOR)': 'C',
        'CLASIFICACIONES Y ETIQUETADO': 'D'
    }

```

Figura 3.6: Clasificación de tipos de tabla.

El mecanismo implantado esta diseñado para no procesar tablas cuyas cabeceras no sean válidas en el sentido de que han sido correctamente identificadas y al tiempo clasificar la tabla según el tipo (A,B,C,D) para posteriormente procesarlas por tipo de tabla.

Una vez disponíamos de una estructura de tablas con coherencia, usamos la biblioteca easyocr para escanear el contenido de las imágenes seleccionando de su bbox las filas, columnas, o su intersección (celdas) que contiene la información a extraer. En esta fase de desarrollo, AgriChat solo procesa tablas de tipo A, lo que supone el 78 % del contenido de cada pdf.

Por último, una vez reconstruida la tabla se exporta en formato .csv para su descarga o procesamiento por otro sistema.

3.2. Construcción del sistema RAG.

3.2.1. Arquitectura del modelo RAG.

En nuestro caso, y dado el carácter exploratorio del mismo, optamos por la implementación de un modelo RAG avanzado que nos permitiera desarrollar un sistema dentro del espacio temporal del proyecto. En la imagen mostramos la arquitectura empleada y sus componentes principales.

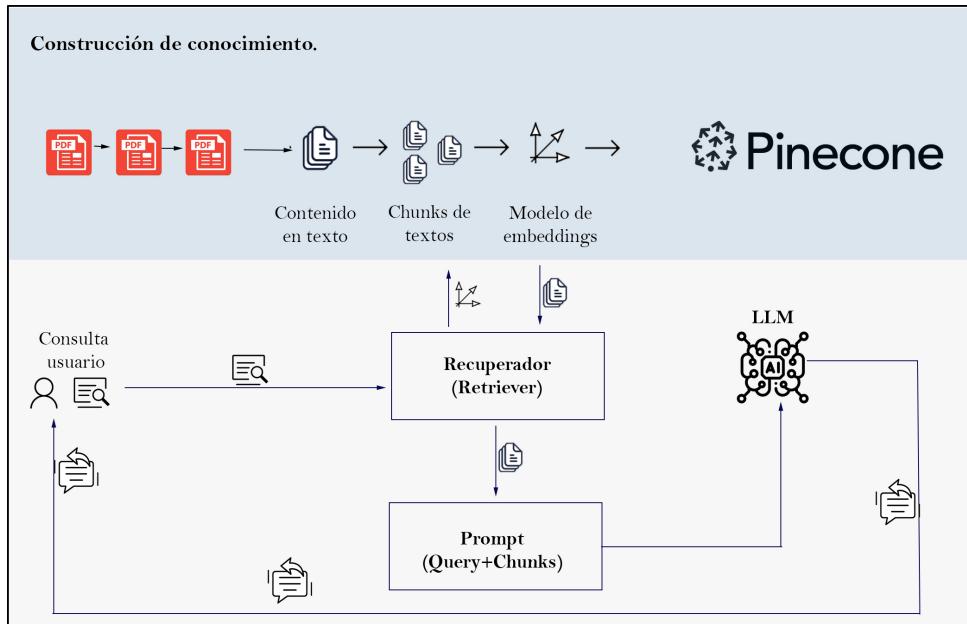


Figura 3.7: Arquitectura RAG empleada.

Tal y como vemos, el sistema se puede dividir en dos grandes bloques: el primero fue la construcción de una base de conocimiento y el segundo la construcción del propio sistema RAG para la explotación de la base de conocimientos. Veamos una descripción de los componentes que forman el RAG centrandonos en primer lugar en la construcción de la base de conocimientos.

La base de conocimientos cumple una función muy importante, actúa como un álmacen de conocimientos, que en nuestro caso supone la información almacenada en los documentos pdf sobre los cuales el usuario generará sus preguntas.

El proceso se inicia con la carga y lectura de los documentos sobre los cuales el usuario quiere obtener conocimiento. A continuación, realizamos un proceso de división de los documentos en chunks para lo cual usamos el framework LLamaIndex con la siguiente configuración: en primer lugar, usamos la técnica de la ventana deslizante (slicing windows) con una superposición (overlap) de 100 palabras y un tamaño de chunk de 512 palabras. Con esta técnica se intenta que los vectores generados fuesen capaces de almacenar información semántica, sintáctica y contextual.

Tras la construcción de los chunks, éstos los traducimos a vectores densos con el modelo de embeddings de OpenIA llamado *text-embedding-3-small* con una dimensión de 1536

componentes.

Por último, tuvimos que buscar una base de datos vectorial para el almacenamiento de los vectores creados. A pesar de la existencia de soluciones open-source para el almacenamiento elegimos el servicio de Pinecone ya que dispone de un plan gratuito que cubría toda la fase de experimentación. Para la configuración del servicio de Pinecone usamos como métrica la similitud coseno.

Una vez configurado e implantado la base de conocimientos nos centramos en la construcción del componente recuperador (retriever), éste es el encargado de recuperar la información de nuestra base de datos de conocimientos más relevante según la consulta de los usuarios. Una vez que el usuario realiza la consulta sobre la documentación cargada al sistema, ésta fue convertida a embeddings con el algoritmo de OpenIA *text-embedding-3-small* para ser enviada a la base de datos vectorial desde donde nos devuelve los 6 chunks con mayor similaridad con respecto a la consulta planteada. El componente devuelve la representación textual de los 6 vectores densos con más similaridad a la consulta planteada por el usuario.

En esta fase, ya disponíamos de la información recuperada desde nuestra base de conocimientos y es hora de construir el prompt. Aunque existen muchas formas de configurar el prompt, en nuestro caso decidimos usar una template básica con dos variables de entrada, la consulta del usuario y el contexto aumentado recuperado.

Por último, el prompt construido se envía al LLM, en nuestro caso se envió al modelo *gpt-3.5-turbo-0125* de OpenAI con un máximo de tokens de 300 y una temperatura de 0.3.

Capítulo 4

Resultados y líneas de trabajo futuras.

4.1. Resultados

Tal y como hemos ido desarrollando en la presente memoria los resultados de nuestra investigación se han desagregado en las dos grandes áreas que hemos descrito: la extracción de información no estructurada y la evaluación del sistema RAG desarrollado mediante el framework RAGAS.

En el área de extracción de información no estructurada los resultados han sido mediados por la falta de tiempo para generar instancias de entrenamiento para ambos modelos, aun así desarrollamos un experimento con 5 documentos elegidos al azar.

Para el modulo de reconocimiento de tablas los resultados fueron los siguientes:

Tabla 4.1: Tablas identificadas.

| Nombre | Num.Pages | Tablas.Totales | Tablas.Identificadas | Porcentaje |
|----------|-----------|----------------|----------------------|------------|
| 11608 | 4 | 14 | 11 | 78 % |
| 11811 | 6 | 14 | 13 | 92 % |
| ES-00237 | 5 | 15 | 15 | 100 % |
| ES-00238 | 3 | 15 | 13 | 86 % |
| 14108 | 6 | 15 | 12 | 80 % |

De los cinco documentos, contamos el total de tablas que el modelo debería de reconocer y lo relacionamos con las realmente identificadas. Podemos observar como este módulo reconoce un conjunto amplio de tablas, su media para los cinco documentos es de alrededor del 87 % de tablas correctamente identificadas.

Por consiguiente, para evaluar los resultados del módulo de reconocimiento de la estructura de las tablas tenemos que tener en cuenta que el mecanismo de seguridad implantado solo procesa las tablas tipo A, las más comunes en los pdfs, por lo que el cuadro siguiente solo recoge los resultados para el tipo de tabla señalado y que han podido ser convertidos en información estructurada en formato .csv para su descarga o transmisión a otros sistemas.

Tabla 4.2: Reconocimiento estructura de Tablas tipo A

| Nombre | Num.Pages | Tablas.Totales | Tablas..csv | Porcentaje |
|----------|-----------|----------------|-------------|------------|
| 11608 | 4 | 11 | 7 | 63 % |
| 11811 | 6 | 12 | 7 | 58 % |
| ES-00237 | 5 | 12 | 8 | 66 % |
| ES-00238 | 3 | 12 | 7 | 58 % |
| 14108 | 6 | 12 | 7 | 58 % |

Así pues, vemos que los resultados de este módulo no han sido muy elevados, ya que su media ronda el 60 %. Debemos de recordar que este módulo fue el que menos instancias tuvo para el entrenamiento.

Por otra parte, para la evaluación de los resultados del RAG se usaron las siguientes métricas que proporciona el propio framework.

Para la fase de generación:

- **Faithfulness:** mide la coherencia fáctica de la respuesta generada frente al contexto dado. Se calcula a partir de la respuesta y el contexto recuperado. La respuesta está escalada al rango (0,1). Cuanto más alto mejor.
- **Answer_relevancy:** se centra en evaluar qué tan pertinente es la respuesta generada para la pregunta dada.

Para la fase de recuperación (retrieval):

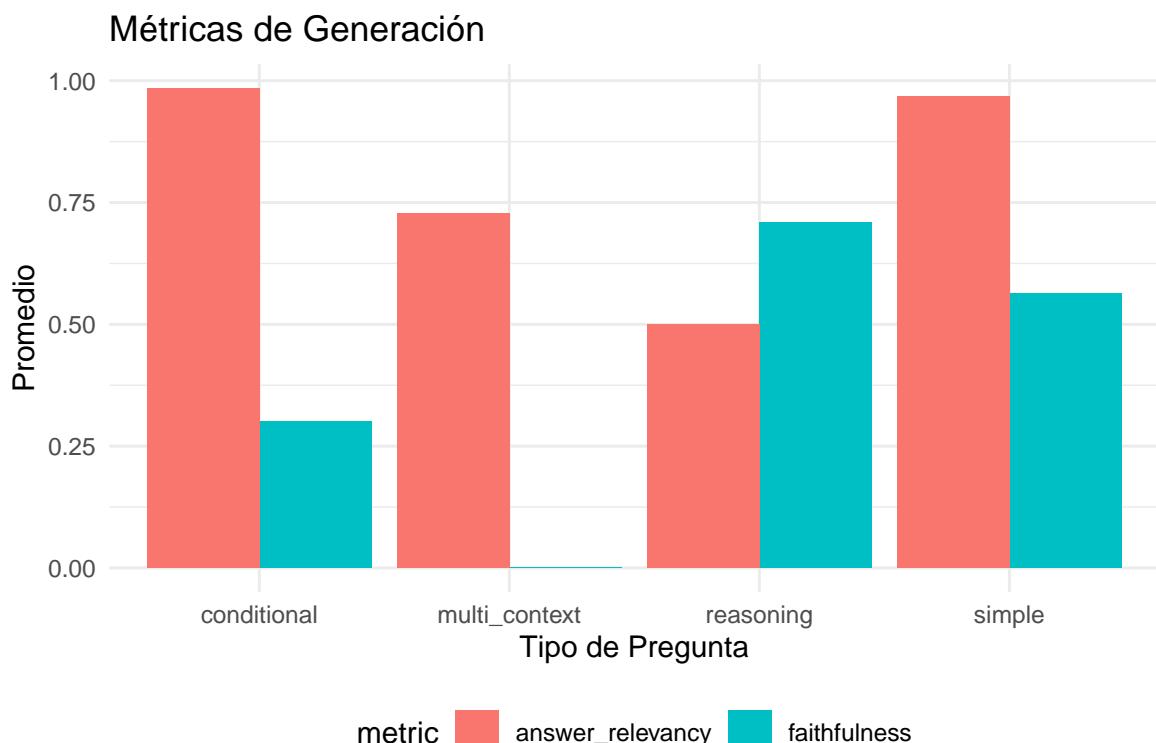
- **context precision:** una métrica que evalúa si todos los elementos relevantes de la verdad fundamental presentes en los contextos recuperados tienen una clasificación más alta o no. Lo ideal es que todos los fragmentos relevantes aparezcan en los primeros puestos.

- **contex recall:** mide hasta qué punto el contexto recuperado se alinea con la respuesta anotada, tratada como la verdad fundamental.

Y para la evaluación del rendimiento general (end to end) usamos la métrica:

- **Answer Correctness:** mide la precisión de la respuesta generada en comparación con la verdad básica. La corrección de la respuesta abarca dos aspectos críticos: la similitud semántica entre la respuesta generada y la verdad fundamental, así como la similitud fáctica

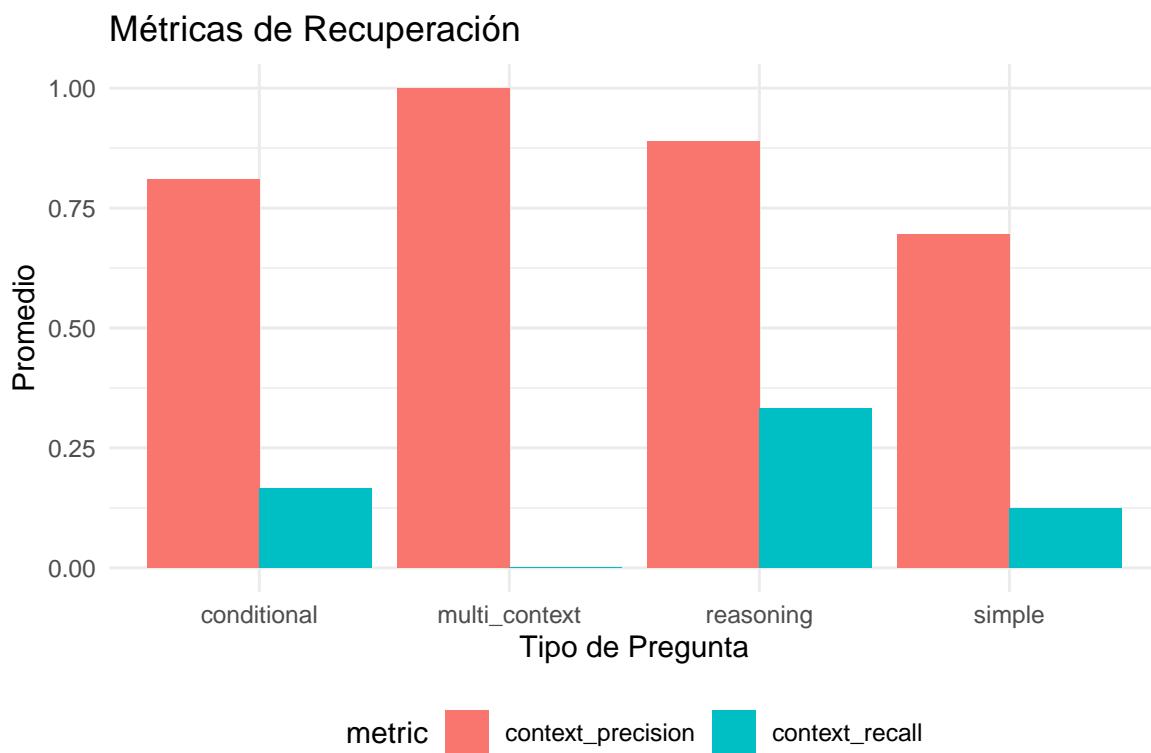
El procedimiento de captura y producción de datos fue usando el propio RAG con un documento pdf elegido al azar y 20 preguntas sintéticas generadas por RAGAS sobre el contenido del documento. Los resultados de las métricas fueron guardado en un .csv del cual analizamos el comportamiento del RAG según el tipo de pregunta generada (simple, reasoning, multi_context y conditional), la fase (generador, retriever) y sus métricas asociadas. A parte se añadio una métrica general del desempeño.



Podemos observar como en la fase de generación de las respuestas, el sistema puntuó bastante bien en preguntas de tipo reasoning y simple según la métrica *Faithfulness* aunque

con valores no muy altos superan el 0.5 con lo que el sistema genera respuestas con coherencia fáctica. Sin embargo, vemos que el tipo de preguntas que más le cuesta son las de tipo conditional y multi_context, siendo esta última la que peor puntuación.

Si nos fijamos ahora en la métrica *answer_relevancy* vemos un buen desempeño en los 4 tipos de pregunta, siendo la mejor las de tipo conditional y simple. De forma general, la fase de generación produce respuestas pertinentes y con coherencia fáctica, excepto cuando las preguntas son de tipo conditional y multi_context donde se producen afirmaciones incoherentes o incorrectas, fenómeno llamado alucinaciones.

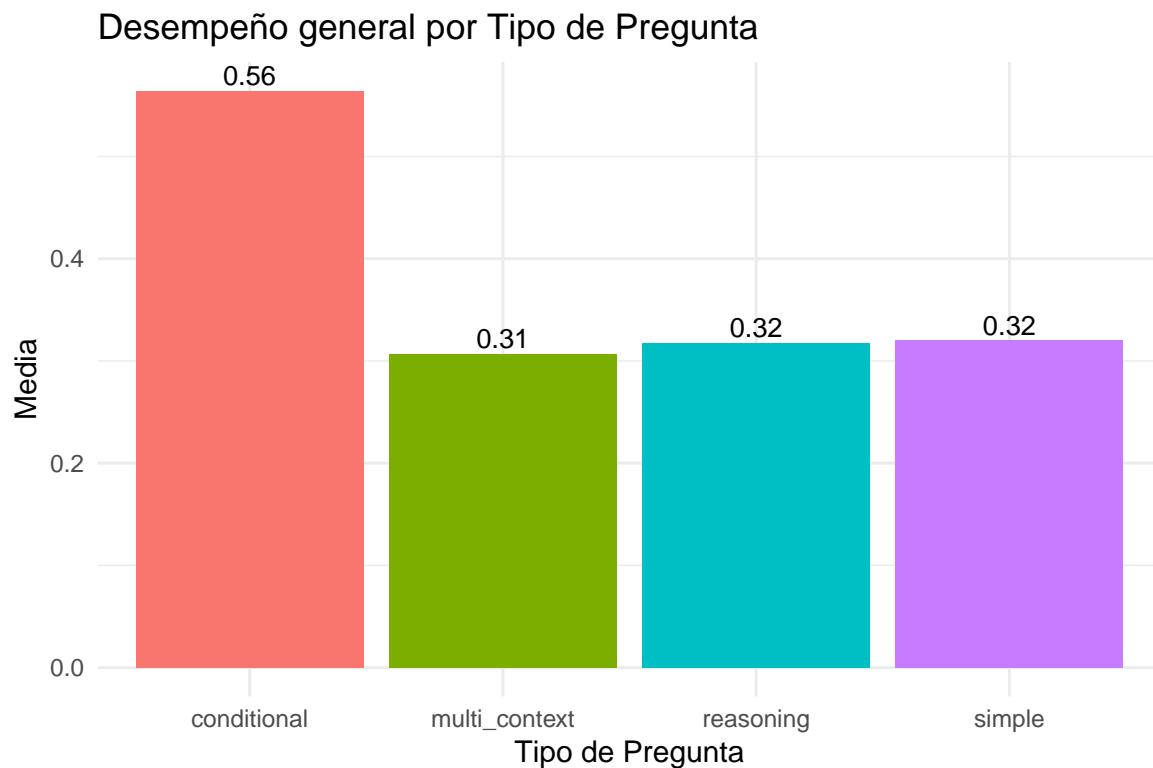


En cuanto a la fase de recuperación semántica desde la base de datos vectorial vemos que la métrica *context_precision* puntuúa bastante alto para todos los tipos de pregunta siendo el mejor valor en las de tipo multi_context lo que nos indica que el sistema recupera contexto con información relevante y priorizada así pues, el modelo tiene a su disposición un contexto coherente con la pregunta planteada.

Por el contrario, la métrica *context_recall* tiene puntuaciones extremadamente bajas para todo tipo de preguntas, por lo que hay que considerar la posibilidad de que los contextos recuperados no cubren todos los conceptos importantes dentro de la verdadera respuesta

(generada por GPT-4) y por lo tanto faltan conceptos importantes que no han podido recuperarse. Sus efectos suelen ser respuestas parciales o incluso incorrectas.

Para terminar vamos a examinar el desempeño general de todo el sistema RAG por tipo de pregunta.



Tal y como se observa, los tipos de pregunta que peor puntuhan son multi_context, reasoning y simple, hecho que ya nos estaba señalando las métricas de recuperación y generación analizadas en los párrafos anteriores. Podemos destacar que las preguntas de tipo condicional son las que mejor puntuhan aunque con valores bajos.

Debemos de tener en cuenta que el proceso de evaluación actua como una línea base o punto cero desde el cual podemos medir el impacto o los efectos de nuestras acciones de mejora en torno a AgriChat y guiarnos en la implantación de futuras líneas de investigación.

4.2. Líneas de trabajo futuras.

El presente trabajo y por ende el producto implementado (AgriChat) está sometido a los vaivenes propios que conlleva la construcción de producto digitales por lo que su final

no termina con el actual trabajo, sino que su evolución es ante todo un proceso de mejora continua.

No obstante, durante el desarrollo del mismo hemos detectado ciertas áreas sin cubrir y que para nosotros supone líneas de trabajo futuras.

El objetivo principal de AgriChat ha pivotado sobre el uso eficiente de los productos fitosanitarios mediante la mejora de los proceso de información que la comunidad agrícola emplea en su día a día. Del conocimiento de los patrones de organización de las explotaciones agrarias se extrae un uso significativo de las aplicaciones de mensajería tipo WhatsApp o Telegram por lo que sería interesante explorar la posibilidad de que AgriChat pase a ser un ChatBot dentro del ecosistema de las aplicaciones móviles. Suponemos que este cambio mejoraría la accesibilidad y rapidez a la información de los productos fitosanitarios al usar un dispositivo móvil y no un ordenador portátil o de sobremesa para el acceso a la misma.

Por otra parte, los procesos de transformación digital de las sociedades también tiene un impacto en las explotaciones agrícolas. Nace la obligación legal de registrar todas las acciones que las explotaciones agrícolas realizan sobre el medioambiente y anotarlas en los llamados “cuadernos de campo” cuyos items actúan como indicadores de sostenibilidad medioambiental. Este hecho supone una nueva área de investigación, de tal forma, que AgriChat no solo podría usarse para obtener información sobre el uso de los productos fitosanitarios sino como un evaluador de la pertinencia del uso de un fitosanitario en una explotación agraria determinada en base al estado de los indicadores medioambientales subyacentes a la misma.

Por último, de los valores entorno a la mejora en la calidad alimentaria han emergido todo un conjunto de patrones de cultivo que podemos clasificar en productos sometidos al uso de fitosanitarios y aquellos que son ecológicos. AgriChat puede mejorar la información que el consumidor final dispone del producto mediante un “mapa de uso de productos fitosanitarios” que informe a la ciudadanía sobre el tipo y cantidad de productos fitosanitarios empleados por la explotación agrícola para la producción alimentaria. Este instrumento actuaria como reductor de la incertidumbre en los mercados agrícolas para cualquier agente de la cadena alimentaria.

Bibliografía

- [1] AHARONI, ROEE; JOHNSON, MELVIN y FIRAT, ORHAN (2019). «Massively Multilingual Neural Machine Translation».
<https://doi.org/10.48550/arXiv.1903.00089>.
- [2] ALLAIRE, JJ; XIE, YIHUI; DERVIEUX, CHRISTOPHE; MCPHERSON, JONATHAN; LURASCHI, JAVIER; USHEY, KEVIN; ATKINS, ARON; WICKHAM, HADLEY; CHENG, JOE; CHANG, WINSTON y IANNONE, RICHARD (2024). *rmarkdown: Dynamic Documents for R*.
<https://github.com/rstudio/rmarkdown>. R package version 2.26.
- [3] BOJANOWSKI, PIOTR; GRAVE, EDOUARD; JOULIN, ARMAND y MIKOLOV, TOMAS (2017). «Enriching Word Vectors with Subword Information».
<https://doi.org/10.48550/arXiv.1607.04606>.
- [4] BROWN, PETER F; COCKE, JOHN; DELLA PIETRA, STEPHEN A; DELLA PIETRA, VINCENT J; JELINEK, FREDERICK; MERCER, ROBERT L y ROOSSIN, PAUL (1988). «A statistical approach to language translation». En: *Coling Budapest 1988 Volume 1: International Conference on Computational Linguistics*, .
- [5] CHANG, YUPENG; WANG, XU; WANG, JINDONG; WU, YUAN; YANG, LINYI; ZHU, KAIJIE; CHEN, HAO; YI, XIAOYUAN; WANG, CUNXIANG; WANG, YIDONG; YE, WEI; ZHANG, YUE; CHANG, YI; YU, PHILIP S.; YANG, QIANG y XIE, XING (2023). «A Survey on Evaluation of Large Language Models».
<https://doi.org/10.48550/arXiv.2307.03109>.
- [6] COLLOBERT, RONAN; WESTON, JASON; BOTTOU, LÉON; KARLEN, MICHAEL; KAVUKCUOGLU, KORAY y KUKSA, PAVEL (2011). «Natural Language Processing (Al-

most) from Scratch».

<http://arxiv.org/abs/1103.0398>.

- [7] DETTMERS, TIM; PAGNONI, ARTIDORO; HOLTZMAN, ARI y ZETTLEMOYER, LUKE (2023). «QLoRA: Efficient Finetuning of Quantized LLMs».
<https://arxiv.org/abs/2305.14314>.
- [8] DEVLIN, JACOB; CHANG, MING-WEI; LEE, KENTON y TOUTANOVA, KRISTINA (2019). «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding».
- [9] ES, SHAHUL; JAMES, JITHIN; ESPINOSA-ANKE, LUIS y SCHOCKAERT, STEVEN (2023). «RAGAS: Automated Evaluation of Retrieval Augmented Generation».
<https://doi.org/10.48550/arXiv.2309.15217>.
- [10] GAO, YUNFAN; XIONG, YUN; GAO, XINYU; JIA, KANGXIANG; PAN, JINLIU; BI, YUXI; DAI, YI; SUN, JIAWEI; GUO, QIANYU; WANG, MENG y WANG, HAOFEN (2024). «Retrieval-Augmented Generation for Large Language Models: A Survey».
<https://doi.org/10.48550/arXiv.2312.10997>.
- [11] GURURANGAN, SUCHIN; MARASOVIĆ, ANA; SWAYAMDIPTA, SWABHA; LO, KYLE; BELTAGY, IZ; DOWNEY, DOUG y SMITH, NOAH A. (2020). «Don't Stop Pretraining: Adapt Language Models to Domains and Tasks».
<https://doi.org/10.48550/arXiv.2004.10964>.
- [12] HAN, ZEYU; GAO, CHAO; LIU, JINYANG; JEFF; ZHANG y ZHANG, SAI QIAN (2024). «Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey».
<https://doi.org/10.48550/arXiv.2403.14608>.
- [13] HOCHREITER, SEPP y SCHMIDHUBER, JÜRGEN (1997). «Long Short-Term Memory». *Neural Computation*, **9**(8), pp. 1735–1780. doi: 10.1162/neco.1997.9.8.1735.
- [14] HOULSBY, NEIL; GIURGIU, ANDREI; JASTRZEBSKI, STANISLAW; MORRONE, BRUNA; DE LAROUSSILHE, QUENTIN; GESMUNDO, ANDREA; ATTARIYAN, MONA y GELLY,

- SYLVAIN (2019). «Parameter-Efficient Transfer Learning for NLP».
<https://arxiv.org/abs/1902.00751v2>.
- [15] HU, DOU; ZHOU, MENGYUAN; DU, XIYANG; YUAN, MENGFEI; JIN, MEIZHI; JIANG, LIANXIN; MO, YANG y SHI, XIAOFENG (2022). «PALI-NLP at SemEval-2022 Task 4: Discriminative Fine-tuning of Transformers for Patronizing and Condescending Language Detection».
<https://doi.org/10.48550/arXiv.2203.04616>.
- [16] HU, EDWARD J.; SHEN, YELONG; WALLIS, PHILLIP; ALLEN-ZHU, ZEYUAN; LI, YUANZHI; WANG, SHEAN; WANG, LU y CHEN, WEIZHU (2021). «LoRA: Low-Rank Adaptation of Large Language Models».
<https://arxiv.org/abs/2106.09685v2>.
- [17] HUANG, LEI; YU, WEIJIANG; MA, WEITAO; ZHONG, WEIHONG; FENG, ZHANGYIN; WANG, HAOTIAN; CHEN, QIAGLONG; PENG, WEIHUA; FENG, XIAOCHENG; QIN, BING y LIU, TING (2023). «A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions».
<https://doi.org/10.48550/arXiv.2311.05232>.
- [18] JELINEK, FREDERICK y MERCER, ROBERT L. (1980). «Interpolated estimation of Markov source parameters from sparse data». En: *In Proceedings of the Workshop on Pattern Recognition in Practice*, pp. 381–397. Amsterdam, The Netherlands: North-Holland.
- [19] JIANG, SHUO; HU, JIE; MAGEE, CHRISTOPHER L. y LUO, JIANXI (2024). «Deep Learning for Technical Document Classification». *IEEE Transactions on Engineering Management*, **71**, p. 1163–1179. ISSN 1558-0040. doi: 10.1109/tem.2022.3152216.
<http://dx.doi.org/10.1109/TEM.2022.3152216>.
- [20] KUDO, TAKU (2018). «Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates».
<https://doi.org/10.48550/arXiv.1804.10959>.

- [21] LEE, NOAH; AN, NA MIN y THORNE, JAMES (2023). «Can Large Language Models Capture Dissenting Human Voices?».
<https://doi.org/10.48550/arXiv.2305.13788>.
- [22] LEWIS, PATRICK; PEREZ, ETHAN; PIKTUS, ALEKSANDRA; PETRONI, FABIO; KARPUKHIN, VLADIMIR; GOYAL, NAMAN; KÜTTLER, HEINRICH; LEWIS, MIKE; TAU YIH, WEN; ROCKTÄSCHEL, TIM; RIEDEL, SEBASTIAN y KIELA, DOUWE (2021). «Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks».
<https://doi.org/10.48550/arXiv.2005.11401>.
- [23] MCCULLOCH, WARREN y PITTS, WALTER (1943). «A Logical Calculus of Ideas Immanent in Nervous Activity». *Bulletin of Mathematical Biophysics*, **5**, pp. 127–147.
- [24] MEUSCHKE, NORMAN; JAGDALE, APURVA; SPINDE, TIMO; MITROVIĆ, JELENA y GIPP, BELA (2023). *A Benchmark of PDF Information Extraction Tools Using a Multi-task and Multi-domain Evaluation Framework for Academic Documents*, p. 383–405. Springer Nature Switzerland. ISBN 9783031280320. doi: 10.1007/978-3-031-28032-0_31.
http://dx.doi.org/10.1007/978-3-031-28032-0_31.
- [25] MIKOLOV, TOMAS; KARAFIÁT, MARTIN; BURGET, LUKAS; CERNOCKÝ, JAN y KHUDANPUR, SANJEEV (2010). «Recurrent neural network based language model». tomo 2, pp. 1045–1048. doi: 10.21437/Interspeech.2010-343.
- [26] MIKOLOV, TOMAS; SUTSKEVER, ILYA; CHEN, KAI; CORRADO, GREG y DEAN, JEFFREY (2013). «Distributed Representations of Words and Phrases and their Compositionality». <https://doi.org/10.48550/arXiv.1310.4546>.
- [27] MIKOLOV, TOMAS; SUTSKEVER, ILYA; CHEN, KAI; CORRADO, G.S y DEAN, JEFFREY (2013). «Distributed Representations of Words and Phrases and their Compositionality». *Advances in Neural Information Processing Systems*, **26**.
- [28] PENNINGTON, JEFFREY; SOCHER, RICHARD y MANNING, CHRISTOPHER (2014).

- «GloVe: Global Vectors for Word Representation». doi: 10.3115/v1/D14-1162.
<https://aclanthology.org/D14-1162>.
- [29] PETERS, MATTHEW E.; NEUMANN, MARK; IYYER, MOHIT; GARDNER, MATT; CLARK, CHRISTOPHER; LEE, KENTON y ZETTLEMOYER, LUKE (2018). «Deep contextualized word representations».
- [30] PRESS, OFIR; SMITH, NOAH A. y LEWIS, MIKE (2022). «Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation».
<https://doi.org/10.48550/arXiv.2108.12409>.
- [31] RADFORD, ALEC; WU, JEFFREY; CHILD, REWON; LUAN, DAVID; AMODEI, DARIO y SUTSKEVER, ILYA (2018). «Language Models are Unsupervised Multitask Learners».
<https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf>.
- [32] SAAD-FALCON, JON; KHATTAB, OMAR; POTTS, CHRISTOPHER y ZAHARIA, MATEI (2023). «ARES: An Automated Evaluation Framework for Retrieval-Augmented Generation Systems».
<https://doi.org/10.48550/arXiv.2311.09476>.
- [33] SCHUSTER, MIKE y NAKAJIMA, KAISUKE (2012). «Japanese and Korean voice search». doi: 10.1109/ICASSP.2012.6289079.
- [34] SENNINICH, RICO; HADDOW, BARRY y BIRCH, ALEXANDRA (2016). «Neural Machine Translation of Rare Words with Subword Units».
<https://doi.org/10.48550/arXiv.1508.07909>.
- [35] SHANNON, C. E. (1948). «A mathematical theory of communication». *The Bell System Technical Journal*, **27(3)**, pp. 379–423. doi: 10.1002/j.1538-7305.1948.tb01338.x.
- [36] SU, JIANLIN; LU, YU; PAN, SHENGFENG; MURTADHA, AHMED; WEN, BO y LIU, YUNFENG (2023). «RoFormer: Enhanced Transformer with Rotary Position Embedding».
<https://doi.org/10.48550/arXiv.2104.09864>.

- [37] TAO, ZHENGWEI; JIN, ZHI; BAI, XIAOYING; ZHAO, HAIYAN; FENG, YANLIN; LI, JIA y HU, WENPENG (2023). «EvEval: A Comprehensive Evaluation of Event Semantics for Large Language Models».
<https://doi.org/10.48550/arXiv.2305.15268>.
- [38] VASWANI, ASHISH; SHAZER, NOAM; PARMAR, NIKI; USZKOREIT, JAKOB; JONES, LLION; GOMEZ, AIDAN N.; KAISER, LUKASZ y POLOSUKHIN, ILLIA (2017). «Attention Is All You Need». *CoRR*, **abs/1706.03762**.
<http://arxiv.org/abs/1706.03762>.
- [39] WANG, ALEX; PRUKSACHATKUN, YADA; NANGIA, NIKITA; SINGH, AMANPREET; MICHAEL, JULIAN; HILL, FELIX; LEVY, OMER y BOWMAN, SAMUEL R. (2020). «SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems».
<https://doi.org/10.48550/arXiv.1905.00537>.
- [40] WANG, ALEX; SINGH, AMANPREET; MICHAEL, JULIAN; HILL, FELIX; LEVY, OMER y BOWMAN, SAMUEL R. (2019). «GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding».
<https://doi.org/10.48550/arXiv.1804.07461>.
- [41] WANG, LEI y LIM, EE-PENG (2024). «The Whole is Better than the Sum: Using Aggregated Demonstrations in In-Context Learning for Sequential Recommendation».
- [42] WEI, JASON; BOSMA, MAARTEN; ZHAO, VINCENT Y.; GUU, KELVIN; YU, ADAMS WEI; LESTER, BRIAN; DU, NAN; DAI, ANDREW M. y LE, QUOC V. (2022). «Finetuned Language Models Are Zero-Shot Learners».
- [43] WEI, JASON; TAY, YI; BOMMASANI, RISHI; RAFFEL, COLIN; ZOPH, BARRET; BORGEAUD, SEBASTIAN; YOGATAMA, DANI; BOSMA, MAARTEN; ZHOU, DENNY; METZLER, DONALD; CHI, ED H.; HASHIMOTO, TATSUNORI; VINYALS, ORIOL; LIANG, PERCY; DEAN, JEFF y FEDUS, WILLIAM (2022). «Emergent Abilities of Large Language Models».

- [44] WEI, JASON; WANG, XUEZHI; SCHUURMANS, DALE; BOSMA, MAARTEN; ICHTER, BRIAN; XIA, FEI; CHI, ED; LE, QUOC y ZHOU, DENNY (2023). «Chain-of-Thought Prompting Elicits Reasoning in Large Language Models».
- [45] XIE, YIHUI (2023). *knitr: A General-Purpose Package for Dynamic Report Generation in R*.
<https://yihui.org/knitr/>. R package version 1.45.
- [46] XU, LINGLING; XIE, HAORAN; QIN, SI-ZHAO JOE; TAO, XIAOHUI y WANG, FU LEE (2023). «Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment».
<https://doi.org/10.48550/arXiv.2312.12148>.
- [47] XU, ZIWEI; JAIN, SANJAY y KANKANHALLI, MOHAN (2024). «Hallucination is Inevitable: An Innate Limitation of Large Language Models».
<https://doi.org/10.48550/arXiv.2401.11817>.
- [48] YANG, JINGFENG; JIN, HONGYE; TANG, RUIXIANG; HAN, XIAOTIAN; FENG, QIZHANG; JIANG, HAOMING; YIN, BING y HU, XIA (2023). «Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond».
- [49] ZHANG, ZHIHAO; ZHU, ALAN; YANG, LIJIE; XU, YIHUA; LI, LANTING; PHOTHILIMTHANA, PHITCHAYA MANGPO y JIA, ZHIHAO (2024). «Accelerating Retrieval-Augmented Language Model Serving with Speculation».
<https://arxiv.org/abs/2401.14021>.
- [50] ZHAO, WAYNE XIN; ZHOU, KUN; LI, JUNYI; TANG, TIANYI; WANG, XIAOLEI; HOU, YUPENG; MIN, YINGQIAN; ZHANG, BEICHEN; ZHANG, JUNJIE; DONG, ZICAN; DU, YIFAN; YANG, CHEN; CHEN, YUSHUO; CHEN, ZHIPENG; JIANG, JINHAO; REN, RUIYANG; LI, YIFAN; TANG, XINYU; LIU, ZIKANG; LIU, PEIYU; NIE, JIAN-YUN y WEN, Ji-RONG (2023). «A Survey of Large Language Models».
<https://doi.org/10.48550/arXiv.2303.18223>.
- [51] ZHU, WENHAO; LIU, HONGYI; DONG, QINGXIU; XU, JINGJING; HUANG, SHUJIAN; KONG, LINGPENG; CHEN, JIAJUN y LI, LEI (2023). «Multilingual Machine Translation

with Large Language Models: Empirical Results and Analysis».

<https://doi.org/10.48550/arXiv.2304.04675>.

- [52] ZHUANG, FUZHEN; QI, ZHIYUAN; DUAN, KEYU; XI, DONGBO; ZHU, YONGCHUN; ZHU, HENGSHU; XIONG, HUI y HE, QING (2020). «A Comprehensive Survey on Transfer Learning».

<https://doi.org/10.48550/arXiv.1911.02685>.