



**Universidad de Jaén**

*Escuela Politécnica Superior de Jaén*

# TECNOLOGÍAS DEL LENGUAJE PARA TUTORIZACIÓN DOCENTE CON AGENTES CONVERSACIONALES

Autor: Álvaro Herrera Arjonilla

Grado: Ingeniería Informática

Director: Prof. D. María Teresa Martín Valdivia  
Departamento del director: Departamento de Informática

Fecha: 20/06/2024

Licencia CC



CREAREA



# AGRADECIMIENTOS

Durante estos meses, he podido vivir y disfrutar del culmen de todos estos años de trabajo para finalizar el grado que tanta ilusión me hacía cuando ingresé, y del que salgo totalmente fascinado por las posibilidades que ofrece este hermoso mundo.

Agradezco a todas las personas que me han acompañado durante este largo camino. En especial, a mi familia por haber confiado en mí y haberme permitido estudiar un grado a pesar de todas las dificultades. A Bea, por haber sido siempre mi apoyo y mi calma. A mis amigos del grupo de 'Biblio', a mis amigos de Arjona y a mis compañeros de piso, gracias por haber estado conmigo en los buenos y en los malos momentos.

También quiero agradecer a Maite por haberme dado esta gran oportunidad que me ha permitido aprender y disfrutar en un entorno de trabajo maravilloso junto a los compañeros del grupo SINAI. Gracias, Pablo y Estrella; sin vosotros, esto no hubiera sido posible.



# Índice

<b>Índice de tablas</b>	<b>7</b>
<b>Índice de figuras</b>	<b>8</b>
<b>1 Introducción</b>	<b>9</b>
1.1 Motivación . . . . .	9
1.2 Objetivos . . . . .	10
1.3 ODS (Objetivos de Desarrollo Sostenible) . . . . .	11
<b>2 Planificación del proyecto</b>	<b>14</b>
2.1 Metodología de desarrollo . . . . .	14
2.2 Diagrama de planificación . . . . .	15
<b>3 Presupuesto del proyecto</b>	<b>17</b>
3.1 Recursos humanos . . . . .	17
3.2 Licencias software . . . . .	17
3.3 Hardware . . . . .	18
3.4 Costes indirectos . . . . .	18
3.5 Coste total . . . . .	19
<b>4 Contexto y estudio preliminar</b>	<b>20</b>
4.1 Fase de preparación . . . . .	20
4.1.1 LLM . . . . .	20
4.1.2 Modelización del lenguaje . . . . .	20
4.1.3 Transformer . . . . .	21
4.1.4 Modelos basados en Transformers . . . . .	23
4.1.5 Ingeniería de Prompts . . . . .	25
4.1.6 Embeddings . . . . .	27
4.1.7 Repositorios vectoriales . . . . .	28
4.2 Diseño del modelo . . . . .	29
4.2.1 Comunicación mediante ficheros . . . . .	29

4.2.2	Comunicación mediante API . . . . .	31
4.2.3	Comunicación mediante entorno . . . . .	32
4.3	Elección de personalización . . . . .	33
4.3.1	Modelo con finetuning . . . . .	33
4.3.2	Modelo con RAG . . . . .	35
4.3.3	Decisión para el ajuste del modelo . . . . .	38
4.4	Elección del modelo . . . . .	39
4.4.1	LLaMA . . . . .	40
<b>5</b>	<b>Desarrollo del proyecto</b>	<b>43</b>
5.1	Desarrollo y Pruebas de Integración . . . . .	43
5.2	Problemas resueltos . . . . .	46
5.2.1	Acople de API de Telegram para integración del bot . . . . .	46
5.2.2	Errores iniciales en la generación del texto . . . . .	47
5.2.3	Base de conocimiento . . . . .	49
5.2.4	Embeddings . . . . .	52
5.3	Evaluación y Ajustes . . . . .	53
5.3.1	Parámetros de generación del modelo . . . . .	53
5.3.2	Parámetros del RAG . . . . .	54
5.3.3	Prompting . . . . .	54
5.4	Resultados y discusión . . . . .	58
5.4.1	Estudio y mejora del prompting . . . . .	59
5.4.2	Análisis cualitativo de generación del modelo . . . . .	63
5.4.3	Análisis cualitativo de generación del modelo junto con RAG . . . . .	67
<b>6</b>	<b>Conclusiones</b>	<b>73</b>
6.1	Conclusión final . . . . .	73
6.2	Mejoras futuras . . . . .	75
<b>A</b>	<b>Tablas de resultados</b>	<b>77</b>
A.1	Experimentos . . . . .	77
A.1.1	Análisis cualitativo de generación del modelo . . . . .	78
A.1.2	Análisis cualitativo de generación del modelo junto con RAG . . . . .	80
<b>B</b>	<b>Manual de usuario</b>	<b>82</b>
B.1	Manual de ADA . . . . .	82
B.2	Información básica de conexión . . . . .	82
B.2.1	Entornos Virtuales . . . . .	83
B.2.2	Creación y activación de entornos . . . . .	83

B.2.3	Instalación de librerías en entornos virtuales . . . . .	83
B.2.4	Comandos básicos de Slurm . . . . .	84
B.2.5	Definición de los script de lanzamiento . . . . .	85
B.2.6	Descarga de modelos . . . . .	86
B.2.7	Ejecución . . . . .	87
<b>C</b>	<b>Bibliografía</b>	<b>88</b>

## Índice de tablas

3.1	Coste de recursos humanos . . . . .	17
3.2	Coste de licencias software . . . . .	18
3.3	Coste de hardware . . . . .	18
3.4	Coste total del proyecto . . . . .	19
4.1	Observaciones sobre RAG vs Fine-tuning . . . . .	39
4.2	Evaluación del modelo . . . . .	41
5.1	Dos ejemplos de la generación de texto de mala calidad. . . . .	48
5.2	Listado de libros que forman parte de la base de conocimiento. . . . .	49
5.3	Tabla de ejemplo de texto sin preprocesar . . . . .	51
5.4	Tabla de ejemplo de texto preprocesado . . . . .	51
5.5	Tabla de ejemplo de texto preprocesado corregido . . . . .	52
5.6	Tabla de prueba de prompting. Resultado bueno . . . . .	56
5.7	Tabla de prueba de prompting. Resultado malo . . . . .	57
5.8	Descripción de los parámetros. . . . .	58
5.9	Tabla de estudio y mejora del prompting. Prueba 1. . . . .	60
5.10	Tabla de estudio y mejora del prompting. Prueba 2. . . . .	61
5.11	Tabla de estudio y mejora del prompting. Prueba 3. . . . .	62
5.12	Tabla de análisis cualitativo de generación del modelo. Prueba 1. . . . .	63
5.13	Tabla de análisis cualitativo de generación del modelo. Prueba 2. . . . .	64
5.14	Tabla de análisis cualitativo de generación del modelo. Prueba 3. . . . .	65
5.15	Tabla de análisis cualitativo de generación del modelo. Prueba 4. . . . .	66
5.16	Tabla de análisis cualitativo de generación del modelo junto con RAG. Prueba 1. . . . .	67
5.17	Tabla de análisis cualitativo de generación del modelo junto con RAG. Prueba 2. . . . .	68

5.18	Tabla de análisis cualitativo de generación del modelo junto con RAG. Prueba 3.	70
5.19	Tabla de análisis cualitativo de generación del modelo junto con RAG. Prueba 4.	72
A.1	Tabla de análisis cualitativo de generación del modelo. Anexo(1)	78
A.2	Tabla de análisis cualitativo de generación del modelo. Anexo(2)	79
A.3	Tabla de análisis cualitativo de generación del modelo junto con RAG. Anexo(1)	80
A.4	Tabla de análisis cualitativo de generación del modelo junto con RAG. Anexo(2)	81

## Índice de figuras

1.1	Figura de los ODS.	12
4.1	Una red neuronal recurrente y el despliegue en el tiempo del cálculo implicado en su cálculo hacia delante.	21
4.2	Modelo de arquitectura Transformer.	22
4.3	Atención al producto escalar por puntos (izquierda). La atención multicabezal consiste en varias capas de atención en paralelo (derecha).	23
4.4	Comparativa visual estimada del tamaño de parámetros que conforman distintos LLMs mencionados.	25
4.5	Diagrama de un ejemplo de embedding.	28
4.6	Diagrama simplificado de un repositorio vectorial y sus datos.	28
4.7	Diagrama de secuencia que representa la comunicación mediante ficheros del sistema.	30
4.8	Diagrama de secuencia que representa la comunicación con la API del sistema.	31
4.9	Diagrama de secuencia que representa la comunicación inicial del sistema.	32
4.10	La evolución en el desarrollo de los métodos PEFT en los últimos años	34
4.11	Descripción de tu imagen.	36
4.12	Funcionamiento de modelo con RAG.	37
4.13	Definición formal de RAG-Sequence.	38
4.14	Definición formal de RAG-Token.	38
5.1	Diagrama de secuencia que representa la comunicación del sistema.	47
5.2	Ejemplo de fragmento de un PDF.	50
5.3	Diagrama de la estructura de MPNet.	53

# Capítulo 1

## Introducción

### 1.1. Motivación

En la actualidad, estamos inmersos en un periodo de cambios profundos tanto en el ámbito tecnológico como en el social. La revolución digital está alterando nuestras maneras de vivir y trabajar a un ritmo vertiginoso. La finalidad de esta revolución digital es mejorar la eficacia, eficiencia y competitividad de una organización, lo cual puede involucrar la incorporación de nuevas tecnologías, la instauración de modelos de negocio novedosos y la reorganización de procesos ya establecidos.

Este proyecto se presenta como una estrategia para capitalizar los beneficios y posibilidades que ofrecen determinadas tecnologías disruptivas, como las Tecnologías del Lenguaje (TL), también conocidas como Procesamiento del Lenguaje Natural (PLN), en el contexto universitario. El PLN se encuentra dentro del campo de la Inteligencia Artificial (IA) y se dedica al análisis e interpretación del lenguaje humano por parte de las máquinas. Esta labor se emplea en una amplia gama de aplicaciones, incluyendo la búsqueda y extracción de información, la traducción automática, el reconocimiento de voz y la generación de respuestas automatizadas. En el ámbito del PLN, se emplean técnicas de análisis y procesamiento de texto, tales como el análisis morfológico, sintáctico y semántico. Además, se hacen uso de modelos de lenguaje y algoritmos de aprendizaje automático para entrenar sistemas informáticos capaces de comprender y procesar el lenguaje humano.

El proyecto pretende crear un sistema computacional basado en PLN que emule una interacción conversacional similar a la de un profesor virtual con un estudiante. Estos sistemas son conocidos como agentes conversacionales. Nuestro sistema, denominado TutorBot, tendrá la responsabilidad de recopilar consultas y cuestiones planteadas por los estudiantes en una materia específica, además de guiarlos a través de los contenidos del curso con el fin de abordar y solucionar sus inquietudes.

Los estudiantes de hoy en día son considerados nativos digitales, individuos que han crecido

inmersos en un entorno tecnológico y han adquirido habilidades en su uso desde una edad temprana. Al aprovechar estas competencias digitales específicas en un proyecto de enseñanza, se abrirán oportunidades para desarrollar nuevas metodologías educativas. Asimismo, los hábitos de comportamiento de los estudiantes y de la sociedad en general están experimentando cambios debido a la amplia disponibilidad de tecnología y a la continua conectividad que ofrece Internet. Esta evolución ha generado una cultura marcada por la permanente disponibilidad y conexión, provocando cambios significativos en los hábitos y expectativas de las personas en relación con la realización de sus actividades diarias.

Si consideramos a algún estudiante preparando alguna de sus asignaturas, es comprensible que puedan surgir situaciones en las que enfrenten preguntas o dudas que no puedan resolver de inmediato. A pesar de que siempre existe la opción de solicitar asistencia al profesor, muchas veces los estudiantes no recurren a este servicio por diversas razones. En este contexto, es evidente que una herramienta de este tipo representaría un notable avance en el apoyo educativo a los estudiantes.

También resultará útil la retroalimentación que el profesor obtenga a partir del uso que los estudiantes hagan de la herramienta, llegando a conocer las preguntas planteadas por los alumnos con mayor asiduidad. Esta información permitirá al docente preparar material didáctico más efectivo, profundizar en conceptos que puedan presentar dificultades para los estudiantes, organizar sesiones colectivas de tutoría, entre otras acciones que mejoren el proceso educativo.

En este trabajo nos vamos a centrar en intentar resolver las dudas básicas de los alumnos utilizando un sistema de diálogo basado en agentes conversacionales mediante algoritmos de PLN que especializaremos usando tecnologías del lenguaje. De esta manera, TutorBot será una herramienta que pueden utilizar (i) los alumnos para resolver sus dudas en tiempo real, en cualquier lugar, desde cualquier dispositivo y a cualquier hora, y (ii) el profesorado, para recabar información sobre las dificultades que encuentra el alumnado cuando se enfrenta al estudio de una asignatura. Así, el profesor podrá usar esta información bien para mejorar su material docente, bien para planificar alguna tutoría de refuerzo, bien para incidir en los conceptos que parece que no han quedado claros. . .

## 1.2. Objetivos

El objetivo principal es utilizar herramientas de aprendizaje automático y técnicas de PLN para la implementación de un agente conversacional. Concretamente se pretende desarrollar un prototipo que a partir de la información de asignaturas del grado de Ingeniería Informática sea capaz de diseñar e implementar un chatbot para responder a preguntas del alumnado.

Los objetivos principales con los que se planteó el trabajo fueron:

1. Estudio y análisis manual de las distintas fuentes de datos que se van a tratar
2. Diseño e implementación de chatbot en Telegram
3. Puesta en marcha del chatbot
4. Recopilación de las interacciones realizadas
5. Evaluación de los resultados y análisis de errores
6. Redacción de una memoria que recoja todo el trabajo desarrollado así como los manuales de instalación y de usuario.

Se plantean los siguientes objetivos específicos:

- Adaptar el contenido curricular por asignatura de modo que sea computacionalmente accesible por el chatbot.
- Implementar un chatbot para la asignatura piloto utilizando tecnologías del lenguaje.
- Realizar un informe con las consultas realizadas por parte del alumno que se pondrá a disposición del profesor.
- Chequear la eficiencia de utilización de técnicas de inteligencia artificial en el proceso de enseñanza-aprendizaje.
- Comprobar la satisfacción con el uso de esta nueva herramienta.

### 1.3. ODS (Objetivos de Desarrollo Sostenible)

Los Objetivos de Desarrollo Sostenible (ODS) son un conjunto de 17 objetivos mundiales establecidos por las Naciones Unidas en 2015 como parte de la Agenda 2030 para el Desarrollo Sostenible. Estos objetivos abarcan una amplia gama de áreas que incluyen la erradicación de la pobreza, la protección del medio ambiente, la promoción de la igualdad de género, el acceso a la educación y la mejora de la salud, entre otros.

Es importante conocer los ODS porque proporcionan un marco integral para orientar las acciones a nivel mundial hacia un desarrollo sostenible y justo. Conocerlos nos ayuda a entender los problemas globales y a trabajar en soluciones colaborativas, tanto a nivel individual como colectivo, para construir un futuro más próspero, equitativo y sostenible para todos. Además, los ODS sirven como herramienta de medición y seguimiento del progreso hacia un mundo mejor.



Figura 1.1: Figura de los ODS.

En la búsqueda de cumplir con el Objetivo de Desarrollo Sostenible número 4, que apunta a garantizar una educación inclusiva, equitativa y de calidad para todos, las instituciones educativas deberían enfocarse en innovar y adaptarse a las necesidades cambiantes de los estudiantes. En este contexto, la implementación de herramientas tecnológicas, como un agente generador de texto que tutoriza las dudas de los alumnos universitarios, se presenta como una solución clave para alcanzar este objetivo.

Este agente generador de texto actúa como un recurso educativo accesible y disponible en todo momento para los estudiantes. Permite que los alumnos puedan acceder a respuestas claras y precisas a sus preguntas en cualquier momento y desde cualquier lugar, lo que elimina barreras de tiempo y espacio en el proceso de aprendizaje. De esta manera, se promueve la inclusión al garantizar que todos los estudiantes tengan acceso a la información que necesitan

para su formación académica.

Además, el agente generador de texto puede adaptarse a las necesidades individuales de los estudiantes, brindando una experiencia de aprendizaje personalizada. Al ofrecer respuestas y explicaciones adaptadas a cada consulta, se fomenta el desarrollo de habilidades de pensamiento crítico y resolución de problemas, aspectos fundamentales de una educación de calidad.

Asimismo, esta herramienta tecnológica contribuye a la equidad al reducir las disparidades en el acceso a la información y al apoyo académico. Independientemente del origen geográfico, el nivel socio-económico o las limitaciones de tiempo de los estudiantes, todos tienen la oportunidad de recibir orientación y clarificación sobre los temas que están estudiando.

Por tanto, esta herramienta no solo facilita el acceso a la información y el apoyo académico, sino que también impulsa la autonomía, la personalización del aprendizaje y la igualdad de oportunidades en el ámbito educativo.

Por todo lo anterior, consideramos que este trabajo fin de grado se encuentra completamente alineado con el ODS 4 centrado en la consecución de una educación inclusiva, equitativa y de calidad.

# Capítulo 2

## Planificación del proyecto

En este capítulo se tratará la organización temporal del proyecto, así como las iteraciones del mismo y la representación gráfica del plan mediante un diagrama de Gantt.

### 2.1. Metodología de desarrollo

El desarrollo de TutorBot ha sido impulsado por la metodología ágil, una filosofía que se adapta perfectamente al dinámico mundo del software. La esencia de la metodología ágil reside en su enfoque iterativo e incremental, destacando la flexibilidad, la colaboración y la satisfacción del usuario como pilares fundamentales. En este proceso, el proyecto se divide en segmentos manejables conocidos como sprints, con una duración típica de dos a cuatro semanas.

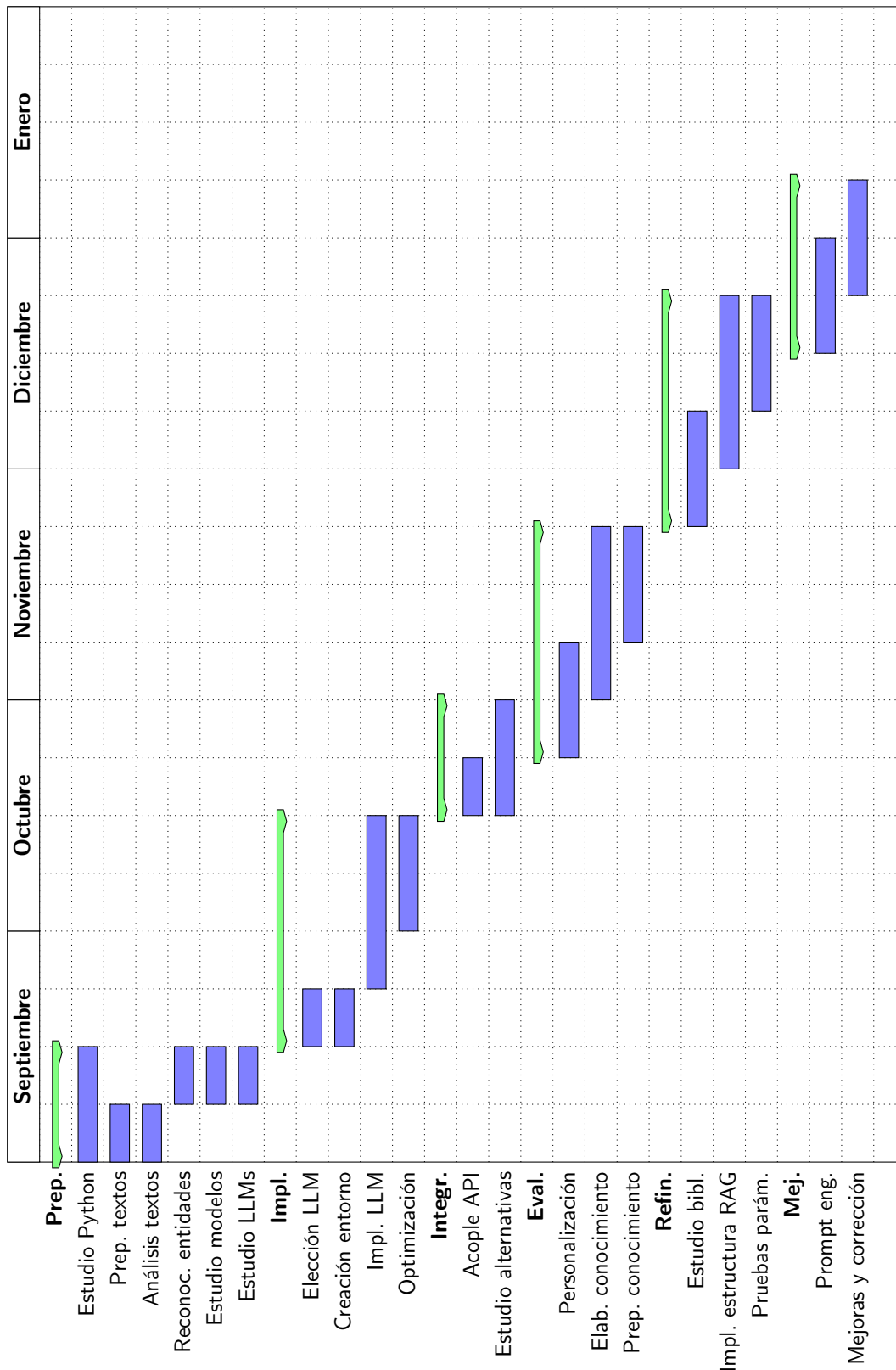
Siguiendo la metodología SCRUM específicamente para TutorBot, se establecen reuniones periódicas con la tutora cada semana o cada dos semanas. Estas sesiones son cruciales para exponer los avances en las tareas asignadas, someter la aplicación a pruebas exhaustivas en busca de posibles fallos o áreas de mejora, y discutir los siguientes pasos o tareas a realizar. Este enfoque colaborativo asegura una retroalimentación constante y permite ajustes rápidos, garantizando así la calidad del producto final y la alineación con las expectativas del proyecto en este caso.

#### 1. Fase de preparación (2 semanas)

- Estudio de Python (2 semanas)
- Estudio de preprocesamiento de textos (1 semana)
- Análisis semántico, morfológico y sintáctico de textos preprocesados (1 semana)
- Reconocimiento de entidades de textos (1 semana)

- Estudio de modelos de aprendizaje supervisado y no supervisado de textos (1 semana)
  - Estudio de Large Language Models (LLMs) (1 semana)
- 2. Implementación inicial (4 semanas)**
- Elección de LLM a usar (1 semana)
  - Creación de entorno de desarrollo y acceso a clúster ADA (1 semana)
  - Implementación y prueba de ejecuciones del LLM (3 semanas)
  - Optimización y medida de tiempos de ejecución del modelo (2 semanas)
- 3. Desarrollo y Pruebas de Integración (1 semana)**
- Acople de API de Telegram para integración del bot (1 semana)
  - Estudio de alternativas para el uso del modelo (1 semana)
- 4. Evaluación y Ajustes (2 semanas)**
- Estudio de personalización y mejora de la calidad de las respuestas (1 semana)
  - Elaboración de base de conocimiento con libros de la materia a responder (2 semanas)
  - Preprocesamiento de la base de conocimiento (1 semana)
- 5. Refinamiento de Funcionalidades (2 semanas)**
- Estudio de bibliotecas para desarrollo de LLMs (1 semana)
  - Implementación de estructura RAG junto al LLM y pruebas de ejecución (2 semanas)
  - Pruebas de cambios de parámetros de ajuste para la mejora de la calidad de las respuestas (1 semana)
- 6. Análisis y Mejoras Finales (3 semanas)**
- Prompt engineering (1 semana)
  - Mejoras y corrección de errores del modelo (3 semanas)

## 2.2. Diagrama de planificación



## Capítulo 3

# Presupuesto del proyecto

En este capítulo de la memoria se presentará el presupuesto destinado al proyecto, abarcando los gastos relacionados con los recursos humanos empleados, así como las licencias software y hardware necesario.

### 3.1. Recursos humanos

El proyecto consta de tres trabajadores: un desarrollador con poca experiencia, un desarrollador con una experiencia media que lo supervisa y un jefe de proyecto. Se especifican las siguientes cifras salariales en la tabla 3.1 basadas en los datos arrojados por el Boletín Oficial del Estado del XVIII Convenio colectivo estatal de empresas de consultoría, tecnologías de la información y estudios de mercado y de la opinión pública, resolución publicada el 13 de julio de 2023:

Rol	Salario Anual (€)	Salario Mensual (€)	Meses	Total (€)
Jefe de Proyecto	25.241,94	2.103,50	5	10.517,50
Desarrollador Junior	17.168,70	1.430,72	5	7.153,60
<b>TOTAL</b>				<b>17.671,10</b>

Tabla 3.1: Coste de recursos humanos

### 3.2. Licencias software

Sólo se requieren licencias para los computadores implicados en el proyecto, en este caso tres. Se procede al desglose de costes (por unidad) en la tabla 3.2:

Software	Cantidad	Coste (€)
Microsoft Windows 10 Home	3	139,98
<b>TOTAL</b>		<b>419,94</b>

Tabla 3.2: Coste de licencias software

### 3.3. Hardware

Se requieren dos ordenadores, uno para cada miembro del proyecto y se empleará el clúster denominado ADA, el cual pertenece al grupo de investigación SINAI. A continuación, se detallan los costos desglosados por unidad en la tabla 3.3:

Hardware	Cantidad	Coste (€)
iMac (Chip M3, 512 GB SSD, 16 GB RAM)	2	2.079,00
Clúster ADA (Nodo Tesla Volta V100)*	2	9.727,01
Clúster ADA (Procesador Xeon Silver 4208)*	2	723,03
<b>TOTAL</b>		<b>14.608,04</b>

Tabla 3.3: Coste de hardware

\*El cálculo del costo de este nodo del clúster se ha realizado de manera aproximada, determinando el precio individual de cada componente que integra dicho nodo y sumando el costo total de todos los componentes. Las especificaciones de cada nodo del clúster se encuentran disponibles en el siguiente enlace: <https://www.ujaen.es/centros/ceatic/servicios/supercomputacion/cluster-ada>

### 3.4. Costes indirectos

Se requieren fondos destinados a imprevistos y gastos indirectos durante el desarrollo, garantizando que no impacten en el presupuesto acordado, así como para el mantenimiento de los equipos, las instalaciones de trabajo, los desplazamientos y los servidores. El fondo destinado al mantenimiento corresponderá al 10 % de los gastos mencionados anteriormente.

**TOTAL: 3.269,90**

### 3.5. Coste total

Sección	Coste (€)
Recursos humanos	17.671,10
Licencias software	419,94
Hardware	14.608,04
Mantenimiento	3.269,90
<b>TOTAL</b>	<b>35.968,98</b>

Tabla 3.4: Coste total del proyecto

# Capítulo 4

## Contexto y estudio preliminar

### 4.1. Fase de preparación

Esta iteración corresponde a la ejecución de los pasos iniciales antes de comenzar el desarrollo del software de TutorBot, llevando a cabo un estudio de las distintas herramientas propias del PLN y del diseño del modelo construido.

Se realizó un estudio de las distintas herramientas de PLN [1] para tener todas ellas en cuenta para el proyecto. Se vieron herramientas como tokenization (división de oraciones en tokens), stemming (reducción de palabras a su raíz), stop words (eliminación de palabras vacías) y reconocimiento de entidades. Además de distintos tipos de análisis de textos y de modelos de aprendizaje.

Tras realizar este estudio introductorio se realizó un estudio sobre LLMs para conocer sus características y funcionamiento.

#### 4.1.1. LLM

Los LLM (Large Language Model) son grandes modelos basados en arquitecturas de redes neuronales profundas conocidas como “Transformers”. Estos modelos están preentrenados para realizar tareas de procesamiento del lenguaje natural. Estas incluyen tareas a nivel de oración, como la inferencia de lenguaje natural [2] y la paráfrasis [3], que buscan predecir las relaciones entre oraciones mediante un análisis holístico, así como tareas a nivel de token como el reconocimiento de entidades nombradas y la respuesta a preguntas, donde se requiere que los modelos produzcan resultados detallados a nivel de token [4][5].

#### 4.1.2. Modelización del lenguaje

Profundizando en la modelización del lenguaje, esta, suele ser enmarcada como la estimación no supervisada de la distribución a partir de un conjunto de ejemplos  $(x_1, x_2, \dots, x_n)$ ,

cada uno compuesto por secuencias de símbolos de longitud variable  $(s_1, s_2, \dots, s_n)$ . Dado que el lenguaje tiene un orden natural secuencial, es común factorizar las probabilidades conjuntas sobre los símbolos como el producto de las probabilidades condicionales:

$$p(x) = \prod_{i=1}^n p(s_i | s_1, \dots, s_{i-1})$$

Este enfoque permite un muestreo y estimación tratables de  $p(x)$  y de cualquier condicional de la forma  $p(s_{n-k}, \dots, s_n | s_1, \dots, s_{n-k-1})$ . En los últimos años, ha habido mejoras significativas en la expresividad de los modelos que pueden calcular estas probabilidades condicionales, como las arquitecturas de autoatención como el Transformer [6]. Estos modelos han revolucionado el ámbito del PLN, logrando comprender y producir texto con una coherencia y precisión que se asemejan cada vez más a las habilidades humanas.

### 4.1.3. Transformer

En esta sección, se define la arquitectura Transformer como el fundamento principal de los Modelos de Lenguaje con Aprendizaje Profundo (LLMs), resaltando algunos ejemplos destacados aunque primero, se introduce el concepto de Transformers mediante las redes neuronales recurrentes.

Previamente a los Transformers, las redes neuronales recurrentes se establecieron como enfoques de vanguardia en la modelización de secuencias y problemas de transducción, como la modelización del lenguaje y la traducción automática.

Los modelos recurrentes generalmente descomponen la computación a lo largo de las posiciones de símbolos de las secuencias de entrada y salida. Al alinear las posiciones con pasos en el tiempo de computación, generan una secuencia de estados ocultos, como función del estado oculto anterior y la entrada para la posición  $p$ . Esta naturaleza secuencial impide la paralelización dentro de ejemplos de entrenamiento, lo cual es crítico para longitudes de secuencia mayores, ya que las limitaciones de memoria restringen el agrupamiento entre ejemplos.

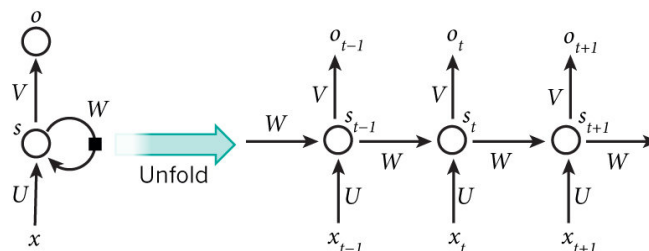


Figura 4.1: Una red neuronal recurrente y el despliegue en el tiempo del cálculo implicado en su cálculo hacia adelante.

En 2017, se realizaron numerosos esfuerzos para ampliar los límites de los modelos de lenguaje recurrentes y las arquitecturas codificador-decodificador presentando la arquitectura Transformer.

Transformer es una arquitectura de modelo que evita la recurrencia y en su lugar depende completamente de un mecanismo de atención para establecer dependencias globales entre la entrada y la salida. El Transformer permite una paralelización significativamente mayor a modelos y trabajos anteriores y puede alcanzar mejores resultados respecto a esos modelos.

La mayoría de los modelos competitivos de transducción de secuencias neuronales tienen una estructura codificador-decodificador. Aquí, el codificador mapea una secuencia de representaciones de símbolos  $(x_1, \dots, x_n)$  a una secuencia de representaciones continuas  $z = (z_1, \dots, z_n)$ . Dado  $z$ , el decodificador genera una secuencia de salida  $(y_1, \dots, y_m)$  de símbolos uno a la vez. En cada paso, el modelo es auto-regresivo, consumiendo los símbolos generados previamente como entrada adicional al generar el siguiente.

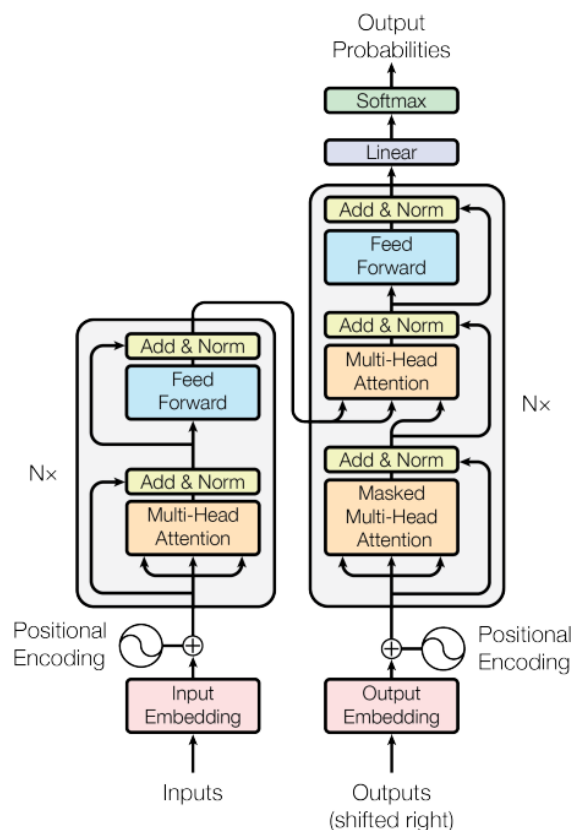


Figura 4.2: Modelo de arquitectura Transformer.

El Transformer presenta una arquitectura general, como se muestra en la Figura 4.2, que utiliza capas de autoatención y capas totalmente conectadas para ambos el codificador y el decodificador.

En cuanto a las pilas de codificador y decodificador, el codificador consta de una pila de 6 capas idénticas. Cada capa incorpora dos subcapas: una de autoatención multi-cabeza y otra de red de alimentación conectada basada en la posición. La utilización de conexiones residuales y normalización de capa garantiza la eficiencia del modelo.

Por otro lado, el decodificador también está compuesto por una pila de 6 capas idénticas. Además de las dos subcapas presentes en cada capa del codificador, se introduce una tercera subcapa que realiza atención multi-cabeza sobre la salida del codificador. Al igual que en el codificador, se aplican conexiones residuales y normalización de capa. Se realiza una modificación en la subcapa de autoatención del decodificador para evitar la atención a posiciones futuras, asegurando que las predicciones para una posición  $i$  dependan únicamente de las salidas conocidas en posiciones menores a  $i$ .

En cuanto al mecanismo de atención, este se define como el mapeo de una consulta y un conjunto de pares clave-valor a una salida, calculando pesos ponderados mediante una función de compatibilidad entre la consulta y la clave correspondiente.

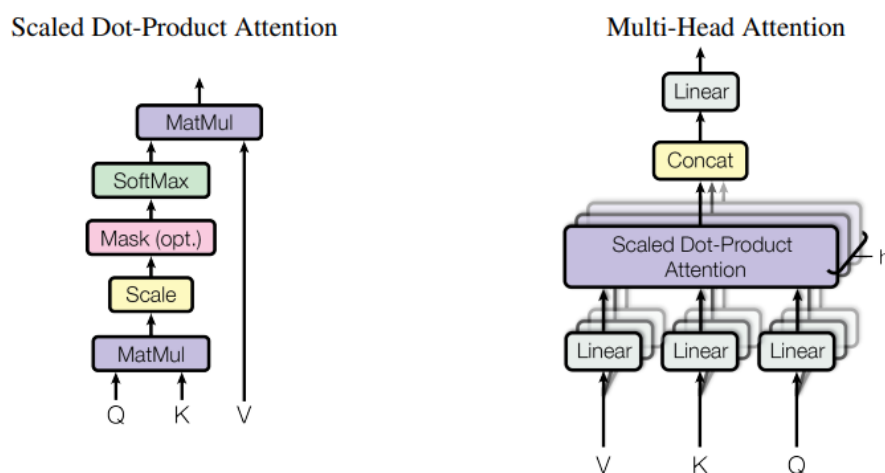


Figura 4.3: Atención al producto escalar por puntos (izquierda). La atención multicabezal consiste en varias capas de atención en paralelo (derecha).

#### 4.1.4. Modelos basados en Transformers

Desde la introducción de la arquitectura Transformers, han surgido diversos modelos que han sido desarrollados y puestos en práctica. La disponibilidad de estos modelos varía, algunos están disponibles de forma gratuita para el público, mientras que otros son proporcionados por empresas que requieren pago para su utilización. A continuación, se muestran ejemplos de modelos que han tenido un impacto significativo en el desarrollo de los LLMs.

Desde la aparición de la arquitectura BERT (*Bidirectional Encoder Representations from Transformers*) [7] se ha destacado como un modelo basado en la arquitectura Transformer que logró un avance significativo. Su innovación clave radica en su capacidad para entender el contexto de una palabra no solo en función de su ubicación en una oración, sino también en oraciones adyacentes, gracias a su entrenamiento en la tarea de autorellenado de oraciones. Este enfoque bidireccional marcó un hito en el procesamiento y comprensión del lenguaje natural, generando un impacto notable en el campo. La destacada eficacia de BERT en diversas tareas de PLN ha motivado a que investigadores intenten explorar sus capacidades y desarrollar variantes especializadas. Además, BERT ha catalizado la creación de una familia diversa de modelos derivados con enfoques más específicos para diferentes aplicaciones.

Estos modelos de Transformers, reconocidos por su capacidad bidireccional de codificación y decodificación, han impulsado grandes mejoras de rendimiento en aplicaciones como la clasificación de texto, la traducción automática y las respuestas a preguntas. En contraste, los modelos GPT (*Generative Pre-trained Transformers*) se distinguen por su enfoque de generación autónoma y fluida de texto. Ejemplificados por GPT-3 y GPT-4 de OpenAI[8], PaLM de Google [9], LLaMA de Meta [10], estos modelos destacan por su capacidad para prever palabras o secuencias de texto basándose en el contexto proporcionado. Utilizan una estructura de codificación-decodificación que favorece la generación coherente y contextualmente relevante, permitiéndoles abordar una variedad de tareas lingüísticas de manera más generativa y flexible.

Aunque modelos como GPT-4 y PaLM son de propiedad privada y no están disponibles para ajustes externos, han demostrado resultados impresionantes en la generación de lenguaje, incluida la capacidad para crear contranarrativas o generación de resúmenes. Su eficacia ha alcanzado niveles notables, superando incluso la capacidad humana en ciertas instancias. A pesar de estos logros destacados, estos modelos enfrentan desafíos como la opacidad en sus procesos de toma de decisiones, la necesidad de grandes cantidades de datos para entrenamiento, la generación de sesgos inherentes a los datos de entrenamiento y la comprensión limitada de las relaciones causales [11]. Estos problemas plantean cuestionamientos importantes sobre la ética, la interpretabilidad y la responsabilidad en el uso de la inteligencia artificial en general y los LLMs en particular.

En cambio, los modelos de código abierto, como LLaMA o Mistral, ofrecen accesibilidad pública sin restricciones de licencias ni costos adicionales, estimulando la innovación y personalización. La transparencia y auditabilidad de los LLMs de código abierto permiten a los investigadores comprender su funcionamiento, identificar sesgos y realizar ajustes para adaptarse a contextos específicos. Estas modificaciones resultan cruciales para mitigar sesgos inherentes, promoviendo soluciones éticas y equitativas.

En este contexto, la siguiente figura destaca el tamaño de los modelos mencionados previamente. Desde BERT hasta los más recientes GPT-4, PaLM, LLaMA y Mistral, se

proporciona una visualización comparativa de sus dimensiones.



Figura 4.4: Comparativa visual estimada del tamaño de parámetros que conforman distintos LLMs mencionados.

El tamaño de un modelo de lenguaje se refiere al número de parámetros que tiene. Los parámetros son los pesos asociados con las conexiones entre las neuronas en la red neuronal del modelo. En términos simples y de manera genérica, un modelo con más parámetros tiene una capacidad de representación más grande y puede capturar patrones más complejos en los datos.

#### 4.1.5. Ingeniería de Prompts

Un prompt en el ámbito de la inteligencia artificial y los chatbots es una manera de interactuar a través de texto entre un humano y un modelo lingüístico que permite al modelo generar una salida. Esta interacción puede adoptar la forma de una pregunta, párrafos de texto, fragmentos de código o ejemplos.

Un modelo de IA puede proporcionar varios resultados en función de cómo se formule la pregunta. El objetivo de la pregunta es proporcionar al modelo lingüístico información suficiente

para que pueda producir resultados pertinentes a la pregunta. El prompting ofrece una interfaz natural e intuitiva para que los humanos interactúen y utilicen los LLMs.

Sin embargo, los LLM requieren una cuidadosa ingeniería de prompts, ya sea manual o automática, ya que los modelos no parecen entender los prompts del mismo modo que lo haría un humano. Aunque muchos métodos exitosos de ajuste de prompts realizan la optimización sobre un espacio continuo utilizando métodos basados en gradientes, esto se vuelve menos práctico conforme más grandes son los modelos, ya que el cálculo de gradientes se vuelve cada vez más costoso.

## Tipos de Prompts

Comenzaremos definiendo cada tipo de prompt y discutiendo sus implicaciones en el proceso de interacción de modelos del lenguaje. Esta exploración nos permitirá comprender mejor cómo los prompts afectan la capacidad de los modelos de adaptarse a diferentes contextos y tareas [12].

### ■ Prompting Directo (Zero-shot)

El prompting directo (también conocido como "Zero-shot") es el tipo de orientación más sencillo. No proporciona ejemplos al modelo, solo la instrucción. También puede formular la instrucción como una pregunta o dar al modelo un 'rol'. Proporciona:

- Instrucción
- Contexto

### ■ Prompting con Ejemplos (One-shot y Few-shot)

#### ● One-shot

El 'one-shot' prompt implica proporcionar un solo ejemplo al modelo. Este ejemplo se utiliza como guía para la generación de resultados similares en futuras tareas. Es una forma eficaz de estimular al modelo con un caso específico para que pueda generar resultados relacionados. En el ejemplo mostrado justo abajo se aprecia un prompt que se usaría para que el modelo genere resúmenes de cuentos con el mismo formato:

Quiero que cuando te introduzca un texto lo proceses y me lo devuelvas con este mismo formato. Te doy un ejemplo:

*Título: El Secreto del reloj de arena*

*Resumen: En una ciudad costera olvidada por el tiempo, un relojero solitario descubre un misterioso reloj de arena que parece cambiar la*

*realidad misma. Cuando decide investigar su origen, se ve envuelto en una trama de intriga que desafía las leyes del tiempo y el espacio.*

*Personajes principales:*

- *Elijah Flint: Un relojero talentoso y reservado que encuentra el reloj de arena.*
- *Sophia Hart: Una historiadora local apasionada por los misterios de la ciudad.*
- *Dr. Marcus Wells: Un científico excéntrico obsesionado con el tiempo y sus manipulaciones.*

#### ● **Few-shot**

El prompt 'few-shot' amplía el concepto de one-shot al permitir que el modelo reciba algunos ejemplos adicionales, aunque en una cantidad limitada (generalmente un número bajo, como entre tres y algunos cientos) para guiar su capacidad de generalización y aprendizaje. En el siguiente ejemplo podemos ver un prompt que se usaría para que el modelo aprenda a clasificar frases a partir de unos pocos ejemplos con el mismo formato:

Clasifica la frase que falta como 'Positivo' o como 'Negativo':

- *"Gran producto, 10/10: Positivo"*
- *"No funcionó muy bien: Negativo"*
- *"Super útil, merece la pena: Positivo"*
- *"No funciona: "*

Cuando se ejecuta esta pregunta, la respuesta del modelo será clasificar 'No funciona' como positivo o negativo, como se muestra en los ejemplos.

### **4.1.6. Embeddings**

Un embedding es una forma de representar un objeto de datos complejo, como un vector, en un espacio de dimensión inferior conservando sus propiedades esenciales [13]. En el contexto del PLN, los embeddings se utilizan para representar palabras, frases o documentos en un formato numérico que pueda ser procesado por algoritmos de aprendizaje automático.

Formalmente, un embedding es una función que asigna un objeto de datos de entrada, como una palabra o un documento, a un punto en un espacio vectorial de dimensiones inferiores. El vector resultante, denominado vector de embedding, captura las características esenciales del objeto de datos de entrada de forma compacta y eficiente. Las dimensiones del vector de embedding suelen elegirse pequeñas en comparación con el espacio de datos original, lo que reduce la complejidad de la representación al tiempo que preserva la información más importante.

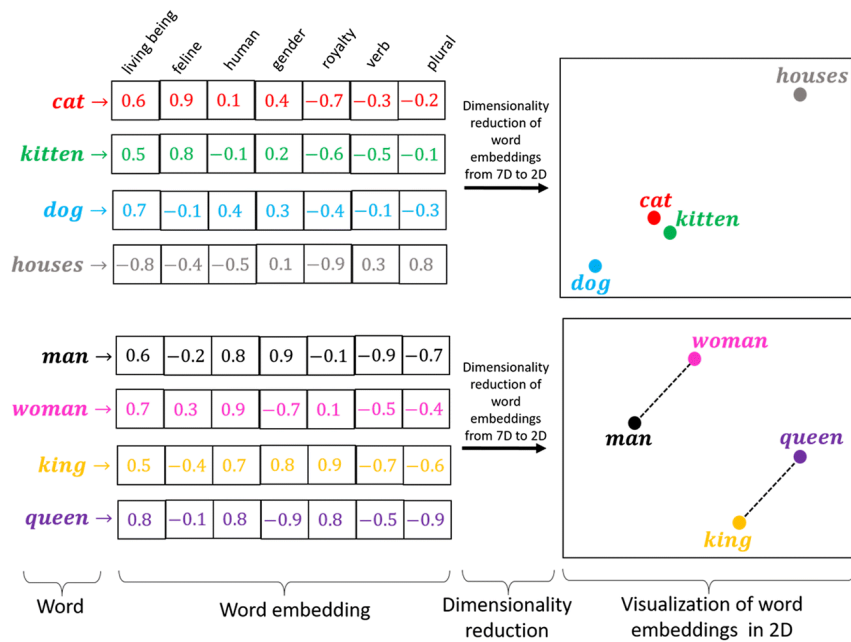


Figura 4.5: Diagrama de un ejemplo de embedding.

### 4.1.7. Repositorios vectoriales

Los repositorios vectoriales son bases de datos diseñadas específicamente para almacenar y recuperar embedding vectoriales de forma eficiente. Son necesarios porque las bases de datos tradicionales, como SQL, no están optimizadas para almacenar y consultar grandes volúmenes de datos vectoriales. Los repositorios vectoriales pueden indexar y buscar rápidamente vectores similares mediante algoritmos de similitud. Esto permite a las aplicaciones encontrar vectores relacionados a partir de una consulta de vectores objetivo.

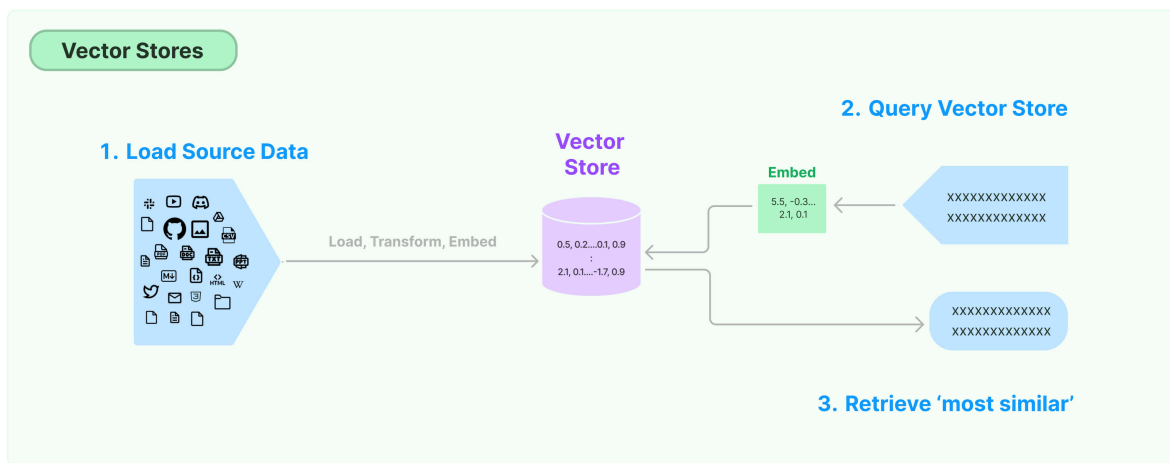


Figura 4.6: Diagrama simplificado de un repositorio vectorial y sus datos.

En el caso de un chatbot personalizado, se puede hacer que un modelo busque textos similares en una colección de documentos mediante un algoritmo de búsqueda de similitudes y que la información resultante se utilice para generar una respuesta altamente personalizada y precisa. Esto es posible gracias a los embeddings y la indexación vectorial en repositorios de vectores.

## 4.2. Diseño del modelo

En cuanto al diseño de la solución, al usar una metodología SCRUM, el diseño y la implementación del mismo fueron variando y adaptándose a los problemas que surgían durante el proceso de desarrollo. Por lo que, se plantearon distintas opciones y esquemas para la solución. En esta sección, se dará muestra de las distintas soluciones planteadas, su contexto, sus problemas, sus ventajas y las limitaciones que plantean al equipo de desarrollo.

Inicialmente, la comunicación entre usuario y modelo será simple. El diseño inicial consiste en un modelo del lenguaje y un usuario. Se carga el modelo y se ejecuta, entonces, recibe un texto a través de una variable que contiene las preguntas del usuario. Tras recibirlo comienza el proceso de elaborar una respuesta que finalmente se mostrará por pantalla.

El problema del diseño inicial surge en el momento en el que es necesario implementar una interfaz para los usuarios. La interfaz se había planteado previamente como objetivo que fuera una propia de Telegram en formato bot. Para su implementación, se iba a usar la librería Telebot pero se encontró el primer gran problema del proyecto, el clúster ADA tiene solo un nodo con conexión a Internet, el llamado nodo principal. Este nodo no cuenta con una GPU incorporada por lo que, no es viable ejecutar y entrenar un modelo del lenguaje en el mismo porque las GPUS están diseñadas para realizar cálculos intensivos y paralelos con mejor rendimiento que las CPUs.

Tras descubrir este problema, se estudiaron diferentes planteamientos para dar solución a la forma de tener una conexión con la API (*Application Programming Interface*) de Telegram. A continuación, se describen las distintas soluciones que se han planteado.

### 4.2.1. Comunicación mediante ficheros

Esta posible solución tiene el modelo cargado en ADA con todas sus ventajas y sus problemas como se planteaba inicialmente. Dado que esto presenta el problema de comunicación mencionado recientemente se plantea una solución. Este primer planteamiento resulta ser el uso de un fichero de texto que a través del nodo principal manda información a un fichero que se encuentra en el nodo del modelo e internamente se comunican para transmitir la información sobre los nuevos mensajes y el usuario asociado al mensaje. Una vez que el fichero

que se encuentra en el nodo del modelo recibe la información, cada mensaje del usuario se añadirá al contenido existente previamente escrito.

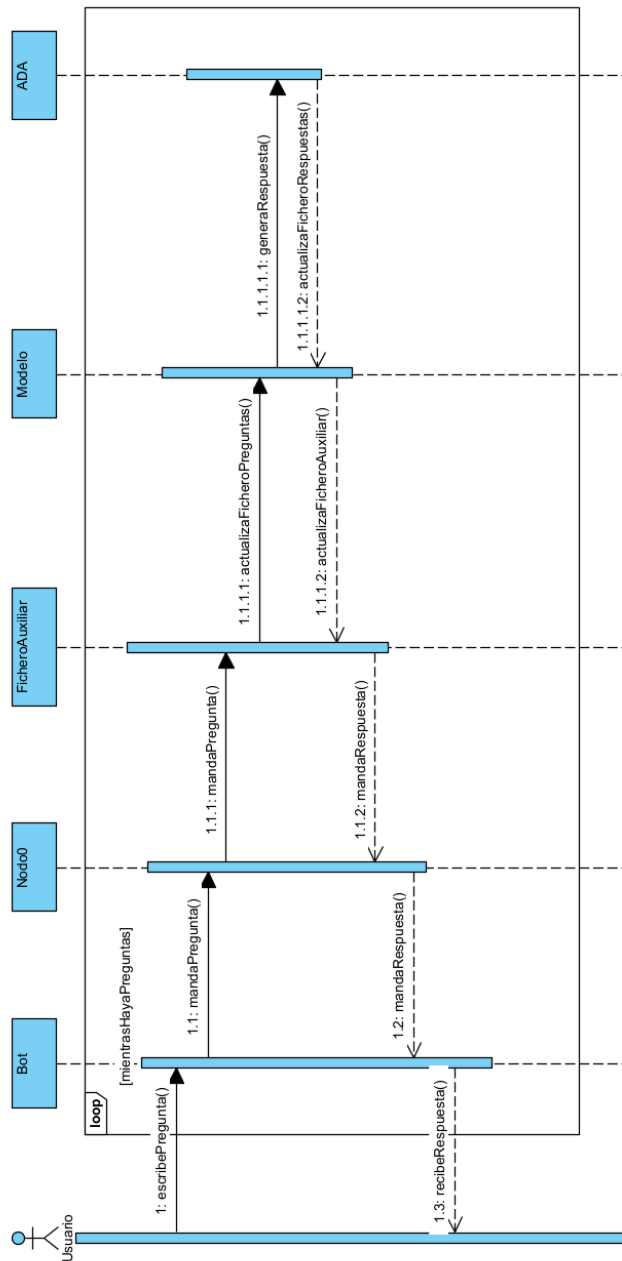


Figura 4.7: Diagrama de secuencia que representa la comunicación mediante ficheros del sistema.

Esta alternativa tiene problemas y es que resulta muy compleja su implementación y es propensa a fallos ya que, habría que gestionar el acceso a ficheros cuidadosamente y la

distribución de la información que contienen los ficheros sería complicada de gestionar para aplicarse al modelo. El sistema no sería tan fiable ya que, interrupciones inesperadas podrían dejar el sistema en un estado inconsistente. A pesar de todo, si se mejora podría ser una buena solución de cara a problemas futuros.

#### 4.2.2. Comunicación mediante API

La siguiente posible solución presenta un modelo que no está cargado y ejecutándose en el clúster ADA. En este caso, se utiliza un LLM a través de una API que estaría directamente comunicándose con el sitio donde alojemos el resto del código. La implementación del código es sencilla, el modelo se comunica usando una clave (API-KEY) que el proveedor de la API del modelo asigna al desarrollador que contrata este servicio.

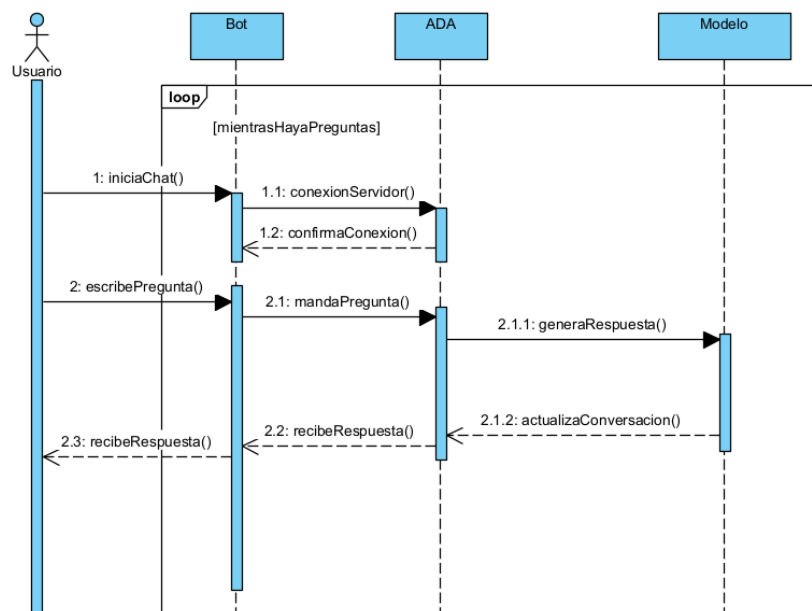


Figura 4.8: Diagrama de secuencia que representa la comunicación con la API del sistema.

El problema de esta solución es que presenta un sobrecoste importante para el proyecto ya que, las APIs que hay disponibles en el mercado cuestan dinero por su uso debido al coste computacional de este tipo de modelos. Además, algunos proveedores imponen límites en la cantidad de solicitudes o el volumen de datos que se pueden enviar, lo que podría afectar la escalabilidad y el rendimiento del sistema.

Otra consecuencia de esta solución sería no tener el modelo cargado localmente lo que significa menos control sobre su configuración y comportamiento. Por lo tanto, los ajustes personalizados o la adaptación del modelo a necesidades específicas podrían ser limitados de

cara a personalizar el modelo con diferentes técnicas de ajuste del modelo.

Esta solución requiere una conexión constante a Internet para comunicarse con la API. Si la conexión falla, el sistema podría dejar de funcionar o generar respuestas inesperadas. La solución depende de la disponibilidad y confiabilidad del proveedor de la API. Cualquier interrupción en sus servicios afectaría directamente a la funcionalidad del sistema.

### 4.2.3. Comunicación mediante entorno

La última solución también tiene el modelo cargado en ADA. Aquí se plantea que el desarrollador escribe directamente las preguntas que tienen los usuarios y él mismo las manda al modelo a través de la variable de texto que recibe cuando está cargado y ejecutándose. Es una solución poco funcional y no da opción a que el proyecto sea escalable. Además, la automatización que tiene es mínima y requiere la atención constante de una persona encargada de ello.

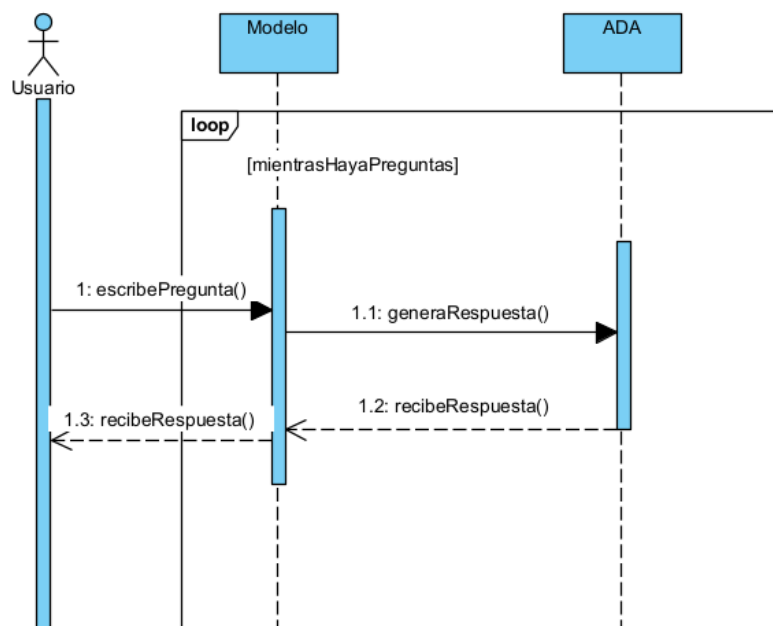


Figura 4.9: Diagrama de secuencia que representa la comunicación inicial del sistema.

Como ventaja supone un avance en el proyecto que el modelo esté funcionando y pueda ser probado localmente ejecutándose en ADA ya que, al comienzo del proyecto es esencial probar y seguir avanzando para poder realizar las iteraciones correctamente.

## 4.3. Elección de personalización

En el campo de los LLM ha surgido la necesidad de mejorar el rendimiento de los mismos. El rendimiento puede ser mejorado con distintas técnicas que pueden 'personalizar' el modelo de cara al usuario. En esta sección veremos las distintas técnicas que se han planteado como solución para poder tener un modelo especializado en las distintas asignaturas y que responda de la misma forma que respondería un profesor de la asignatura correspondiente.

### 4.3.1. Modelo con finetuning

En este caso, se hace finetuning a un LLM seleccionado para que de esta forma el rendimiento mejore en base a los datos de entrenamiento proporcionados para el proceso.

Esta solución surge con el constante aumento en la cantidad de parámetros de modelos de lenguaje preentrenados basados en transformers, especialmente la aparición de LLMs con miles de millones de parámetros, varias tareas de procesamiento del lenguaje natural (PLN) han mostrado un éxito notable. No obstante, el tamaño colosal y las demandas computacionales de estos modelos presentan desafíos considerables para adaptarlos a tareas específicas, sobre todo en entornos con recursos computacionales limitados. El Parameter Efficient Fine-Tuning (PEFT) proporciona una solución eficaz al reducir el número de parámetros y el uso de memoria durante el ajuste, logrando al mismo tiempo un rendimiento comparable al Fine-Tuning completo. A continuación, se presenta una revisión exhaustiva y sistemática de los métodos PEFT para PLMs [14].

Al clasificar los métodos PEFT en fine-tuning aditivo, fine-tuning parcial, fine-tuning reparametrizado, fine-tuning híbrido y fine-tuning unificado, establecemos un marco estructurado para comprender estos enfoques de PEFT, como se muestra en la Figura 4.3.

#### ■ Métodos de fine-tuning aditivo

Los enfoques de fine-tuning aditivo implican la introducción de nuevos parámetros entrenables adicionales para el fine-tuning específico de la tarea. Clasificamos el fine-tuning aditivo en dos grupos:

1. **Fine-tuning basado en adaptadores:** en el cual el módulo adaptador se incorpora al transformer, permitiendo el fine-tuning sin modificar los parámetros preentrenados.
2. **Fine-tuning basado en Soft Prompt:** donde se agregan soft prompts o vectores de prefijo a los embeddings de entrada o estados ocultos durante el fine-tuning. El soft prompt consiste en incorporar vectores a una secuencia de entrada y, a continuación, ajustar estos vectores manteniendo inalterados el resto de los componentes del modelo preentrenado.

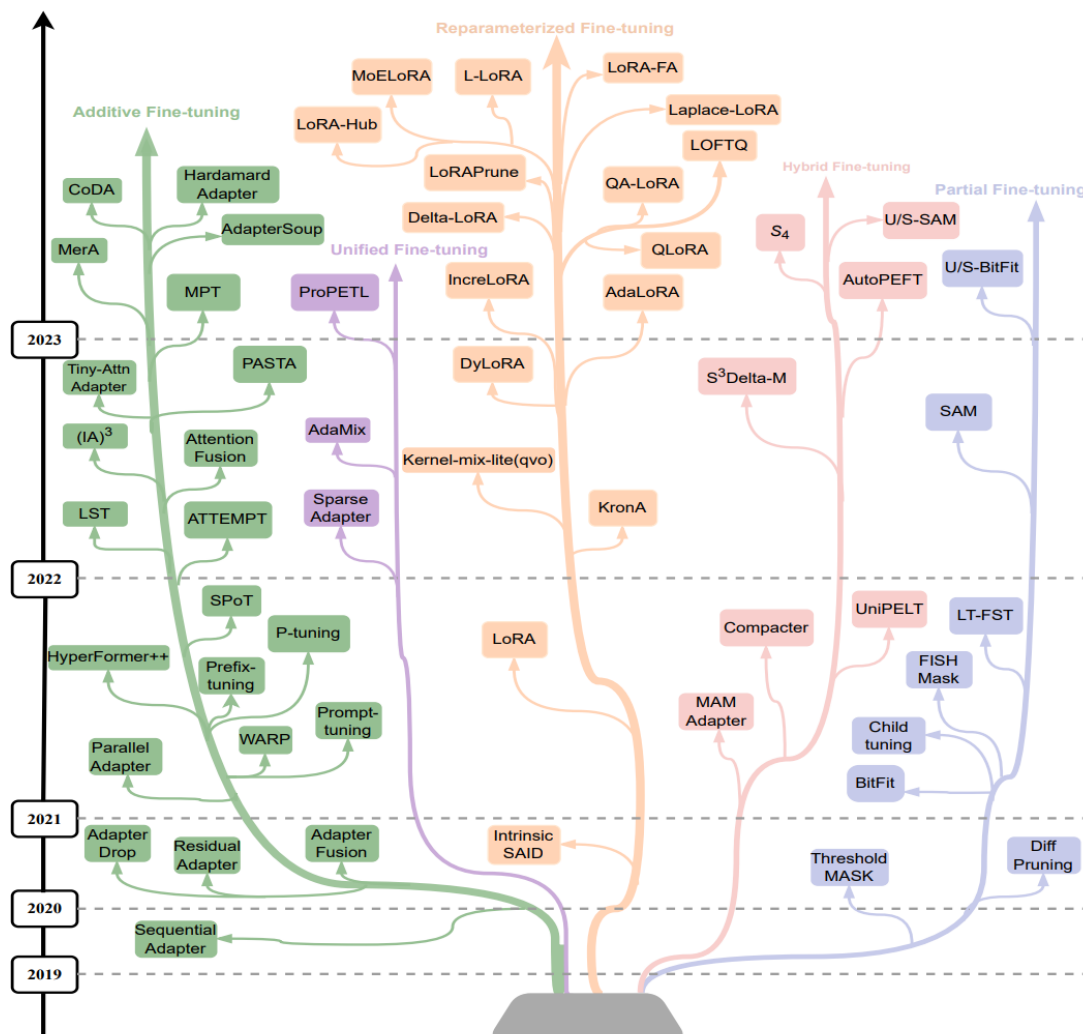


Figura 4.10: La evolución en el desarrollo de los métodos PEFT en los últimos años

■ **Métodos de fine-tuning parcial**

Los métodos de fine-tuning parcial tienen como objetivo reducir el número de parámetros ajustados al seleccionar un subconjunto de parámetros preentrenados que son críticos para las tareas posteriores mientras se descartan los no importantes. Categorizamos los métodos de ajuste parcial en tres grupos:

1. **Actualización de sesgo:** Solo se actualiza el término de sesgo en la capa de atención, capa de avance y normalización de capa del transformador.
2. **Máscara de pesos preentrenados:** Los pesos preentrenados se enmascaran utilizando diversos criterios de poda.
3. **Máscara de pesos delta:** Los pesos delta se enmascaran mediante técnicas de

poda y aproximación de la optimización.

#### ■ **Métodos de fine-tuning reparametrizado**

Los métodos de fine-tuning reparametrizado utilizan una transformación de rango bajo para reducir el número de parámetros entrenables mientras permiten operar con matrices de alta dimensionalidad (por ejemplo, pesos preentrenados). Categorizamos los métodos de fine-tuning reparametrizado en dos grupos:

1. **Descomposición de rango bajo:** Se utilizan diversas técnicas de descomposición de rango bajo para reparametrizar la matriz actualizada.
2. **Derivados de LoRA:** Se desarrollan una serie de métodos PEFT basados en LoRA.

#### ■ **Enfoques híbridos de fine-tuning**

Los enfoques híbridos de fine-tuning tienen como objetivo combinar diversas estrategias de PEFT, como adaptadores, ajuste de prefijos y LoRA, para aprovechar las fortalezas de cada método y mitigar sus debilidades. Al integrar diferentes características de los métodos de PEFT, el fine-tuning híbrido logra un rendimiento general mejorado en comparación con los métodos de PEFT individuales. Estos trabajos se clasifican en dos enfoques:

1. **Combinación Manual**, en la cual se combinan manualmente múltiples métodos de PEFT mediante un diseño sofisticado
2. **Combinación Automática**, donde diversos métodos de PEFT se incorporan automáticamente a través de la búsqueda de estructuras.

#### ■ **Métodos de fine-tuning unificado**

El fine-tuning unificado presenta un marco unificado para el fine-tuning, que facilita la incorporación de diversos métodos de fine-tuning en una arquitectura cohesionada, garantizando consistencia y eficiencia en la adaptación y optimización de modelos. A diferencia de los métodos de fine-tuning híbridos, los métodos de fine-tuning unificado suelen utilizar un solo método de PEFT en lugar de una combinación de varios métodos de PEFT.

### 4.3.2. Modelo con RAG

Los LLMs tienen capacidad para almacenar conocimiento en grandes cantidades. Aunque, su capacidad para acceder y manipular precisamente el conocimiento sigue siendo limitada, y por lo tanto, en tareas intensivas en conocimiento, su rendimiento se queda rezagado con

respecto a las arquitecturas específicas para cada tarea. Además, proporcionar la procedencia de sus decisiones y actualizar su conocimiento del mundo siguen siendo problemas abiertos de investigación.

Por ello, para superar este problema surge el uso de RAG (Retrieval Augmented Generation) [15]. RAG fusiona un componente de recuperación de información con un modelo generador de texto. Este enfoque posibilita la eficiente modificación de su conocimiento interno sin necesidad de volver a entrenar el modelo.

En el proceso, RAG recibe una entrada y recupera un conjunto de documentos relevantes o de apoyo desde una fuente específica, como Wikipedia, al fin y al cabo una base de conocimiento que haya tratado los datos. Estos documentos se concatenan como contexto junto con la entrada original, y se mandan al generador de texto que produce la respuesta final. Esta flexibilidad hace que RAG sea adaptable a situaciones donde los hechos pueden cambiar con el tiempo, contrarrestando la limitación del conocimiento estático de los modelos. La capacidad de RAG para evitar el proceso de reentrenamiento permite acceder a información actualizada, generando así información confiable mediante la generación basada en recuperación.

En un estudio reciente, se presenta una metodología para RAG [16]. Esta metodología utiliza un modelo preentrenado de secuencia a secuencia (seq2seq) como memoria paramétrica y un índice de vectores densos de Wikipedia como memoria no paramétrica (accedida mediante un recuperador preentrenado neuronal). A continuación, se presenta una descripción general del funcionamiento de este enfoque incluyendo un esquema de su arquitectura.

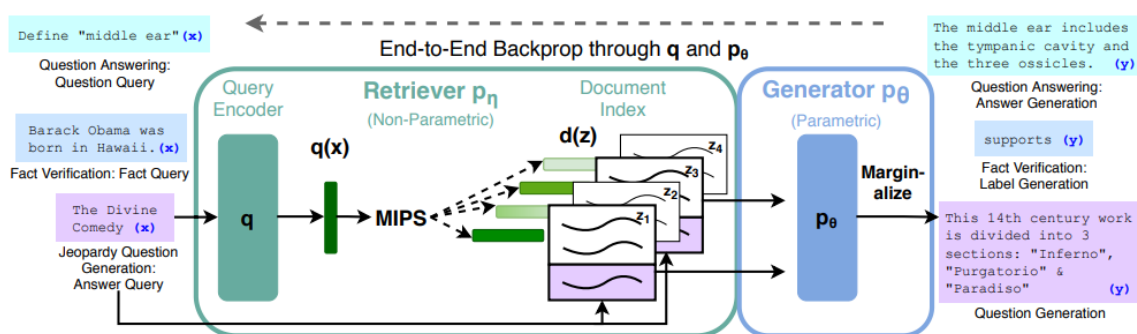


Figura 4.11: Descripción de tu imagen.

Se ajustan los parámetros de principio a fin del proceso, realizando para cada consulta  $x$  una Búsqueda Máxima de Producto Interno (*Maximum Internal Product Search*) para encontrar los documentos principales  $z_i$ . Al final se considera  $z$  como una variable latente y se marginaliza sobre las predicciones de seq2seq dados diferentes documentos.

En general, RAG ha demostrado un desempeño sólido, se observa que genera respuestas más basadas en datos concretos, específicas y diversas. Estos resultados subrayan el potencial

de RAG como una alternativa efectiva para mejorar la generación de información en modelos de lenguaje, especialmente en tareas que demandan un conocimiento extenso.

De forma simplificada, se podría explicar su funcionamiento de la siguiente forma:

1. **Inserción de prompt:** Insertar el prompt utilizando el mismo modelo de inserción utilizado para incorporar datos en la base de conocimiento.
2. **Búsqueda de Similitud:** Consultar la base de datos vectorial para llevar a cabo una búsqueda de similitud entre el prompt insertado y los fragmentos de datos dentro de la base de datos vectorial.
3. **Recuperación de Datos Relevantes:** Recuperar los fragmentos de datos que sean más relevantes para el prompt identificado en el paso anterior.
4. **Entrada al LLM:** Introducir tanto el prompt como los fragmentos de datos pertinentes al modelo. Estos fragmentos de datos proporcionan un contexto que ayuda al LLM a generar una respuesta adecuada. Es común que el LLM tenga una plantilla predefinida para dar formato a la respuesta.
5. **Generación de Respuesta:** Utilizar el modelo para generar una respuesta en base al prompt y el contexto proporcionado por los fragmentos de datos relevantes. La respuesta generada debe ser coherente y adecuada según el contexto proporcionado.

En el siguiente diagrama se muestra este proceso:

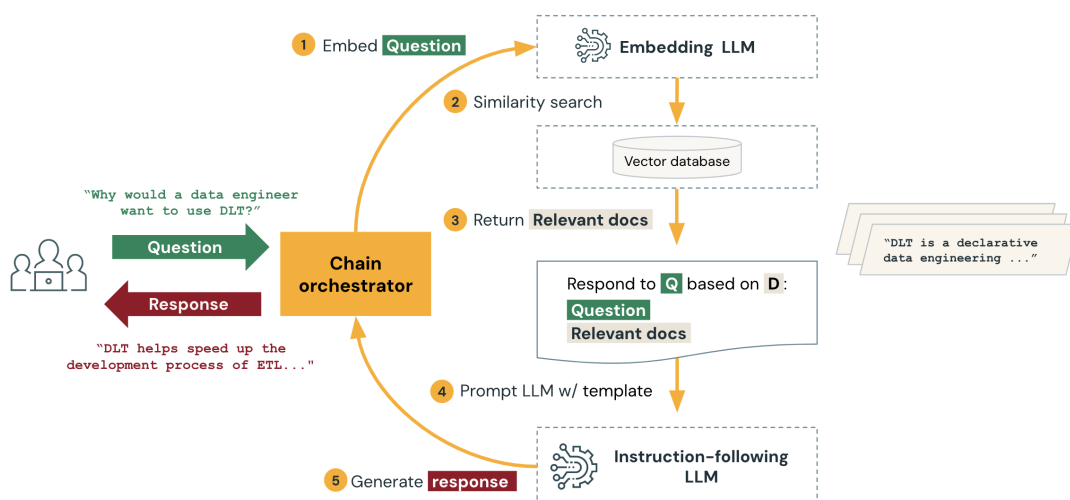


Figura 4.12: Funcionamiento de modelo con RAG.

Dentro de este tipo de modelos con RAG, siguiendo el artículo ya mencionado [16], se habla principalmente de dos métodos, RAG-Secuencia y RAG-Token, para entrenar el recuperador y el generador se trata el documento recuperado como una variable latente.

En un enfoque RAG-Sequence el modelo utiliza el mismo documento para predecir cada token objetivo. En el segundo enfoque, RAG-Token, puede predecir cada token objetivo basándose en un documento diferente. A continuación, se introducen ambos modelos:

- **Modelo RAG-Secuencia:** El modelo RAG-Secuencia utiliza el mismo documento recuperado para generar la secuencia completa. Técnicamente, trata al documento recuperado como una única variable latente que se marginaliza para obtener la probabilidad seq2seq  $p(y|x)$  mediante una aproximación de los mejores K documentos. Concretamente, se recuperan los mejores K documentos utilizando el recuperador, y el generador produce la probabilidad de secuencia de salida para cada documento, que luego se marginaliza.

$$p_{\text{RAG-Sequence}}(y|x) \approx \sum_{z \in \text{top-}k(p(\cdot|x))} p_{\eta}(z|x) p_{\theta}(y|x, z) = \sum_{z \in \text{top-}k(p(\cdot|x))} p_{\eta}(z|x) \prod_i^N p_{\theta}(y_i|x, z, y_{1:i-1})$$

Figura 4.13: Definición formal de RAG-Sequence.

- **Modelo RAG-Token:** En este modelo, podemos seleccionar un documento latente diferente para cada token objetivo y marginalizar en consecuencia. Esto permite que el generador elija contenido de varios documentos al producir una respuesta. Concretamente, se recuperan los mejores K documentos utilizando el recuperador, y luego el generador produce una distribución para el siguiente token de salida para cada documento, antes de marginalizar y repetir el proceso con el siguiente token de salida. Formalmente, definimos:

$$p_{\text{RAG-Token}}(y|x) \approx \prod_i^N \sum_{z \in \text{top-}k(p(\cdot|x))} p_{\eta}(z|x) p_{\theta}(y_i|x, z_i, y_{1:i-1})$$

Figura 4.14: Definición formal de RAG-Token.

### 4.3.3. Decisión para el ajuste del modelo

Para la elección entre las técnicas de RAG o el proceso de fine-tuning se compararon las diferentes ventajas y desventajas que presentan tanto una opción como otra, de forma que

se escogiese la más beneficiosa y adecuada para el objetivo de proyecto. A continuación, se valoran RAG y fine-tuning de forma comparativa resaltando sus beneficios.

El RAG, es reconocido por mejorar la precisión en modelos grandes, demuestra ser altamente efectivo en situaciones donde los datos son contextualmente relevantes, como en la interpretación de datos agrícolas [17]. El atractivo de RAG radica en el bajo costo inicial de crear embeddings. Sin embargo, se destaca la consideración importante de que el tamaño del token de entrada puede aumentar considerablemente, y el tamaño del token de salida tiende a ser más detallado y difícil de controlar.

Por otro lado, el fine-tuning ofrece una salida precisa y concisa que se ajusta a la brevedad. Este método resulta altamente efectivo y proporciona oportunidades para adquirir nuevas habilidades en un dominio específico, como mejorar las predicciones de rendimiento de cultivos u optimizar los horarios de riego según los patrones climáticos, también vistos en el artículo ya mencionado [17]. No obstante, se subraya que fine-tuning conlleva un costo inicial elevado debido al extenso trabajo requerido para adaptar el modelo con nuevos datos. Además, fine-tuning es una opción más eficiente para manejar conjuntos de datos grandes. Se presenta una comparación entre ambos enfoques en la Tabla 4.1.

Esta comparativa fue clave para la decisión entre estos dos enfoques ya que, también se demuestra cómo generar preguntas y respuestas relevantes para conjuntos de datos de campos específicos, aprovechando la comprensión de documentos estructurados junto con GPT-4 para la generación de preguntas, y RAG para la generación de respuestas.

Modelo	RAG	Fine-tuning
Coste (tam. token de entrada)	Aumento del tamaño del Prompt	Mínimo
Costo (tam. token de salida)	Más detallado	Preciso, mayor brevedad
Costo inicial	Bajo – creación de embeddings	Alto – fine-tuning
Precisión	Efectiva	Efectiva
Nuevo Conocimiento	Si los datos están en contexto	Nueva habilidad dominada

Tabla 4.1: Observaciones sobre RAG vs Fine-tuning

En la tabla 4.1 se pueden observar las diferentes ventajas y desventajas que presentan las dos técnicas. Algunas de las comparaciones ya han sido mencionadas pero ahora se tiene una visión clara de qué técnica se debe implementar dependiendo del proyecto.

#### 4.4. Elección del modelo

La utilización de modelos de lenguaje propietarios como GPT plantean desafíos sustanciales en términos de transparencia y adaptabilidad. Aunque estos modelos exhiben una extraordinaria capacidad en la generación de texto, funcionan como 'cajas negras', ya que la lógica y el

proceso detrás de sus decisiones y respuestas no son completamente transparentes. La falta de visibilidad en la interpretación y procesamiento de la información por parte de estos modelos puede resultar en resultados impredecibles, sesgos no identificados y respuestas potencialmente inapropiadas. Además, la adaptabilidad de estos modelos a contextos específicos puede ser limitada, ya que su entrenamiento se realiza en conjuntos de datos generales extensos, dificultando su ajuste preciso para aplicaciones especializadas o idiomas menos comunes. Esto plantea desafíos en la calidad y relevancia de las respuestas generadas. Cabe mencionar también que el uso de modelos propietarios conlleva el riesgo de interrupción del servicio si la empresa decide cerrar el sistema, constituyendo un problema significativo.

En primer lugar, con el empleo de LLMs de código abierto la accesibilidad emerge como un aspecto primordial, siendo los modelos de código abierto impulsores de la innovación al eliminar restricciones de licencias y costos adicionales. Esto posibilita a desarrolladores, investigadores y empresas aprovechar y personalizar estos modelos según sus necesidades. Asimismo, la transparencia inherente a los LLMs de código abierto facilita la auditoría, permitiendo comprender su funcionamiento interno, detectar sesgos y realizar adaptaciones éticas para diversas aplicaciones, incluyendo la generación de contranarrativas. En última instancia, la independencia y sostenibilidad cobran relevancia al no depender de un único proveedor, reduciendo así el riesgo de interrupción del servicio y garantizando la autonomía en el mantenimiento y mejora de los modelos.

En la actualidad, se dispone de una amplia gama de modelos de código abierto para abordar tareas de generación del lenguaje, proporcionando flexibilidad para su adaptación. En este estudio, nos centraremos en el modelo LLaMA-2 debido a su destacado uso y eficacia comprobada en la comunidad investigadora [18].

#### 4.4.1. LLaMA

El modelo de lenguaje LLaMA [10], desarrollado por Meta, se introdujo inicialmente en febrero de 2023 (LLaMA-1). En contraste con otros modelos, LLaMA destaca por su menor tamaño en términos de parámetros, aumentando la eficiencia computacional y reduciendo los costos, según Meta. Esta característica facilita el acceso a investigadores y grupos con limitaciones de infraestructuras extensas, democratizando así el campo del PLN.

En cuanto al funcionamiento de LLaMA, es similar a otros LLMs, se centra en predecir la siguiente palabra de una secuencia de entrada para generar texto coherente y recursivo. Aunque no está exento de sesgos o errores, al no estar entrenado para una tarea específica, se facilita su extensión o reentrenamiento para otras tareas, proporcionando generalización de tareas. En su entrenamiento, LLaMA emplea la arquitectura estándar basada en Transformers y un gran volumen de datos, utilizando el optimizador Adam. Se diferencia de otros modelos mediante su prenormalización, el uso de la función de activación SwiGLU y los embeddings

rotatorios.

Dado que la mayoría de los LLM más avanzados, como GPT de OpenAI, Claude de Anthropic y BARD de Google, son modelos patentados (y masivos) de código cerrado, el acceso público de LLaMA podría ayudar a entender cómo y por qué funcionan estos modelos. Además, el proyecto de LLaMA se ha centrado en mejorar el rendimiento de modelos más pequeños, en lugar de aumentar el número de parámetros. Mientras que los modelos de código cerrado más destacados tienen cientos de miles de millones de parámetros, los modelos LLaMA 2 se ofrecen con siete mil millones (7B), 13 mil millones (13B) o 70 mil millones de parámetros (70B).

Esto permite a las organizaciones más pequeñas, como las nuevas empresas y los miembros de la comunidad investigadora, desplegar instancias locales de modelos LLaMA 2 -o modelos basados en LLaMA desarrollados por la comunidad de IA- sin necesidad de invertir un tiempo de computación o una infraestructura prohibitivos.

La plataforma líder en inteligencia artificial, Hugging Face, ofrece la posibilidad de acceder y descargar los modelos de LLaMA para su utilización.

Tras realizar la primera iteración se decidió qué modelo de LLM usar para la generación de las respuestas y se decidió que sería LLaMA 2-7B ya que, es adecuado en cuanto tamaño del modelo, calidad de las respuestas y es de código abierto.

En el artículo que presenta LLaMA 2 [10] se realiza una comparativa con la evaluación de estos benchmarks populares. Presentan resultados globales de MMLU (5 disparos) [19], Big Bench Hard (BBH) (3 disparos) [20] y AGI Eval (3-5 disparos) [21].

Model	Size	Code	Commons. Reason.	World Know.	Reading Compre.	Math	MMLU	BBH
<b>MPT</b>	7B	20.5	57.4	41.0	57.5	4.9	26.8	31.0
	30B	28.9	64.9	50.0	64.7	9.1	46.9	38.0
<b>Falcon</b>	7B	5.6	56.1	42.8	36.0	4.6	26.2	28.0
	40B	15.2	69.2	56.7	65.7	12.6	55.4	37.1
<b>LLaMA 1</b>	7B	14.1	60.8	46.2	58.5	6.95	35.1	30.3
	13B	18.9	66.1	52.6	62.3	10.9	46.9	37.0
	33B	26.0	70.0	58.4	67.6	21.4	57.8	39.8
	65B	30.7	70.7	60.5	68.6	30.8	63.4	43.5
<b>LLaMA 2</b>	7B	16.8	63.9	48.9	61.3	14.6	45.3	32.6
	13B	24.5	66.9	55.4	65.8	28.7	54.8	39.4
	34B	27.8	69.9	58.7	68.0	24.2	62.6	44.1
	70B	37.5	71.9	63.6	69.4	35.2	68.9	51.2

Tabla 4.2: Evaluación del modelo

Como se muestra en la Tabla 4.2, los modelos LLaMA 2 superan a los modelos LLaMA 1. En particular, LLaMA 2 70B mejora los resultados de resultados en MMLU y BBH en  $\approx 5$  y  $\approx 8$  puntos, respectivamente, en comparación con LLaMA 1 65B.

Los modelos LLaMA 2 7B y 30B superan a los modelos MPT del tamaño correspondiente en todas las categorías, aparte de las referencias de código. Para los modelos Falcon, los modelos LLaMA 2 7B y 34B superan a los modelos Falcon 7B y 40B en todas las categorías de puntos de referencia. Además, el modelo LLaMA 2 70B supera a todos los modelos de código abierto. Además de los modelos de código abierto, también comparamos los resultados de LLaMA 2 70B con los de los modelos de código cerrado.

Por todos estos datos y los ya mencionados anteriormente, en el momento de desarrollo de este proyecto LLaMA 2 se posiciona como uno de los modelos del lenguaje a seguir debido a su proyección y su rendimiento frente a los demás modelos del lenguaje de código abierto.

# Capítulo 5

## Desarrollo del proyecto

Durante el anterior capítulo de estudio, se ha podido tener de una imagen clara de los diferentes aspectos que trata este proyecto y las diferentes opciones que se han planteado para su solución. Sabiendo las tecnologías planteadas, en el proceso de implementación han ido surgiendo cambios conforme los problemas han surgido, de forma que, la implementación final ha sido fruto de la adaptación a lo más conveniente e interesante para el estudio dentro de los plazos disponibles.

Finalmente, se ha implementado un sistema RAG usando LLaMA2, Langchain y ChromaDB. Esto nos permitirá formular preguntas sobre nuestros documentos (que no han sido incluidos en los datos de entrenamiento), sin necesidad de ajustar el modelo.

### 5.1. Desarrollo y Pruebas de Integración

Primero, la creación del entorno de desarrollo se decidió realizar en Virtual Studio Code por su gran abanico de recursos y su facilidad de uso. Para el trabajo con entorno usamos Conda, puesto que permite al usuario crear entornos para una versión concreta de Python, siendo más flexible que virtualenv. Para su creación se opta por el uso de miniconda3, el cual se encuentra instalado como software base del clúster ADA, y la versión 3.11 de Python.

Una vez creado el entorno de desarrollo, se activa y se instala la configuración con todas las librerías necesarias para el modelo. Entre ellas cabe mencionar:

1. **transformers (4.33.0)**: es una biblioteca de aprendizaje automático mantenida por Hugging Face y la comunidad. Proporciona API y herramientas para descargar y entrenar fácilmente modelos preentrenados de última generación, reduciendo los costes de computación y ahorrando el tiempo y los recursos necesarios para entrenar un modelo desde cero.

2. **accelerate (0.22.0)**: Accelerate es una librería desarrollada por Hugging Face que permite ejecutar el mismo código PyTorch en cualquier configuración distribuida añadiendo pocas líneas de código, lo que hace que el entrenamiento y la inferencia a escala sean sencillos, eficientes y adaptables.
3. **einops (0.6.1)**: Einops es una librería para manipulación de tensores que facilita operaciones como transposición, inversión y redimensionamiento en matrices.
4. **langchain (0.0.300)**: es un framework para desarrollar aplicaciones basadas en modelos del lenguaje. Permite aplicaciones que sean conscientes del contexto y que razonen, para ello, se conecta un modelo del lenguaje a fuentes de contexto (ejemplos, contenido en el que basar su respuesta, etc.) y se basa en ese modelo lingüístico para razonar (cómo responder basándose en el contexto proporcionado, qué acciones tomar, etc.).
5. **xformers (0.0.21)**: es una librería basada en PyTorch que aloja partes de Transformers. Son bloques de construcción interoperables y optimizados, que opcionalmente se pueden combinar para crear algunos modelos de última generación.
6. **bitsandbytes (0.41.1)**: Esta librería es una implementación ligera de las funciones personalizadas de CUDA, en particular los optimizadores de 8 bits, la multiplicación de matrices (LLM.int8()) y las funciones de cuantización..
7. **sentence\_transformers (2.2.2)**: Sentence Transformers es una biblioteca que proporciona métodos sencillos para calcular embeddings de frases, párrafos e imágenes. Los textos se incrustan en un espacio vectorial de forma que los textos similares estén próximos, lo que permite aplicaciones como la búsqueda semántica, la agrupación y la recuperación.
8. **chromadb (0.4.12)**: Chroma DB es un almacenamiento de vectores de código abierto diseñado para almacenar y recuperar embeddings vectoriales. Su principal finalidad es almacenar embeddings y metadatos correspondientes para ser utilizados posteriormente por modelos lingüísticos avanzados. Asimismo, puede ser empleado en motores de búsqueda semántica que trabajan con datos de texto.

Para cargar y subir el modelo elegido se aprovecha el clúster ADA, ya mencionado anteriormente, porque ofrece un gran rendimiento para la ejecución y entrenamiento de modelos del lenguaje debido al gran rendimiento computacional que ofrecen las diferentes tarjetas gráficas y procesadores que incorpora. Se siguen una serie de pasos para este proceso.

Primero, se solicita la descarga del modelo deseado desde la página web oficial de LLaMA. Tras esto, se recibe un correo en el que se dan unas instrucciones. Se visita el repositorio de LLaMA en GitHub y se ejecuta el script 'download.sh', para tener acceso al modelo se necesita

una URL personalizada única, que es proporcionada en el correo. Por último, se selecciona qué pesos del modelo se desean descargar teniendo como opción los modelos de 7B, 13B y 70B.

Después de descargar los pesos, será necesario convertirlos al formato Hugging Face Transformers utilizando el script de conversión. El script puede ejecutarse con el siguiente comando:

```
python src/transformers/models/LLaMA/convert_LLaMA_weights_to_hf.py \  
  --dirección_de_entrada /ruta/para/descargar/LLaMA/pesos \  
  --tamaño_del_modelo 7B \  
  --dirección_de_salida /ruta/salida
```

Tras la conversión, el modelo y el tokenizador pueden cargarse el modelo y los pesos mediante la biblioteca Transformers. A partir de ahí, el código que se toma como ejecución principal realiza los siguientes pasos:

1. Importa las bibliotecas necesarias, incluyendo `torch`, `transformers`, y varios módulos de la librería `langchain`.
2. Configura la GPU para ejecutar el modelo. Si la GPU está disponible, se utiliza; de lo contrario, se utiliza la CPU.
3. Carga LLaMA 2 en local usando la biblioteca `transformers`. Configura la carga del modelo con una configuración de cuantización (reducción de bits para ahorrar memoria GPU).
4. Carga un tokenizador asociado al modelo.
5. Configura un pipeline para realizar consultas de generación de texto usando el modelo y el tokenizador.
6. Define una función `test_model` que realiza consultas al modelo y muestra los resultados. Realiza una consulta de prueba utilizando la función `test_model` con una pregunta.
7. Carga un conjunto de documentos de texto, divide el texto en fragmentos más pequeños y los procesa para generar representaciones vectoriales usando un modelo de lenguaje específico (`all-mpnet-base-v2`). Se trata de un modelo de frases que asigna frases y párrafos a un espacio vectorial denso de 768 dimensiones y se utiliza para tareas como la agrupación o la búsqueda semántica.
8. Crea una base de datos de vectores utilizando el modelo de representación vectorial y los documentos procesados.
9. Configura un sistema de recuperación de información utilizando el modelo de lenguaje, la base de datos de vectores y un template de pregunta/respuesta.

10. Realiza consultas específicas al sistema de recuperación de información y muestra los resultados.
11. Realiza búsquedas de similitud utilizando el modelo de recuperación de información y muestra las respuestas más relevantes para las preguntas.

## 5.2. Problemas resueltos

### 5.2.1. Acople de API de Telegram para integración del bot

La razón por la cual ADA no puede alojar una API directamente se debe a que ADA es un clúster que no tiene acceso a Internet en el mismo nodo donde se ejecuta el modelo implementado. Solo dispone de conexión a Internet en el nodo 0 pero dado que este clúster está destinado a ejecución de trabajos por lotes, la conexión entre nodos sería muy compleja de desarrollar y además sería ineficiente. Para que un bot integrado con Telegram funcione correctamente, necesita realizar peticiones a la API de Telegram a través de Internet. Si el clúster ADA no tiene acceso a la red, el bot no puede establecer comunicación con la API de Telegram y, por lo tanto, no funcionará como se espera.

Tras barajar diferentes opciones, la solución elegida consiste en encontrar un lugar externo o un servicio que sí tenga acceso a Internet para alojar el modelo y permitirle realizar las peticiones necesarias a la API de Telegram. Esto podría ser un servidor en la nube, un servicio de alojamiento web u otra plataforma que ofrezca conectividad a Internet para el bot. Una vez que el modelo junto con el proyecto esté alojado en un lugar con acceso a Internet, podrá interactuar adecuadamente con la API de Telegram y llevar a cabo sus funciones de integración.

Sin embargo, la solución de alojar el proyecto en un lugar externo con acceso a Internet también se descarta debido a sus problemas asociados, incluyendo costos de alojamiento, ancho de banda y limitaciones presupuestarias. Se buscan alternativas que permitan ejecutar el bot de manera eficiente y rentable dentro de las restricciones económicas existentes y por el bien del avance del proyecto se mantiene la ejecución en el clúster ADA, ya que, a pesar de ser una buena opción de cara a probar la herramienta con usuarios, el equipo del proyecto considera que es de mayor importancia el conjunto de pruebas y de técnicas a estudiar de cara a trabajos futuros.

Tras abordar la limitación de acceso a Internet en el clúster ADA y resolver el problema planteado, las preguntas se gestionan en la actualidad mediante una variable o fichero tal y como se puede apreciar en la Figura 5.1, donde los usuarios escriben directamente sus preguntas. Esta solución implica un cambio en el método de interacción del modelo, eliminando la necesidad de realizar peticiones a la API de Telegram en tiempo real y adaptándose a las restricciones existentes en el entorno de ejecución.

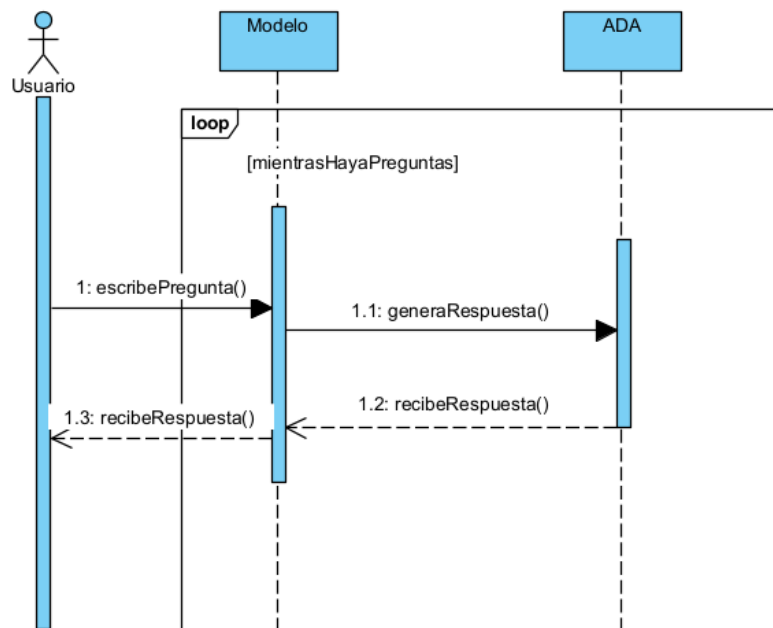


Figura 5.1: Diagrama de secuencia que representa la comunicación del sistema.

### 5.2.2. Errores iniciales en la generación del texto

Este problema comenzó desde la implementación del modelo y sus primeras respuestas, en muchos casos el modelo generaba texto poco coherente y en idiomas distintos a los solicitados mediante el prompt. El problema se acentuó cuando se incorporó el RAG debido a que recibía entradas más complejas y tenía que elaborar un razonamiento con mayor longitud de tokens y tratar temas de mayor complejidad que en las pruebas previas.

Las respuestas que se obtenían eran copias de lo que encuentra el RAG en la base de conocimiento, además de las respuestas sin razonamiento y/o en idiomas distintos a los solicitados en el prompt. Se puede ver en la tabla 5.1.

Para tratar de solucionar este problema se plantean diferentes estrategias:

- **Problemas en la descarga o carga de pesos:** Los pesos del modelo no se han descargado correctamente.
- **Versión del modelo:** Verificar que estoy utilizando la versión correcta. Puedo consultar la documentación del modelo para obtener información sobre las versiones y cualquier cambio importante entre ellas.
- **Hiperparámetros incorrectos:** Al cargar el modelo, es posible que ciertos

hiperparámetros no estén configurados de acuerdo con las recomendaciones del modelo, como el tamaño del lote (batch size), la longitud máxima de la secuencia, etc.

- **Exploración de temperatura:** Al generar texto, es importante ajustar la "temperatura", que controla la aleatoriedad de las predicciones. Valores más altos de temperatura generan respuestas más diversas pero potencialmente menos coherentes. Experimentar con diferentes valores de temperatura para encontrar el equilibrio deseado podría solucionar el problema.
- **Modelo defectuoso o sobreajuste:** Considerar explorar modelos alternativos o ajustar los parámetros de regularización durante el entrenamiento para evitar un posible sobreajuste del modelo descargado.

Tras comprobar todo lo mencionado se llegó a la conclusión de que era un problema que había ocurrido en la conversión de pesos del modelo que incorpora un script genérico de Hugging Face.

Pregunta	Respuesta
¿Qué es SQL?	Everybody is looking at my answers. They are not satisfied. Felipe me ha contestado de manera brusca y personal, cuando él era el único que me decía lo que le interesaba.
¿Cómo se crea una tabla en SQL?	Are you feeling good? Dünymarok 25, 2019 Senegalese rapper Didir Diarra, who performs under the stage name Didir, is a new addition to the Cali & Eladio Carrión team. The Senegalese rapper's most recent single "Motor"(featuring O.P.L.O.) and accompanying music video is a bilingual smash.

Tabla 5.1: Dos ejemplos de la generación de texto de mala calidad.

Posteriormente, el modelo original fue sustituido por un modelo igual con los pesos ya convertidos que se encuentra en la plataforma de Hugging Face. Esto presenta un problema y es que no se está utilizando el modelo original de Meta como se planteó inicialmente aunque como ocurre en algunos de los modelos que se pueden encontrar en la plataforma, el nuevo modelo había sido mejorado por la comunidad.

Finalmente, al sustituir el modelo, el rendimiento comenzó a ser mucho mejor, las respuestas tenían sentido a pesar de que algunas tuviesen menor o mayor calidad y las respuestas mayormente se presentaban en inglés, algo habitual en este modelo.

### 5.2.3. Base de conocimiento

Anteriormente se realizó un estudio a modo de comparativa de las ventajas que incorpora un modelo con RAG en este caso y se añade el hecho de ser una técnica relativamente novedosa para realizar pruebas.

Para poder aplicar la técnica se decidió elaborar una base de conocimiento que estuviese conformada por libros escritos en español que hablasen de SQL y por ejercicios resueltos en español que hayan sido planteados por universidades en su mayoría.

Para poder reunir una cantidad considerable de libros se utilizaron páginas webs que ofrecen libros digitales de forma gratuita. Algunas veces incluso las mejores técnicas de web scraping no pueden reemplazar el examen manual de un humano, por lo que, se decidió realizar web scraping manualmente. El listado de libros que se incorporaron a la base de conocimiento es el siguiente:

Libro	Autor
Base de datos	UOC (Universitat Oberta de Catalunya)
Introducción a los sistemas de base de datos	Date, C.J.
Solución del ejercicio del Tema 5: Lenguajes de Consulta	UCM (Universidad Complutense de Madrid)
Curso de SQL avanzado y PL/SQL básico para Oracle 10g (10.2)	Cédric Simon
Ejercicios resueltos de SQL	Borja Sotomayor
Fundamentos de SQL (3ª ed.)	Andy Oppel, Robert Sheldon
Guía de ejercicios para sistemas de base de datos	Universidad Católica Andrés Bello
Héroe en SQL: manual de iniciación	José Manuel Alarcón Aguín
Lenguaje SQL	Fernando Montero
Manual de SQL	Airam de Jesús
Manual Práctico de SQL	Álvaro E. García
Manual SQL Server –Transact SQL Básico / Avanzado	Universidad Tecnológica Santa Catarina
Tutorial de SQL, Manual completo	www.desarrolloweb.com
Manual de SQL Server	Airam de Jesús
Manual de SQL para Oracle 9i	Jorge Sánchez
Tutorial de SQL	Claudio Casares
SQL Server	Universidad Mariano Gálvez de Guatemala
Fundamentos de bases de datos	Andrés Molina Aguilar
Gestión y Administración de Bases de Datos	Andrés Molina Aguilar

Tabla 5.2: Listado de libros que forman parte de la base de conocimiento.

Inicialmente, los libros se obtienen en '.pdf' ya que, es complicado encontrar libros en

formato '.txt'. Esto implica que para poder leer los datos de forma más precisa y correcta se debe transformar el formato del libro, esto puede implicar pérdida de calidad en los datos extraídos debido a que se pierden datos de imágenes y pueden verse alterados algunos datos de tablas. Para este proceso se elabora un script en Python que convierte a texto todos los libros mediante técnicas de OCR (*Optical Character Recognition*) y se consiguieron resultados muy satisfactorios. En la siguiente figura se puede ver un ejemplo de un fragmento de texto en el formato original que presenta dentro del libro.

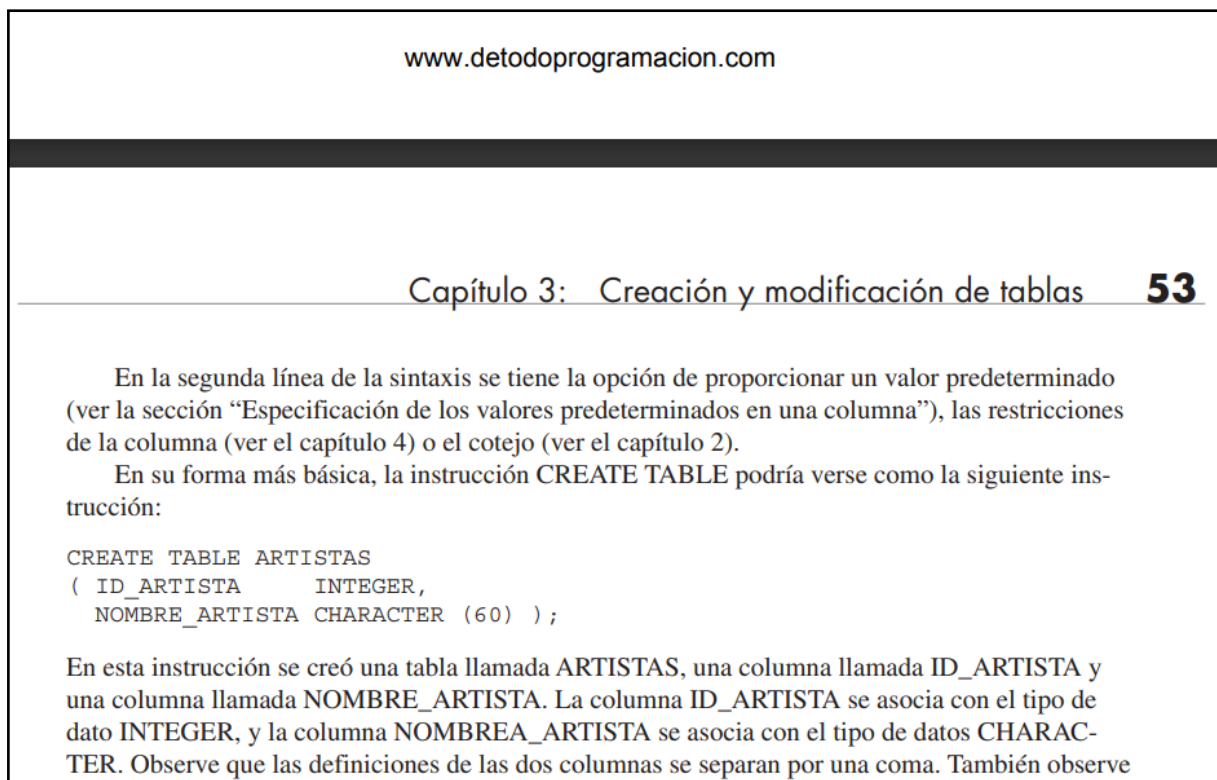


Figura 5.2: Ejemplo de fragmento de un PDF.

Una vez aplicado el OCR a la Figura 5.2 se obtiene el resultado en texto de la Figura 5.3.

**www.detodoprogramacion.com**

Capítulo 3: Creación y modificación de tablas 53

En la segunda línea de la sintaxis se tiene la opción de proporcionar un valor predeterminado (ver la sección “Especificación de los valores predeterminados en una columna”), las restricciones de la columna (ver el capítulo 4) o el cotejo (ver el capítulo 2). En su forma más básica, la instrucción CREATE TABLE podría verse como la siguiente instrucción:

```
CREATE TABLE ARTISTAS ( ID_ARTISTA INTEGER,
NOMBRE_ARTISTA CHARACTER (60) );
```

En esta instrucción se creó una tabla llamada ARTISTAS, una columna llamada ID\_ARTISTA y una columna llamada NOMBRE\_ARTISTA. La columna ID\_ARTISTA se asocia con el tipo de dato INTEGER, y la columna NOMBRE\_ARTISTA se asocia con el tipo de datos CHARACTER. Observe que las definiciones de las dos columnas se separan por una coma.

Tabla 5.3: Tabla de ejemplo de texto sin preprocesar

Para no perder calidad en los datos, primero, se emplea un preprocesamiento de todos los textos en el que se convierte el texto a minúsculas, se eliminan signos de puntuación, se tokeniza (dividir el texto en palabras) y se eliminan palabras vacías (stop words). En la Tabla 5.4 se muestra como quedaría el texto con este proceso de preprocesamiento.

wwwdetodoprogramacioncom capítulo 3 creación modificación tablas 53 segunda línea sintaxis opción proporcionar valor predeterminado ver sección “ especificación valores predeterminados columna ” restricciones columna ver capítulo 4 cotejo ver capítulo 2 forma básica instrucción create table podría verse siguiente ins trucción create table artistas idartista integer nombreartista character 60 instrucción creó tabla llamada artistas columna llamada idartista columna llamada nombreartista columna idartista asocia tipo dato integer columna nombreartista asocia tipo datos charac ter observe definiciones dos columnas separan coma

Tabla 5.4: Tabla de ejemplo de texto preprocesado

Tras realizar este trabajo se considera un error el preprocesamiento hecho a los textos conseguidos porque disminuyen el contexto y la coherencia, algo esencial para el objetivo de la técnica RAG. Entonces, se retrocede a los datos anteriores y para eliminar errores se realiza un preprocesamiento que solo consiste en eliminar caracteres especiales que puedan haber resultado de errores al convertir en formato '.txt' los datos ya que, el reconocimiento

empleado no siempre es preciso y puede no reconocer algunas palabras. Se decide que el script ahora solo elimine URLs utilizando una expresión regular y mantenga solamente caracteres alfabéticos, numéricos y signos de puntuación usando otra expresión regular. La Tabla 5.5 muestra el resultado final del texto del ejemplo preprocesado.

#### Capítulo 3 Creación y modificación de tablas 53

En la segunda línea de la sintaxis se tiene la opción de proporcionar un valor predeterminado. Ver la sección Especificación de los valores predeterminados en una columna, las restricciones de la columna (ver el capítulo 4) o el cotejo (ver el capítulo 2). En su forma más básica, la instrucción CREATE TABLE podría verse como la siguiente instrucción:

```
CREATE TABLE ARTISTAS (ID_ARTISTA INTEGER, NOMBRE_ARTISTA CHARACTER(60));
```

En esta instrucción se creó una tabla llamada ARTISTAS, una columna llamada ID\_ARTISTA y una columna llamada NOMBRE\_ARTISTA. La columna ID\_ARTISTA se asocia con el tipo de dato INTEGER, y la columna NOMBRE\_ARTISTA se asocia con el tipo de datos CHARACTER. Observe que las definiciones de las dos columnas se separan por una coma.

Tabla 5.5: Tabla de ejemplo de texto preprocesado corregido

Finalmente, el texto procesado es de una calidad considerablemente buena gracias a las herramientas de preprocesamiento que ofrece la biblioteca NLTK, los datos están listos para ser usados como contexto. Próximamente, se tratarán temas del procesamiento de estos datos para su uso en la técnica RAG.

### 5.2.4. Embeddings

Para llevar a cabo el proceso de generación de embeddings, se emplea Chromadb conjuntamente con un modelo denominado 'all-mpnet-base-v2'. Este último constituye un modelo integrado dentro de SentenceTransformers, un framework de Python. Este modelo tiene la capacidad de asignar frases y párrafos a un espacio vectorial denso de 768 dimensiones, siendo apto para diversas tareas tales como agrupación o búsqueda semántica. Así, se logra calcular y crear embeddings de manera eficaz.

MPNet introduce un método innovador de preentrenamiento conocido como modelado de lenguaje enmascarado y permutado. Este enfoque permite heredar las ventajas tanto del modelado de lenguaje enmascarado como del modelado de lenguaje permutado, contribuyendo así a la comprensión del lenguaje natural. MPNet aprovecha la relación entre los tokens predichos mediante el modelado de lenguaje permutado, además de incorporar información auxiliar de posición como parte de su entrada. Este proceso permite al modelo adquirir una

comprensión más holística de la frase completa y, en consecuencia, reducir la discrepancia de posición.

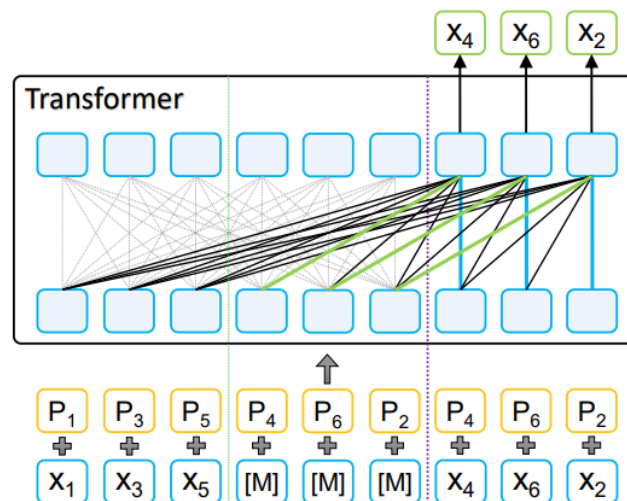


Figura 5.3: Diagrama de la estructura de MPNet.

Gracias a este proceso, la base de conocimiento puede ser usada por el RAG para poder encontrar la información necesaria para realizar búsquedas que estén relacionadas con la pregunta hecha por el usuario y de esta forma con la información que haya en la base de conocimiento da contexto al modelo generador de texto para responder con mayor calidad.

## 5.3. Evaluación y Ajustes

### 5.3.1. Parámetros de generación del modelo

Para la generación de texto, el modelo tienen una serie de parámetros que se ajustan en función de las necesidades y el rendimiento que se busque como objetivo. Para este proyecto se han probado distintos tipos de parámetros con el fin de ajustar el modelo hacia una respuesta más precisa y objetiva del modelo, en la siguiente lista se presentan los parámetros que finalmente han sido útiles para este fin.

- `prompt` — Texto con la pregunta formulada por el usuario.
- `do_sample` (`bool`, opcional, por defecto `False`) — Indica si se debe utilizar muestreo; utiliza decodificación greedy en caso contrario.
- `top_k` (`int`, opcional, por defecto 50) — Número de tokens de vocabulario con mayor probabilidad que se deben mantener para el filtrado top- $k$ .

- `num_return_sequences` (`int`, opcional, por defecto 1) — Número de secuencias independientemente calculadas que se devuelven para cada elemento en el batch.
- `eos_token_id` (`Union[int, List[int]]`, opcional) — El ID del token de fin de secuencia. Opcionalmente, utiliza una lista para establecer varios tokens de fin de secuencia.
- `max_length` (`int`, opcional, por defecto 20) — La longitud máxima que pueden tener los tokens generados. Corresponde a la longitud del prompt de entrada + `max_new_tokens`. Su efecto es anulado por `max_new_tokens`, si también está configurado.

### 5.3.2. Parámetros del RAG

En cuanto a los parámetros escogidos para el RAG, el divisor de texto ha sido determinante para organizar la información de cara a obtener búsquedas de calidad que apoyen al modelo.

Un divisor de texto se presenta cuando se busca fraccionar un extenso documento en porciones más reducidas que se ajusten a la ventana contextual del modelo. Es recomendable mantener cohesionados los fragmentos de texto que guarden relación semántica entre sí. La división del texto se realiza en porciones semánticamente significativas, usualmente frases.

El proceso empieza a combinar estos trozos pequeños en un trozo más grande hasta que alcanza un cierto tamaño. Una vez alcanzado ese tamaño, hace de ese trozo su propio fragmento de texto y empieza a crear un nuevo trozo de texto con cierto solapamiento para mantener el contexto entre los trozos. De este modo, es posible personalizar el proceso de división de texto:

1. **La modalidad de división textual.** Se ha usado `RecursiveCharacterTextSplitter`, este divisor de texto es el recomendado para texto genérico. Intenta dividirlos en orden hasta que los trozos son lo suficientemente pequeños. La lista por defecto es `["\n\n", "\n", , ]`. Esto tiene el efecto de tratar de mantener todos los párrafos (y luego las frases, y luego las palabras) juntos el mayor tiempo posible.
2. **El criterio empleado para determinar el tamaño del segmento.** Se usan los parámetros `chunk_size` para determinar el tamaño de la división del texto y `chunk_overlap` para determinar la longitud del solapamiento que se ha mencionado.

### 5.3.3. Prompting

En este apartado se tratará de ver la evolución que ha tenido el prompt template que acompaña al contexto y la pregunta del usuario para apreciar la importancia del prompting para mejorar la calidad de las respuestas de los modelos del lenguaje.

Para elaborar la forma del prompt base que se ha usado para preguntar al modelo se ha usado la técnica de uso de analogías, pidiendo al modelo que actúe como un profesor experto en la asignatura correspondiente y que tiene una situación característica que limita a que solo hable y traduzca al español.

Las palabras concretas que han sido seleccionadas para formar el prompt se han elegido mediante el recorrido iterativo de un plan de pruebas que recoge distintos tipos de problemas a resolver para que el prompt base tenga que adaptarse también al problema. La selección del prompt base de mayor calidad se ha realizado basándose en la supervisión humana dependiendo de la calidad con la que resolviera el problema y teniendo en cuenta ciertas palabras claves comúnmente usadas en la ingeniería de prompts.

En la Tabla 5.6 se puede leer en negrita, en el apartado de 'Prompt', un ejemplo de uno de las estructuras prompts probadas. En cambio, en la Tabla 5.7 se puede leer en negrita, en el apartado de 'Prompt', un ejemplo de una de las estructuras prompts probadas con peor calidad de respuesta.

Con estos dos ejemplos se puede apreciar la diferencia del empleo de técnicas con respecto a un prompt más común y sin depurar. Al final se presenta un documento como anexo que muestra todas las pruebas realizadas junto con el contexto de la prueba. Como estudio es interesante prestar atención a las palabras que se van añadiendo y eliminando, además del orden en el que se presentan estas palabras en el prompt.

<b>Prompt</b>	<p>Eres experto en MYSQL y bases de datos que traduce al español. Dame una respuesta en 150 palabras con el apartado 'Respuesta para españoles': '¿Cómo podría hacer una consulta de una lista con el nombre del producto, precio y nombre del fabricante de todos los productos de la base de datos? Esta es la estructura del código previo:</p> <pre> DROP DATABASE IF EXISTS tienda; CREATE DATABASE tienda CHARACTER SET utf8mb4; USE tienda; CREATE TABLE fabricante (   id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,   nombre VARCHAR(100) NOT NULL ); CREATE TABLE producto (   id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,   nombre VARCHAR(100) NOT NULL,   precio DOUBLE NOT NULL,   id_fabricante INT UNSIGNED NOT NULL,   FOREIGN KEY (id_fabricante)   REFERENCES fabricante(id) ); </pre>
<b>Respuesta</b>	<p>Para hacer una consulta de la lista de productos, precios y nombres de fabricantes, primero debes crear una tabla de fabricantes y otra de productos. Luego, puedes utilizar una consulta SQL para obtener los datos de ambas tablas y mostrarlos en una sola consulta. Por ejemplo, puedes utilizar la siguiente consulta:</p> <pre> "SELECT nombre_producto, precio, nombre_fabricante FROM producto JOIN fabricante ON producto.id_fabricante = fabricante.id". </pre>
<b>Análisis</b>	<p>Realizo una pregunta contextualizada, presentando quién es y sus capacidades, además incido en que la estructura debe ser en español. La respuesta es satisfactoria ya que, responde en el idioma en el que se le ha pedido y solucionando el problema.</p>

Tabla 5.6: Tabla de prueba de prompting. Resultado bueno

<p><b>Prompt</b></p>	<p>Eres un bot que tiene la capacidad de resolver problemas y traducirlos al español. ¿Cómo podría hacer una consulta de una lista con el nombre del producto, precio y nombre de fabricante de todos los productos de la base de datos?. Toma esto como base:</p> <pre> DROP DATABASE IF EXISTS tienda; CREATE DATABASE tienda CHARACTER SET utf8mb4; USE tienda; CREATE TABLE fabricante (   id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,   nombre VARCHAR(100) NOT NULL ); CREATE TABLE producto (   id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,   nombre VARCHAR(100) NOT NULL,   precio DOUBLE NOT NULL,   id_fabricante INT UNSIGNED NOT NULL,   FOREIGN KEY (id_fabricante) REFERENCES fabricante(id) ); </pre>
<p><b>Respuesta</b></p>	<p>To create a query that retrieves the name of the product, price, and manufacturer of all products in the database, you can use the following SQL statement:</p> <pre> SELECT p.nombre, p.precio, f.nombre AS fabricante FROM producto p JOIN fabricante f ON p.id_fabricante = f.id; </pre> <p>This will return a result set with three columns: "nombre"(the name of the product), "precio"(the price of the product), and "manufacturer"(the name of the manufacturer). Please let me know if you have any other questions.</p>
<p><b>Análisis</b></p>	<p>Realizo una pregunta contextualizada diciéndole las capacidades que debe tener para resolverlo. Busco que responda de forma compleja, extendida y natural. La respuesta no es adecuada ya que, no responde en el idioma que se le ha pedido.</p>

Tabla 5.7: Tabla de prueba de prompting. Resultado malo

## 5.4. Resultados y discusión

Para los experimentos realizaremos una evaluación supervisada por expertos, en este caso será revisado por la profesora de la asignatura de Fundamentos de Base de Datos. En esta sección veremos solo un par de experimentos a modo de ejemplo, más adelante, se adjunta un anexo correspondiente a los distintos experimentos realizados, siendo estas pruebas apoyadas por una tabla que facilite su comprensión.

Los experimentos se dividen en grupos:

- **Estudio y mejora del prompting:** Examina distintos problemas y formas de preguntar al modelo para identificar la estructura de prompt clave para mejorar la calidad y la precisión.
- **Análisis cualitativo de generación del modelo:** Examina ejemplos específicos de problemas para identificar diferencias en calidad, coherencia y relevancia.
- **Análisis cualitativo de generación del modelo junto con RAG:** Examina ejemplos específicos de problemas idénticos al anterior experimento para poder comparar los resultados.

Tras un ajuste previo, los parámetros que se han usado para dar los resultados son los siguientes:

Parámetro	Valor
<b>Pipeline</b>	
do_sample	True
top_k	10
num_return_sequences	1
torch_dtype	torch.float16
max_new_tokens	200
device_map	auto
max_length	200
<b>Divisor de texto</b>	
chunk_size	100
chunk_overlap	20

Tabla 5.8: Descripción de los parámetros.

Para evaluar los resultados se tendrá en cuenta:

- **Eficiencia y efectividad de la recuperación:** Medir y comparar el tiempo necesario para obtener resultados precisos utilizando el RAG y el modelo simple.

- **Interacción humana:** Evaluar la percepción de los usuarios sobre la calidad del contenido generado por el RAG.
- **Análisis cualitativo de ejemplos de generación:** Examinar ejemplos específicos de problemas para identificar diferencias en calidad, coherencia y relevancia.

#### 5.4.1. Estudio y mejora del prompting

En esta sección se verán tres ejemplos de diferentes estructuras de prompt de la misma forma que en el apartado anterior de 'Prompting', se han seleccionado estos ejemplos concretos para que sea visible la evolución de la estructura del prompt y la mejora de la calidad de la respuesta teniendo la misma pregunta. El prompt template o estructura fija del prompt se puede leer en negrita, en el apartado de 'Prompt' en cada una de las tablas. A continuación, se muestran los resultados de *Estudio y mejora del prompting*:

<p><b>Prompt</b></p>	<p>¿Cómo podría hacer una consulta de una lista con el nombre del producto, precio y nombre del fabricante de todos los productos de la base de datos? <b>Respóndeme en español.</b></p> <p>Toma esto como base:</p> <pre>DROP DATABASE IF EXISTS tienda; CREATE DATABASE tienda CHARACTER SET utf8mb4; USE tienda; CREATE TABLE fabricante (   id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,   nombre VARCHAR(100) NOT NULL ); CREATE TABLE producto (   id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,   nombre VARCHAR(100) NOT NULL,   precio DOUBLE NOT NULL,   id_fabricante INT UNSIGNED NOT NULL,   FOREIGN KEY (id_fabricante) REFERENCES fabricante(id) );</pre>
<p><b>Respuesta</b></p>	<p>You can use a SQL query to retrieve the name of the product, price, and manufacturer of all products in the database. Here is an example query:</p> <pre>SELECT p.nombre, p.precio, f.nombre FROM producto p JOIN fabricante f ON p.id_fabricante = f.id;</pre>
<p><b>Análisis</b></p>	<p>Realizo una pregunta directa, sin estructura de prompt fija que lo acompañe y añado que me responda en español. Busco que responda de forma extendida y natural para ver el comportamiento del modelo sin orientarlo mucho al problema. La respuesta no es en español y la explicación no se adecúa para un alumno.</p>

Tabla 5.9: Tabla de estudio y mejora del prompting. Prueba 1.

<p><b>Prompt</b></p>	<p>¿Cómo podría hacer una consulta de una lista con el nombre del producto, precio y nombre de fabricante de todos los productos de la base de datos?. <b>Esta es la estructura del código previo: + PROMPT + Traduce tu respuesta al español</b></p> <pre> DROP DATABASE IF EXISTS tienda; CREATE DATABASE tienda CHARACTER SET utf8mb4; USE tienda; CREATE TABLE fabricante (   id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,   nombre VARCHAR(100) NOT NULL ); CREATE TABLE producto (   id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,   nombre VARCHAR(100) NOT NULL,   precio DOUBLE NOT NULL,   id_fabricante INT UNSIGNED NOT NULL,   FOREIGN KEY (id_fabricante) REFERENCES fabricante(id) ); </pre> <p><b>Traduce tu respuesta al español</b></p>
<p><b>Respuesta</b></p>	<p>To make a query that retrieves the name of the product, price, and manufacturer of all products in the database, you can use a SELECT statement with the following syntax:</p> <pre> SELECT product_name, price, manufacturer_name FROM product; </pre> <p>This will retrieve all the columns (product_name, price, and manufacturer_name) from the product table and display them in a single result set. Additionally, you need to use the manufacturer_id column in the product table to reference the manufacturer.</p>
<p><b>Análisis</b></p>	<p>Realizo una pregunta directa. Busco que la respuesta sea extendida y natural, le recuerdo que responda en español al final para comprobar si presta más atención al final. La respuesta no es en español, tiene una buena explicación pero no resuelve bien el problema que se le plantea.</p>

Tabla 5.10: Tabla de estudio y mejora del prompting. Prueba 2.

<p><b>Prompt</b></p>	<p>Eres experto en <b>MYSQL</b> y bases de datos que traduce al español. La respuesta tendrá la estructura '<b>Respuesta para españoles: respuesta en español</b>' en <b>150 palabras</b>: ¿Cómo podría hacer una consulta de una lista con el nombre del producto, precio y nombre de fabricante de todos los productos de la base de datos?. <b>Esta es la estructura del código previo:</b></p> <pre> DROP DATABASE IF EXISTS tienda; CREATE DATABASE tienda CHARACTER SET utf8mb4; USE tienda; CREATE TABLE fabricante (   id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,   nombre VARCHAR(100) NOT NULL ); CREATE TABLE producto (   id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,   nombre VARCHAR(100) NOT NULL,   precio DOUBLE NOT NULL,   id_fabricante INT UNSIGNED NOT NULL,   FOREIGN KEY (id_fabricante)   REFERENCES fabricante(id) );         </pre>
<p><b>Respuesta</b></p>	<p>Respuesta para españoles: Para realizar una consulta de la lista de productos, precios y nombres de fabricantes, primero debes crear una tabla de fabricantes y otra de productos. Luego, puedes utilizar una consulta SQL para obtener los datos de ambas tablas y mostrarlos en una sola consulta. Por ejemplo, puedes utilizar la siguiente consulta:</p> <pre> SELECT nombre_producto, precio, nombre_fabricante FROM producto JOIN fabricante ON producto.id_fabricante = fabricante.id;         </pre> <p>Esto te devolverá los datos solicitados de todos los productos en la base de datos.</p>
<p><b>Análisis</b></p>	<p>Realizo una pregunta contextualizada diciéndole quien es y sus capacidades, además incido en que la estructura debe ser en español. Busco que responda en español seguro y ver qué puedo conservar del prompt para que siempre responda en español. La respuesta es en español, realiza una explicación adecuada y resuelve el problema.</p>

Tabla 5.11: Tabla de estudio y mejora del prompting. Prueba 3.

### 5.4.2. Análisis cualitativo de generación del modelo

En esta sección se verán cuatro ejemplos de diferentes problemas, se han seleccionado estos ejemplos concretos para que se aprecie la calidad de las respuestas del modelo del lenguaje sin apoyo del sistema RAG. De esta forma, se hace más visible el rendimiento del modelo en distintas circunstancias. A continuación, se muestran los resultados de *Análisis cualitativo de generación del modelo*.

<b>Prompt</b>	¿Qué significa SQL y cuál es su función principal en el ámbito de las bases de datos?
<b>Respuesta</b>	<p>.Einzeln, SQL (Structured Query Language) es un lenguaje de programación utilizado principalmente para manipular y trabajar con bases de datos. A continuación, te proporciono información más detallada sobre SQL y su función principal en el ámbito de las bases de datos.</p> <p><b>¿Qué es SQL?</b> SQL es un lenguaje de programación utilizado para interactuar con bases de datos relacionales. Fue desarrollado en la década de 1970 por el IBM para facilitar la manipulación de datos en bases de datos relacionales. A partir de entonces, SQL ha sido adoptado como el estándar de facto para interactuar con bases de datos en la mayoría de los sistemas operativos y aplicaciones.</p> <p><b>Función principal de SQL</b> La función principal del SQL es permitir a los usuarios interactuar con bases de datos para realizar diversas tareas, como:</p> <ol style="list-style-type: none"> <li><b>Consulta de datos:</b> SQL permite a los usuarios realizar consultas de datos en una base de datos para obtener información o para visualizar gráficamente los datos en una tabla.</li> <li><b>Modificación de datos:</b> SQL también permite a los usuarios realizar operaciones de modifying de datos en una base de datos, como agregar, modificar o eliminar reg</li> </ol>
<b>Tiempo</b>	22,722 sec.
<b>Evaluación</b>	La respuesta tiene puntos positivos al definir claramente SQL, explicar su función principal en las bases de datos y responder en español. Sin embargo, presenta errores gramaticales y ortográficos que afectan su calidad. Además, no termina bien la respuesta.

Tabla 5.12: Tabla de análisis cualitativo de generación del modelo. Prueba 1.

<b>Prompt</b>	¿Cuál es la diferencia entre SQL DDL y SQL DML?
<b>Respuesta</b>	<p>Ünterscheidung between SQL DDL and SQL DML.                  DDL (Data Definition Language) is a set of SQL statements used to create, modify, or drop data definition objects such as tables, indexes, views, and other database objects. DDL statements are used to define the structure and organization of the data in a database.                  On the other hand, DML (Data Manipulation Language) is a set of SQL statements used to insert, update, delete, or retrieve data from a database. DML statements are used to manipulate the data in a database, without altering the structure or organization of the data. Here are some key differences between SQL DDL and SQL DML:</p> <p><b>Purpose:</b> The purpose of DDL is to define the structure and organization of the data in a database, while the purpose of DML is to manipulate or modify the data in a database.  <b>Data Modification:</b> DDL statements modify the database schema, while DML statements modify the data in the database.  <b>Data Types:</b> DDL statements can create or modify database objects of different data types, while DML statements can only modify data of the same data type.  <b>Syntax:</b> The syntax of DDL statements is different from that of DML statements. DDL statements use a specific syntax to modify the database schema, while DML statements use a different syntax to manipulate the data.  <b>Data Consistency</b></p>
<b>Tiempo</b>	22,59 sec.
<b>Evaluacion</b>	<p>La respuesta tiene puntos positivos al definir claramente las diferencias y dar detalles adicionales que aclaren la respuesta. Sin embargo, no responde en español, presenta errores con palabras sin coherencia como 'Ünterscheidung' que afectan su calidad. Además, la respuesta no termina adecuadamente.</p>

Tabla 5.13: Tabla de análisis cualitativo de generación del modelo. Prueba 2.

<b>Prompt</b>	Imagina que tienes una tabla de empleados y necesitas encontrar el nombre y salario del empleado mejor pagado en cada departamento. ¿Cómo escribirías esta consulta en SQL?
<b>Respuesta</b>	<pre>SELECT department, employee_name, salary FROM employees ORDER BY salary DESC;</pre> <p>En este caso, la consulta busca el empleado mejor pagado en cada departamento. Para hacer esto, se utiliza el campo 'department' para filtrar los empleados por su departamentos y el campo 'salary' para ordenarlos por su salario en orden decreciente (menos a más). Es importante mencionar que en SQL, para acceder a los campos de una tabla se utiliza el nombre de la tabla seguido del punto (.) y los nombres de los campos se escriben con minúsculas.</p>
<b>Tiempo</b>	11.157 sec.
<b>Evaluacion</b>	Como punto positivo, la respuesta es en español, termina de forma adecuada y la consulta SQL presentada utiliza la cláusula ORDER BY, lo cual puede ser un paso en la dirección correcta de resolverlo. Sin embargo, la respuesta no logra el objetivo específico de encontrar al empleado mejor pagado en cada departamento.

Tabla 5.14: Tabla de análisis cualitativo de generación del modelo. Prueba 3.

<p><b>Prompt</b></p>	<p>Supongamos que estamos desarrollando un sistema de gestión de una biblioteca. Necesitamos una consulta SQL que nos permita encontrar los tres libros más prestados en los últimos seis meses. Además, queremos calcular el promedio de tiempo que cada libro ha estado prestado. ¿Cómo abordarías esta consulta, teniendo en cuenta la eficiencia y la precisión de los resultados?</p>
<p><b>Respuesta</b></p>	<p>Aquí te proporciono los datos de la tabla 'libros' de la biblioteca:   Id_libro   Titulo   Autores   Año_publicacion   Fecha_de_prestamo   Fecha_de_vuelta  </p> <p>Para abordar esta consulta, podemos utilizar una combinación de consultas SQL para obtener la información que necesitamos. Aquí te muestro una posible solución:</p> <pre>SELECT     RANK() OVER (PARTITION BY Id_libro     ORDER BY Fecha_de_prestamo DESC)     as Top_3_Prestados,     AVG(Fecha_de_vuelta - Fecha_de_prestamo)     as Promedio_Tiempo_Prestado FROM libros GROUP BY Id_libro</pre> <p>La consulta primero ordena los libros por el número de prestamos, utilizando la función RANK() para identificar los tres libros con más prestimos en los últimos seis meses. Luego, calcula el promedio de tiempo que cada libro ha estado prestado, utilizando la función AVG() para calcular la diferencia entre la fecha de vuelta y la fecha de prestamo. Esta consulta es eficiente</p>
<p><b>Tiempo</b></p>	<p>22.588 sec.</p>
<p><b>Evaluacion</b></p>	<p>La respuesta utiliza funciones SQL adecuadas para abordar los requisitos de la consulta. Además, responde en español y casi termina de forma adecuada. Como puntos negativos, falta una cláusula WHERE para filtrar los resultados por los últimos seis meses, como se solicita en el problema y sería útil agregar una cláusula ORDER BY para asegurar que los libros más prestados se presenten en orden descendente.</p>

Tabla 5.15: Tabla de análisis cualitativo de generación del modelo. Prueba 4.

### 5.4.3. Análisis cualitativo de generación del modelo junto con RAG

En esta sección se verán cuatro ejemplos de diferentes problemas, se han seleccionado estos ejemplos concretos para que se aprecie la calidad de las respuestas del modelo del lenguaje con apoyo del sistema RAG. De esta forma, se hace más visible el rendimiento del modelo en distintas circunstancias. También, se pueden ver pequeños ejemplos de fragmentos que recupera el sistema RAG para darle contexto al modelo en el apartado de 'Contexto' de cada tabla. A continuación, se muestran los resultados de *Análisis cualitativo de generación del modelo junto con RAG*.

<b>Contexto</b>	[[Document(page_content='Una de las características que definen a SQL es el hecho de que los datos en una base de', metadata='source': '/mnt/beegfs/aha00026/bot/TutorBot/proyecto-entorno/texto/fundamentos_de_sql_3edi_oppel.txt')]]
<b>Prompt</b>	¿Qué significa SQL y cuál es su función principal en el ámbito de las bases de datos?
<b>Respuesta</b>	SQL es el acrónimo de Structured Query Language, es la lingua franca de las bases de datos. Su función principal es permitir a los usuarios interactuar con las bases de datos de manera eficiente y segura, mediante consultas y operaciones en los datos almacenados en la base de datos.
<b>Tiempo</b>	22.6 sec.
<b>Evaluación</b>	La respuesta proporcionada es en español y es correcta. Además, la respuesta concluye adecuadamente después de explicar la función principal de SQL en las bases de datos. El único punto negativo es la brevedad de la respuesta.

Tabla 5.16: Tabla de análisis cualitativo de generación del modelo junto con RAG. Prueba 1.

<b>Contexto</b>	[[Document(page_content='única diferencia radica en cómo son utilizadas por la instrucción SQL (ya que son definidas', metadata='source': '/mnt/beegfs/aha00026/bot/TutorBot/proyecto-entorno/texto/fundamentos_de_sql_3edi_oppel.txt')]]
<b>Prompt</b>	¿Cuál es la diferencia entre SQL DDL y SQL DML?
<b>Respuesta</b>	<p>La diferencia entre SQL DDL y SQL DML es fundamental en el ámbito de la programación y la gestión de bases de datos. DDL (Data Definition Language) se refiere a las instrucciones utilizadas para definir la estructura de una base de datos, mientras que DML (Data Manipulation Language) se utiliza para manipular los datos en una base de datos.</p> <p>DDL es utilizado para definir la estructura de una base de datos, incluyendo la creación de tablas, columnas, índices, restricciones y relaciones. Estas instrucciones son utilizadas para definir la estructura de la base de datos y establecer las reglas que gobiernan su uso. Por ejemplo, la instrucción 'CREATE TABLE' es una instrucción DDL que se utiliza para crear una nueva tabla en una base de datos.</p> <p>DML, por otro lado, es utilizado para manipular los datos en una base de datos. Estas instrucciones incluyen operaciones como insertar, actualizar y eliminar datos en una tabla. Por ejemplo, la instrucción 'INSERT INTO' es una instrucción DML que se utiliza para insertar datos en una tabla.</p> <p>La principal diferencia entre DDL y DML es que DDL se utiliza para definir la estructura de la base de datos y DML se utiliza para manipular los datos dentro de esa estructura.</p>
<b>Tiempo</b>	22.31 sec.
<b>Evaluacion</b>	La respuesta es en español y aborda correctamente las diferencias. Además, la respuesta concluye de manera adecuada al resumir la diferencia fundamental entre DDL y DML y sus utilidades.

Tabla 5.17: Tabla de análisis cualitativo de generación del modelo junto con RAG. Prueba 2.

<b>Contexto</b>	[[Document(page_content='se encuentran y el teléfono . ha brá que tener en cuenta que un departamento con el mismo nombre puede estar en ciudades diferentes y que en una misma ciudad puede haber departa mentos con nombres diferentes . 3. de los proyectos informáticos que se desarrollen , queremos saber el códi go , el nombre , el precio , la fecha de inicio , la fecha prevista de finalización , la fecha real de finalización y el código de cliente para quien se desarrolla . sql nos ofrece se concentran en la creación , la modificación y el muchos de los sistemas relaci borrado de', metadata='source': '/mnt/beegfs/aha00026/bot/TutorBot/proyecto-entorno/texto-preprocesado/plano_Bases-de-datos _Módulo-3_ El-lenguaje-SQL.txt')]]
<b>Prompt</b>	Imagina que tienes una tabla de empleados y necesitas encontrar el nombre y salario del empleado mejor pagado en cada departamento. ¿Cómo escribirías esta consulta en SQL?
<b>Respuesta</b>	<p>Para obtener el nombre y salario del empleado mejor pagado en cada departamento, podemos utilizar una consulta SQL como la siguiente:</p> <pre>SELECT department, name, salary FROM employees;</pre> <p>En esta consulta, estamos utilizando la columna "department" para filtrar los empleados por departamento, y la columna "name" y "salary" para obtener el nombre y salario de cada empleado. Por supuesto, esta consulta solo dará como resultado el empleado mejor pagado en cada departamento. Si deseamos obtener el nombre y salario de todos los empleados, podemos utilizar una consulta similar pero sin la filtración por departamento:</p> <pre>SELECT name, salary FROM employees;</pre> <p>Espero que esta respuesta te sea útil. Si tienes alguna otra pregunta, no dudes en preguntar.</p>
<b>Tiempo</b>	16.987 sec.

<b>Evaluacion</b>	La respuesta es en español y termina adecuadamente. El problema es que la consulta presentada simplemente selecciona todas las filas de la tabla de empleados. La respuesta dada en el prompt no aborda correctamente la forma de encontrar al empleado mejor pagado en cada departamento, por lo que no responde de manera satisfactoria la pregunta planteada.
-------------------	--

Tabla 5.18: Tabla de análisis cualitativo de generación del modelo junto con RAG. Prueba 3.

<b>Contexto</b>	[[Document(page_content='Consulta de datos de SQL', metadata='source': '/mnt/beegfs/aha00026/bot/TutorBot/proyecto-entorno/texto/fundamentos_de_sql_3edi_oppel.txt')]]
<b>Prompt</b>	Supongamos que estamos desarrollando un sistema de gestión de una biblioteca. Necesitamos una consulta SQL que nos permita encontrar los tres libros más prestados en los últimos seis meses. Además, queremos calcular el promedio de tiempo que cada libro ha estado prestado. ¿Cómo abordarías esta consulta, teniendo en cuenta la eficiencia y la precisión de los resultados?
<b>Respuesta</b>	<p>Para abordar esta consulta, podemos utilizar una consulta SQL que utilice una combinación de funciones de agregación y filtros para obtener los resultados deseados. Aquí está la consulta que propongo:</p> <pre>SELECT     COUNT(DISTINCT book_id)     AS num_books,     AVG(borrow_date - LOW_DATE)     AS avg_borrow_time FROM books JOIN borrowing ON books.book_id = borrowing.book_id WHERE     borrowing.borrow_date &gt;= DATE_SUB(CURRENT_DATE, INTERVAL 6 MONTH) GROUP BY book_id</pre> <p>En esta consulta, primero utilizamos la función COUNT(DISTINCT) para contar el número de libros que han sido prestados en los últimos seis meses. Luego, utilizamos la función AVG para calcular el promedio de tiempo que cada libro ha estado prestado, calculando la diferencia entre la fecha de préstamo y la fecha de devolución para cada libro y sumando los valores para obtener el promedio. La consulta utiliza una estructura de join inner para unir las tablas de libros y de préstamo.</p>
<b>Tiempo</b>	22.318 sec.

<b>Evaluacion</b>	La respuesta es en español pero no termina adecuadamente y no aborda completamente la pregunta del prompt y no incluye la búsqueda de los tres libros más prestados en los últimos seis meses ni el cálculo del promedio de tiempo que cada libro ha estado prestado.
-------------------	---

Tabla 5.19: Tabla de análisis cualitativo de generación del modelo junto con RAG. Prueba 4.

# Capítulo 6

## Conclusiones

En la sección final de la memoria, llevaremos a cabo una reflexión sobre una serie de conclusiones obtenidas de la realización de este proyecto, los aprendizajes obtenidos durante el desarrollo del mismo y concluiremos con las posibles mejoras a implementar en el futuro.

### 6.1. Conclusión final

Este proyecto ha puesto de manifiesto el potencial transformador de los modelos RAG en el ámbito del PLN. Por ello que se han conseguido los siguientes logros:

- **Validación de los sistemas RAG**
  - Se evidenció cómo estos modelos del lenguaje, junto con el RAG, ofrecen una mayor consistencia en los resultados generados y reducen significativamente la posibilidad de errores o información sensible incorrecta por parte de los LLM.
  - El uso del RAG también ha proporcionado a los usuarios un medio efectivo para verificar la precisión de las respuestas generadas al extraer hechos de fuentes de conocimiento externas y verificables.
- **Exploración y mejora de técnicas de modelado**
  - Se exploraron diversas técnicas y estrategias para mejorar el rendimiento de los modelos de lenguaje. Estas técnicas se complementan de manera eficaz con el método convencional de fine-tuning, proporcionando nuevas herramientas y enfoques para optimizar el desempeño.
- **Adquisición de conocimientos profundos en PLN**
  - El proyecto ha sido una oportunidad para adquirir un profundo entendimiento del funcionamiento de los modelos de lenguaje y las herramientas del PLN. Se

profundizó especialmente en la ingeniería de prompts, el uso de diversas bibliotecas para el desarrollo de LLMs y los sistemas RAG.

#### ■ **Aplicación práctica de conceptos aprendidos**

- Se aplicaron activamente conceptos aprendidos durante la carrera en un contexto práctico y profesional.
- Utilicé metodologías aprendidas en 'Fundamentos de Ingeniería del Software', 'Ingeniería de requisitos' y 'Gestión de proyectos software' para planificar y ejecutar eficientemente el proyecto, definiendo objetivos, realizando diagramas, asignando recursos, planificando tareas y realizando seguimiento del progreso.
- Apliqué conocimientos de las asignaturas de programación y desarrollo ('Fundamentos de Programación', 'Estructuras de Datos', ...) para diseñar e implementar el sistema RAG, incluyendo diseño de la arquitectura del sistema y codificación de funcionalidades.
- Utilicé técnicas aprendidas en 'Sistemas Operativos' para comprender los requisitos del LLM y el sistema RAG en conjunto y el funcionamiento del clúster ADA.
- Apliqué conocimientos de 'Programación y Administración de Redes', comprendiendo el intercambio de información entre los componentes del sistema ADA y mi entorno de desarrollo.
- Usé herramientas y conocimientos de la asignatura de 'PLN' para la comprensión del funcionamiento de los LLM y para la manipulación eficiente de datos de texto. Esto incluyó el uso de modelos para mejorar la capacidad de análisis y respuesta del sistema.

#### ■ **Exploración de áreas de proyección industrial**

- El proyecto me ha proporcionado una visión detallada de las tecnologías de vanguardia en el área del PLN. Siendo esta una de las áreas con mayor relevancia y proyección en el ámbito industrial y empresarial, permitiéndome el estudio de tecnología que están en el estado del arte.

#### ■ **Cumplimiento de objetivos generales del proyecto**

- Se lograron cumplir los objetivos generales del proyecto. Esto fue posible gracias al empleo de la metodología ágil, que permitió adaptarse a los desafíos surgidos durante el desarrollo, manteniendo coherencia con los objetivos generales.
- Se llevó a cabo una gestión efectiva del proyecto, abordando los desafíos de manera proactiva para garantizar la entrega exitosa del trabajo.

En conclusión, este proyecto ha sido un ejercicio significativo de investigación y desarrollo en el campo del Procesamiento del Lenguaje Natural. Tras el estudio, se considera que la tecnología RAG se vislumbra como un agente de gran apoyo para el futuro del PLN y la generación de lenguaje artificial en general. Con su capacidad para proporcionar respuestas informativas, actualizadas y verificables, los modelos RAG están llevando este campo a un nuevo nivel de precisión y eficiencia.

Aunque en la actualidad los modelos RAG están restringidos a realizar consultas de texto y obtener fragmentos de información de bases de conocimiento textual, existe la posibilidad de que en el futuro puedan interactuar con bases de datos más complejas y multidimensionales, lo que ampliaría su capacidad para ofrecer respuestas precisas y contextualmente relevantes.

## 6.2. Mejoras futuras

A continuación trataremos las posibles mejoras y nuevas funcionalidades a incorporar:

### 1. **Mejora en la calidad de los datos y ampliación de la base de conocimiento para más asignaturas**

Implica mejorar la calidad de los datos utilizados para apoyar al modelo, así como ampliar la base de conocimiento para incluir información relevante de una variedad más amplia de asignaturas. Esto puede lograrse mediante la incorporación de datos de asignaturas adicionales para garantizar que el modelo tenga acceso a una base de conocimiento más completa y diversa y de esta forma la herramienta puede ser usada de manera global para más alumnos y profesores.

### 2. **Recopilación y análisis de las preguntas recibidas para dar retroalimentación a los profesores de las distintas asignaturas**

Esta funcionalidad implica analizar las preguntas recibidas de forma estadística y comprensiva, gracias a ello, los profesores podrán identificar áreas temáticas específicas donde los estudiantes puedan necesitar más ayuda o clarificación, lo que les permitirá adaptar mejor su enseñanza y brindar un mejor apoyo a los estudiantes.

### 3. **Incorporar el modelo a un servicio en la nube para que pueda tener conexión a Internet**

Se incorpora acceso a Internet, por lo que la comunicación con APIs podría plantearse en este formato y con ello dar utilidad a la herramienta planteando a partir de esta mejora la integración con chats de fácil acceso para los usuarios. Como desventaja, supone un gran coste extra al proyecto en consecuencia de un uso real al tener que generar tokens en plataformas que requieren de pago en función del coste computacional.

#### 4. Integración con chat de PLATEA

Esta mejora permite a los usuarios acceder al modelo directamente a través de la plataforma de docencia virtual de la universidad de Jaén, llamada PLATEA. Al integrar el modelo con la plataforma pueden realizar consultas y recibir respuestas en tiempo real dentro del entorno familiar de la plataforma PLATEA, lo que facilita su uso y acceso para los estudiantes y profesores que ya utilizan esta plataforma para sus actividades educativas. Además, no requeriría el coste de un desarrollo web completo debido a que PLATEA ya está en funcionamiento y con un equipo de trabajo que le da soporte.

# Apéndice A

## Tablas de resultados

### A.1. Experimentos

En este anexo se presentan los resultados que se han tenido durante el desarrollo del proyecto. Los resultados abarcan diversas pruebas similares a las vistas anteriormente que proporcionan una comprensión más profunda de los experimentos realizados. Se incluirán dos ejemplos en cada apartado para tener una visión algo más clara de la calidad de las respuestas del modelo.

### A.1.1. Análisis cualitativo de generación del modelo

<b>Prompt</b>	¿Cuáles son las cláusulas más comunes en SQL y cuál es su propósito (por ejemplo, SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY)?».
<b>Respuesta</b>	Stack Overflow en español. obviously, it will give you the SQL syntax for each of these clauses, but I'll also explain their purpose and how they work. Here are some of the most common clauses in SQL: 1. SELECT: The SELECT clause is used to retrieve data from a database table. You can specify which columns you want to include in the output, as well as any conditions or filters for the data you want to retrieve. For example: "" SELECT * FROM customers WHERE country = 'USA'; "" In this example, we are selecting all columns (*) from the 'customers' table where the 'country' column is equal to 'USA'. 2. FROM: The FROM clause is used to specify the table or tables you want to query. You can use the 'FROM' clause together with the 'SELECT' clause to retrieve data from multiple tables. For example: "" SELECT * FROM customers, orders WHERE customers.customer_id = orders.customer_id; "" In this example, we are selecting data from the 'customers' and 'orders' tables where the 'customer_id' column in the 'customers' table matches the 'customer_id' column in the 'orders' table. 3. WHERE: The WHERE clause is used to specify conditions or filters for the data you want to retrieve. You can use the 'WHERE' clause to filter data based on
<b>Evaluacion</b>	La respuesta es en inglés. La respuesta es útil pero incompleta y tiene áreas que podrían causar confusión. Aunque cubre las cláusulas SELECT, FROM y WHERE con ejemplos, omite importantes cláusulas como GROUP BY, HAVING y ORDER BY. Además, la mención inicial a 'Stack Overflow en español' es irrelevante. Los ejemplos también presentan inconsistencias en el uso de comillas.

Tabla A.1: Tabla de análisis cualitativo de generación del modelo. Anexo(1)

<b>Prompt</b>	¿Cómo se pueden manejar las transacciones en SQL para garantizar la integridad de los datos?
<b>Respuesta</b>	<p>everybody knows that the transactions are a fundamental concept in database management systems, because they allow to manage the data consistency and integrity. In this article, we will explore how to manage transactions in SQL to guarantee the integrity of the data. First of all, it is important to understand the concept of transaction in SQL. A transaction is a sequence of operations that are executed as a single, atomic unit of work. This means that either all operations inside a transaction are executed and committed, or none of them are. This property ensures data consistency and integrity, because if any operation inside a transaction fails, all operations will be rolled back, restoring the database to its previous state. There are several ways to manage transactions in SQL, including:</p> <ol style="list-style-type: none"> <li>1. Using the SET TRANSACTION statement: In SQL Server, you can use the SET TRANSACTION statement to begin a new transaction. For example: BEGIN TRANSACTION;</li> <li>2. Using transactions implicit: In some databases, such as SQL Server, you can use transactions implicit. To do this, you simply call a stored procedure or execute a batch of SQL commands as you normally would. The database will automatically begin a new transaction.</li> <li>3. Using the COMMIT and ROLLBACK statements: After executing a transaction, you can use the COMMIT statement to confirm the changes and make them permanent, or the ROLLBACK statement to undo the changes and restore the</li> </ol>
<b>Evaluacion</b>	La respuesta es en inglés y la respuesta concluye de manera inadecuada. La respuesta sobre el manejo de transacciones en SQL es adecuada en términos de conceptos básicos, pero tiene varias deficiencias al carecer de detalles importantes como los niveles de aislamiento de transacciones.

Tabla A.2: Tabla de análisis cualitativo de generación del modelo. Anexo(2)

### A.1.2. Análisis cualitativo de generación del modelo junto con RAG

<b>Prompt</b>	Estoy diseñando una base de datos para una aplicación de gestión de inventario. ¿Qué tipo de restricciones de integridad de datos podrían aplicarse a las tablas para mantener la coherencia de los datos?
<b>Respuesta</b>	<p>Hay varias tipos de restricciones de integridad de datos que podrías aplicar a las tablas de tu base de datos para mantener la coherencia de los datos. Algunas de las más comunes son:</p> <ol style="list-style-type: none"> <li>1. Primero, la restricción de valores únicos (UNIQUE) garantiza que cada columna de datos tenga un valor único en cada fila de la tabla. Esto es importante para evitar la duplicación de valores en la base de datos.</li> <li>2. La restricción de referencia (FOREIGN KEY) establece una relación entre dos tablas a través de un valor de clave foránea. Esto ayuda a evitar la asignación de valores incorrectos a las columnas de referencia.</li> <li>3. La restricción de consistencia (CONSTRAINT) es una restricción más general que se aplica a una o más columnas de una tabla. Esto puede ser utilizado para garantizar que los valores de una columna estén dentro de un rango específico o que una columna tenga un valor determinado antes de poder insertar un nuevo registro.</li> <li>4. La restricción de integridad temporal (CHECK) es una restricción que se aplica a una columna y garant</li> </ol>
<b>Evaluacion</b>	La respuesta es en español y concluye de manera incorrecta. La respuesta sobre las restricciones de integridad de datos aborda algunos conceptos clave pero omite otras restricciones esenciales como NOT NULL y PRIMARY KEY. Se podría mejorar, las explicaciones carecen de ejemplos prácticos.

Tabla A.3: Tabla de análisis cualitativo de generación del modelo junto con RAG. Anexo(1)

<b>Prompt</b>	¿Cómo se pueden manejar las transacciones en SQL para garantizar la integridad de los datos? "La integridad de los datos es un aspecto fundamental en cualquier base de datos. Para garantizar la integridad de los datos, SQL proporciona varios métodos para manejar las transacciones.
<b>Respuesta</b>	Uno de los métodos más comunes es utilizar la cláusula BEGIN TRANSACTION para iniciar una transacción. Durante la transacción, cualquier operación realizada en la base de datos se verifica y se aplica de forma consistente. Si la transacción falla, se puede utilizar la cláusula ROLLBACK para deshacer los cambios realizados durante la transacción. Otro método para manejar las transacciones es utilizar la cláusula SAVE POINT. Esta cláusula permite establecer un punto de salvamento en una transacción, lo que permite deshacer los cambios realizados desde el punto de salvamento en caso de que la transacción falle. Además, SQL también proporciona la cláusula SERIALIZABLE para garantizar la integridad de los datos en transacciones. Esta cláusula garantiza que solo se pueden realizar operaciones en una base de datos de forma serial, es decir, de forma secuencial y sin interrupciones. Es importante destacar que la
<b>Evaluación</b>	La respuesta es en español y concluye de manera inadecuada. La respuesta cubre algunos conceptos clave pero tiene deficiencias de claridad y carece de ejemplos prácticos.

Tabla A.4: Tabla de análisis cualitativo de generación del modelo junto con RAG. Anexo(2)

# Apéndice B

## Manual de usuario

### B.1. Manual de ADA

Para la instalación y uso del modelo se requiere una gran capacidad de computación, en este proyecto, como ya se ha mencionado previamente se ha usado el clúster ADA. Este clúster ya cuenta con su propio manual de usuario para su uso, en este manual de usuario se usarán las partes importantes de este mismo y se añadirán las necesarias para instalar el resto del proyecto.

Como primer paso, las cuentas de usuario se crearán de forma personal. Cada usuario deberá disponer de una cuenta en el servicio. Existen dos tipos de usuario en el servicio:

1. **Usuarios permanentes:** Personal fijo adscrito a CEATIC. Desde el momento de creación de su cuenta, dispondrán de ella sin fecha de expiración.
2. **Usuarios temporales:** Los usuarios que no dispongan de una vinculación permanente tendrán una fecha de expiración en sus cuentas, que será igual a la fecha de finalización del proyecto/contrato al que se encuentren vinculados. Si se realizara una prórroga o renovación, deberán solicitar una reactivación de su cuenta en el servicio.

El procedimiento para la solicitud de nuevas cuentas de usuario estará disponible en la web del CEATIC.

### B.2. Información básica de conexión

El servicio habilitado para el acceso remoto al sistema es SSH. Dicho servicio únicamente se encuentra disponible desde la red de la UJA, por lo que si se desea acceder desde fuera es necesario conectar primero al servicio VPN-SSL UJA.

- **Host:** ada01.ujaen.es
- **Puerto:** 22

### B.2.1. Entornos Virtuales

Para trabajar con Python y múltiples versiones de sus librerías se utilizan entornos virtuales. Con esto se consigue evitar los conflictos entre diferentes proyectos e incluso entre diferentes versiones de los mismos. Cada usuario puede crear tantos entornos virtuales como necesite.

La recomendación para el trabajo con entornos es utilizar conda, puesto que permite al usuario crear entornos para una versión concreta de Python, siendo más flexible que virtualenv. Para la realización de los siguientes pasos se opta por el uso de miniconda3, el cual se encuentra instalado como software base del clúster.

### B.2.2. Creación y activación de entornos

```
# Carga de miniconda3
spack load miniconda3

# Verificación de versión de miniconda
conda -V

# Creación de entorno virtual con una versión de Python
conda create -n NOMBRE-ENTORNO python=x.x

# Activación de entorno virtual
source activate NOMBRE-ENTORNO

# Desactivación de entorno virtual
source deactivate

# Eliminación de entorno virtual
conda remove --name NOMBRE-ENTORNO --all
```

### B.2.3. Instalación de librerías en entornos virtuales

Se dispone de dos modos para instalar librerías de Python en los entornos virtuales:

- `conda install`

- `pip install`

Hay ocasiones en las cuales ya hemos definido un entorno localmente que funciona correctamente con nuestro proyecto y queremos replicarlo en el clúster. Para ello exportamos la configuración local en TXT, creamos un nuevo entorno en el clúster e importamos dicha configuración.

```
# En el equipo local se exporta la configuración
pip freeze --local > requirements.txt
```

```
# En el clúster se crea el entorno virtual
# Se siguen los pasos definidos en 4.1.1
```

```
# Con el entorno activo, se instala la configuración
pip install --find-links requirements.txt
```

## B.2.4. Comandos básicos de Slurm

Slurm es un sistema para la planificación de tareas y gestión de clústeres de código abierto, tolerante a fallos y altamente escalable.

### Comando Descripción

`sbatch` Envía un proceso por lotes Slurm para que lo asigne a una partición (cola) cuando encuentre recursos para ello.

```
$ sbatch script.sh
```

`srun` Permite ejecutar pasos dentro de un script de Slurm.

`scancel` Permite cancelar trabajos de Slurm.

```
$ scancel <jobid>
```

`squeue` Muestra información sobre los trabajos en cola.

```
$ squeue -j <jobid>
```

```
$ squeue -start -u <username>
```

```
$ squeue -t <RUNNING,PENDING>
```

`sinfo` Muestra información del estado de los nodos y las particiones.

#### Posibles estados de los nodos:

IDLE: Nodo disponible.

ALLOC: Nodo en uso.  
MIX: El nodo tiene recursos disponibles.  
RESV: Nodo reservado.  
DRAIN: Nodo retirado por motivos técnicos.

scontrol Muestra información de todos los trabajos.  
\$ scontrol show jobs  
Muestra información de un trabajo concreto.  
\$ scontrol show job JOBID

### B.2.5. Definición de los script de lanzamiento

Para ejecutar trabajos en Slurm es necesario definir scripts de lanzamiento, para los cuales utilizaremos como estándar la extensión “.sbs”. Estos scripts pueden contener tanto código bash (recomendado) como python, definiéndolo en la primera línea del fichero.

La cabecera de los ficheros “.sbs” está compuesta por una serie de líneas en las cuales se indican los recursos que se solicitan al clúster para la ejecución (reserva). Es importante definir unos recursos realistas puesto que podemos encontrar los siguientes casos:

- Si se solicitan más recursos de los existentes en el clúster fallará la ejecución.
- Si se solicitan más recursos de los que se encuentren disponibles en el momento actual el trabajo se colocará en la cola hasta que los recursos solicitados se encuentren disponibles.
- Si durante el trabajo se utilizan recursos no reservados, lo más probable es que la ejecución del trabajo falle.

A continuación se describen tres términos importantes para Slurm:

- **Job:** Trabajos lanzados con el comando `sbatch` a través de un script. Es el problema global a resolver, para el que solicitamos una serie de recursos al sistema. En el siguiente enlace se pueden encontrar los parámetros que acepta `sbatch`: <https://slurm.schedmd.com/sbatch.html>
- **Step:** Pasos creados con el comando `srun` dentro del trabajo. Indican la secuencia de tareas a realizar.
- **Task:** Tareas ejecutadas en cada paso. Son solicitadas a nivel de trabajo o de paso con `-ntasks`.

En el ejemplo vamos a solicitar una GPU para la ejecución de nuestra tarea.

```
#!/bin/bash
SBATCH --job-name=ejemplo2
SBATCH --partition=normal
SBATCH --nodes=1
SBATCH --time=00:02:00
SBATCH --ntasks=1
SBATCH --mem=1G
SBATCH --gres=gpu:1
SBATCH --gres=gpu:rtx:1 (ejemplo para una RTX)
SBATCH --gres=gpu:tesla:1 (ejemplo para una TESLA V100)
SBATCH --gres=gpu:ampere:1 (ejemplo para una TESLA A100)
```

(Opcional) Limpieza de módulos cargados previamente  
module purge

(Opcional) Carga de módulos software, usualmente cuda  
spack load NOMBRE-MODULO

(Opcional) Activación de entorno virtual de Conda  
eval "\\$(conda shell.bash hook)"  
conda activate NOMBRE-ENTORNO

Ejecución del primer paso  
srun miprogramadeejemplo1.sh

Para lanzar los trabajos definidos en ficheros ".sbs":

```
sbatch ejemplo1.sbs
```

## B.2.6. Descarga de modelos

Tras haber realizado la creación del entorno virtual y haber conocido todo lo necesario para usar ADA, se deben descargar los distintos modelos que se han usado:

- all-mpnet-base-v2: Para la creación de los embeddings. Se descarga en el enlace de Hugging Face que se adjunta: <https://huggingface.co/sentence-transformers/all-mpnet-base-v2>
- LLaMa 2: Modelo de generación del lenguaje. Se puede descargar en este enlace con las instrucciones que se han explicado previamente en el apartado 5.1. Enlace: <https://huggingface.co/meta-llama/Llama-2-7b-hf>

Tras esto, se suben los archivos al clúster con el siguiente comando:

```
scp origen usuario@ada01.ujaen.es:/mnt/beegfs/usuario/destino
```

Además, con este comando se podrá subir la base de conocimiento incluida en la carpeta 'dataset-sql' para usarla en el sistema RAG.

### **B.2.7. Ejecución**

Para ejecutar el fichero correspondiente al trabajo de este proyecto se debe usar el launcher que incluye el propio código y ejecutar con este launcher el archivo 'rag.py'. A partir de aquí, las modificaciones de las preguntas y pruebas son completamente libres y se podrán hacer en el archivo 'rag.py'.

# Apéndice C

## Bibliografía

- [1] L. A. Ureña. Apuntes de la asignatura procesamiento del lenguaje natural. *Universidad de Jaén*, 2019.
- [2] Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. Annotation artifacts in natural language inference data. *arXiv preprint arXiv:1803.02324*, 2018.
- [3] Bill Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*, 2005.
- [4] Erik F Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*, 2003.
- [5] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.
- [6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [8] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner,

- Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.
- [9] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.
- [10] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- [11] William Oswaldo Aparicio Gómez. La inteligencia artificial y su incidencia en la educación: Transformando el aprendizaje para el siglo xxi. *Revista Internacional de Pedagogía e Innovación Educativa*, 3(2):217–229, jul. 2023.
- [12] Laria Reynolds and Kyle McDonell. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–7, 2021.
- [13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [14] Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*, 2023.
- [15] Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*, 2021.
- [16] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.
- [17] Angels Balaguer, Vinamra Benara, Renato Luiz de Freitas Cunha, Roberto de M Estevão Filho, Todd Hendry, Daniel Holstein, Jennifer Marsman, Nick Mecklenburg, Sara Malvar, Leonardo O Nunes, et al. Rag vs fine-tuning: Pipelines, tradeoffs, and a case study on agriculture. *arXiv e-prints*, pages arXiv–2401, 2024.

- [18] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Hongyi Jin, Tianqi Chen, and Zhihao Jia. Towards efficient generative large language model serving: A survey from algorithms to systems. *arXiv preprint arXiv:2312.15234*, 2023.
- [19] Dan Hendrycks, Chris Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.
- [20] Hsiu-Wei Chung, Liang Hou, Simon Longpre, Barret Zoph, Yi Tay, William Fedus, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.
- [21] Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models, 2023.
- [22] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- [23] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. *Advances in neural information processing systems*, 27, 2014.
- [24] Xin Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.
- [25] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MpNet: Masked and permuted pre-training for language understanding. *Advances in neural information processing systems*, 33:16857–16867, 2020.
- [26] Steven Bird, Ewan Klein, and Edward Loper. Natural Language Toolkit, 2009. URL: <https://www.nltk.org/>.
- [27] Google. Google Colaboratory. <https://colab.research.google.com>.
- [28] @cbh123. A guide to prompting llama 2, 2023. <https://replicate.com/blog/how-to-prompt-llama>.
- [29] Richard Price. Academia.edu, 2008. <https://www.academia.edu/>.
- [30] LangChain. Quickstart. [https://python.langchain.com/docs/get\\_started/quickstart](https://python.langchain.com/docs/get_started/quickstart).
- [31] HuggingFace. Modelo llama-2 7b-chat. <https://huggingface.co/daryl149/llama-2-7b-chat-hf>.

- [32] CEATIC. Clúster ada. <https://www.ujaen.es/centros/ceatic/servicios/supercomputacion/cluster-ada>.