Universidades de Burgos, León y Valladolid

Máster universitario

Inteligencia de Negocio y Big Data en Entornos Seguros

TFM del Máster Inteligencia de Negocio y Big Data en Entornos Seguros

Creación de una IA optimizada para la asignatura de Fundamentos de Programación

Presentado por Víctor Hernando Aragón
en Universidad de Burgos — 24 de diciembre de 2025
Tutor: Carlos E. Vivaracho Pascual

Universidades de Burgos, León y Valladolid

Máster universitario en Inteligencia de Negocio y Big Data en Entornos Seguros

D. Carlos E. Vivaracho Pascual, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. Víctor Hernando Aragón, con DNI 71233566J, ha realizado el Trabajo final de Máster en Inteligencia de Negocio y Big Data en Entornos Seguros titulado Creación de una IA optimizada para la asignatura de Fundamentos de Programación.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 24 de diciembre de 2025

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. nombre tutor	D. nombre co-tutor

Resumen

Este proyecto abarca la necesidad de la creación de un modelo de inteligencia artificial open source el cual este adaptado a las necesidades de la asignatura de Fundamentos de Programación impartida en el grado de Ingeniería informática de la universidad de Valladolid a través de un re entrenamiento con el código en Java y normas impartidas en la asignatura.

Este proyecto permitirá a los alumnos emplear un modelo de IA que este adaptado al contexto de la asignatura que están cursando, no como los modelos que normalmente suelen ser consultados, de esta manera obtendrán respuestas y resolverán las dudas adaptadas a los contenidos de la asignatura.

El proyecto será desarrollado en el lenguaje de programación de Python y con el hardware de la escuela de ingeniería informática de Valladolid.

Descriptores

Inteligencia artificial, Fundamentos de Programación, Java, LLM.

Abstract

This project covers the need for the creation of an open source artificial intelligence model which is adapted to the needs of the Fundamentals of Programming course taught in the degree of Computer Engineering at the University of Valladolid through a retraining with the Java code and standards taught in the course.

This project will allow students to use an AI model that is adapted to the context of the subject they are taking, not like the models that are usually consulted, so they will get answers and solve the doubts adapted to the contents of the subject.

The project will be developed in the Python programming language and with the hardware of the school of computer engineering of Valladolid.

Keywords

Artificial intelligence, Fundamentals of Programming,Java, LLM

Índice general

Índice de figuras

Índice de tablas

Memoria

Introducción

1.1. Contexto

Los avances de los últimos años en Inteligencia Artificial generativa y LLM's han llevado a la creación de aplicaciones de ámbito general (e.g., ChatGPT, CoPilot, DeepSeek) que actualmente son utilizadas por la mayoría de estudiantes para resolver dudas y ampliar los conocimientos de sus asignaturas.

Sin embargo, sus modelos carecen del contexto del alumno o de los conocimientos que son impartidos en las asignaturas que se usan, ya que han sido entrenados a través de documentación genérica de internet. Así, los resultados que ofrece para preguntas relacionadas con algunas asignaturas no están contextualizadas, no siempre son precisas y/o no son correctas, resultando en un aprendizaje negativo para los estudiantes.

Fundamentos de Programación es una asignatura del Grado de Ingeniería Informática que se encuentra en el primer cuatrimestre del primer año de carrera, donde típicamente sucede este problema, en el que personas que pueden haber o no empezado en el mundo de la programación por lo que es esencial aprender los conceptos correctamente.

La asignatura hace especial hincapié en las buenas prácticas de programación utilizando el lenguaje de programación Java y el paradigma de programación estructurada. Sin embargo, cuando se pregunta a estas aplicaciones sobre determinados ejercicios de programación, sus respuestas no están contextualizadas al paradigma utilizado. Por esta razón, se quiere adaptar un modelo de IA que sea capaz de indicar a los alumnos los errores que tienen en sus programas y plantear alternativas correctas a los mismos. No se busca que el sistema, como cualquiera de los actuales, dé una solución

a un problema, sino que la solución esté contextualizada. Si no es correcta, le puede indicar dónde y por qué y/o proponer soluciones correctas.

1.2. Objetivos

Como se ha comentado en la introducción, el principal objetivo de este proyecto es la creación de un modelo basado en una IA openSource que permita a los alumnos enviar preguntas de texto así como código en el lenguaje de programación Java principalmente para comprobar si el código introducido es correcto según el contexto de la asignatura de Fundamentos de Programación impartida en el grado de Ingeniería Informática de la Universidad de Valladolid en el primer cuatrimestre del primer año de carrera. Con ella los alumnos serán capaces de obtener los resultados esperados tanto a nivel de programación como aprendizaje de los alumnos, asentando los conceptos básicos impartidos en esta asignatura para que a lo largo de la carrera se irán desarrollando.

A parte de este que es el objetivo principal para llegar a el deberemos de cumplir los siguientes objetivos:

- Realizar un análisis inicial de los principales modelos de I.A open source, teniendo en cuenta diferentes aspectos como tamaño del lenguaje, que permita la generación de código, ademas de que permita entender el lenguaje natural en el idioma español.
- Contrastar las diferentes formas que existen en la actualidad para adaptar el modelo a la tarea a la funcionalidad que irá destinado.
- Hacer un análisis de los recursos hardware que serán necesarios para la adaptación del modelo.
- Investigar e implementar las tecnologías necesarias para llevar a cabo la adaptación del modelo.
- Realizar diferentes pruebas para ver si el rendimiento del proyecto es correcto y las respuestas ofrecidas satisfacen las necesidades del proyecto
- Pensar en las posibles líneas futuras para posibles futuros avances en el proyecto.

1.3. Nicho de mercado

Al tratarse de un proyecto totalmente personalizado para los requisitos de un cliente concreto, que en este caso es mi tutor, no ha sido necesario realizar un estudio de mercado.

En cuanto a los usuarios objetivo, la intención del cliente es que el simulador sea empleado por los alumnos de la Universidad de Valladolid en la Escuela de Ingeniería Informática en la asignatura de Fundamentos de Programación, asignatura obligatoria del 1º curso.

1.4. Estructura de la memoria

Este documento se estructura de la siguiente forma:

Capítulo 1 Introducción: En este capítulo se describe el contexto y motivación a través de los cuales el proyecto empezó a realizarse, así como los objetivos principales que debe de cumplir el proyecto y el nicho de mercado al que va destinado.

Capítulo 2 Metodología y planificación: En este capítulo se describe la metodología usada durante el desarrollo del proyecto, el análisis y plan de riesgos, asi como un análisis de costes y seguimiento del proyecto

Capítulo 3 Conceptos teóricos : En este capítulo se describen los principales conceptos que han sido necesarios para llevar a cabo la realización de este proyecto y su entendimiento.

Capítulo 4 Alternativas de diferentes IA´s : En este capítulo se describe el estudio que se realizó de diferentes modelos de inteligencia artificial con el fin de que cumplieran con el propósito del proyecto y sus objetivos.

Capítulo 5 Formas de adaptación del modelo : En este capítulo se describen las diferentes formas de adaptar el modelo base seleccionado en el capítulo anterior al contexto del proyecto, diferenciando entre las múltiples alternativas que existen así como un listado de las ventajas y desventajas de usar cada opción.

Capítulo 6 Selección del modelo y proceso de adaptación : En este capítulo se habla del modelo de inteligencia artificial seleccionado y

el tipo de adaptación seleccionada para reentrenar el modelo para el contexto del proyecto.

Capítulo 7 Dataset de reentrenamiento y frontal : En este capítulo se habla del conjunto de datos empleado para reentrenar el modelo, que tipo de formato se ha seguido así como ejemplos del mismo, ademas se añade la tecnología empleada para la creación de la interfaz a través de la que los usuarios harán preguntas al modelo.

Capítulo 8 Tecnologías utilizadas : En este capítulo se habla de las tecnologías software empleadas para la realización del proyecto.

Capítulo 9 Conclusiones y líneas de trabajo futuras : En este capítulo se habla de las conclusiones tras la realización del proyecto, las posibles mejoras a realizar y trabajos a futuro.

Metodología y planificación

2.1. Metodología en cascada

Los orígenes de esta metodología [14] vienen del teórico de la informática Winston W. Royce, quien en 1970 elaboró su ensayo *Managing the Development of Large Software Systems*, donde proponía que el modelo en cascada se efectuara de manera iterativa. Mas tarde, a partir de 1985, este modelo se hizo famoso en el desarrollo del software cuando el Departamento de Defensa de los Estados Unidos publicó el Estándar 2167 para el desarrollo de software militar, el cual era una variante de su modelo, denominado cascada rígida.

A la hora de emplear la metodología en cascada[12] tenemos que tener en cuenta sus ventajas y desventajas. Sus principales ventajas son que tiene una estructura clara y permite transmitir bien la información al cliente de en qué paso del desarrollo nos encontramos exactamente. Además, la estructura en cascada también permite la realización de las fases en paralelo o el uso de prototipos, según el tipo de proyecto. En cuanto a sus desventajas, las principales son que dificulta la modificación de requisitos durante el desarrollo, así como que la realización de pruebas se debe realizar al final del desarrollo. Sin embargo, este último punto en nuestro caso tiene poca importancia, ya que este proyecto requiere que su implementación sea al menos parcialmente funcional para poder ser probado.

Esta metodología se empleo en la primera fase del proyecto en la cual se realizó una investigación de que modelos de IA serían los adecuados para la tarea, que tipo de entrenamiento se realizaría asi como se realizaría ese entrenamiento, todas estas fases se han realizado de manera secuencial de

tal manera que cuando tenía una de ellas se pasaba a la siguiente, siguiendo el modelo en cascada.

El principal motivo de elección de este modelo de gestión del proyecto es debido a la índole de investigación que ha requerido para desarrollar las primeras fases del proyecto y poder tener el flujo completo funcionando. Lo cual ha requerido no solo investigación por mi parte sino contacto con la Universidad de Valladolid para el uso de recursos que permitan el entrenamiento de un modelo de IA de una manera optima.

Fases del Proyecto

Una vez que hemos decido emplear la metodología en cascada distinguiremos las siguientes fases a lo largo del desarrollo del proyecto:

- Documentación: en esta fase se ha dedicado a comprender como poder realizar un reentrenamiento de un modelo de IA, diferentes formas de entrenamiento asi como los componentes de los propios LLM, asentando los conceptos necesarios para realizarlo.
- Investigación: tras entender la parte teórica, se ha procedido a realizar una investigación de aquellos modelos de IA open source que pueden servir para la realización del proyecto, asi como las herramientas que nos permitirán realizar la tarea de rentrenamiento y los recursos harware como software necesarios para la tarea.
- Análisis: tras obtener varios posibles candidatos entre los modelos de IA se ha realizado un estudio en profundidad de los mismos con el fin de obtener el modelo que mejor se ajusta a la tarea de este proyecto
- Diseño, implementación: tras obtener el modelo adecuado y las herramientas para poder realizar el entrenamiento se ha realizado su implementación. Este punto ha sido en concreto donde se ha aplicado una metodología de ensayo error. En el cual se han realizado un ciclo de prueba, evaluación y ajuste hasta obtener el resultado que queremos del modelo.
- Pruebas de rendimiento del modelo: tras realizar su reentrenamiento se ha pasado el modelo a través de diferentes métricas y evaluaciones con el fin de ver si el modelo reentrenado cumple con las necesidades del proyecto, en el caso de no ser así se han realizado tareas de ajuste sobre los parámetros para obtener la configuración mas optima del mismo

- Finalización de la documentación del proyecto: se han realizado las últimas modificaciones al documento con el fin de la mejora de esta memoria de cara a su presentación.

2.2. Planificación inicial del proyecto

Empleando el modelo en cascada visto en el apartado anterior, se elaboró al inicio del proyecto una planificación en la cual abarcamos sus distintas fases.

2.3. Diagrama de Gantt inicial del proyecto

En la Figura 2.1 podemos apreciar la duración prevista de cada una de las etapas del proyecto. En este diagrama aparecen las tareas principales especificadas anteriormente en la Tabla 2.1.

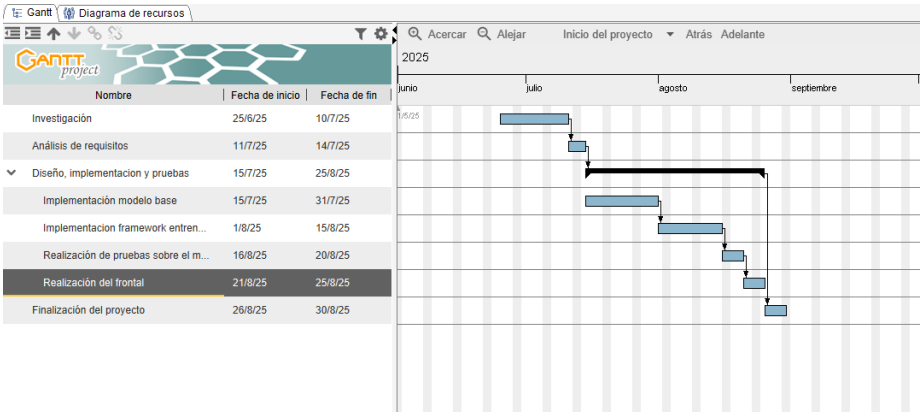

Figura 2.1: Foto del diagrama de gant previsto al iniciar el proyecto[8]

2.4. Planificación inicial del proyecto

Empleando el modelo en cascada visto en el apartado anterior, se elaboró al inicio del proyecto una planificación en la cual abarcamos sus distintas fases. Abarcando las 225 horas que aproximadamente se corresponden con los 9 créditos designados al proyecto.

La Tabla 2.1 muestra las fechas establecidas para cada fase del TFG.

Tarea	Duración	Comienzo	Fin
INVESTIGACIÓN			
Investigación	100 horas	25/06/25	10/07/25
ANALISIS			
Análisis de los requisitos del proyecto	10 horas	11/07/25	14/07/25
DISEÑO, IMPLEMENTACIÓN Y PRUEBAS			
Implementación del modelo base	20 horas	15/07/2025	31/07/25
Implementación del framework para entrenar el modelo	20 horas	01/08/2025	15/08/2
Realización de pruebas sobre el modelo reentrenado	30 horas	16/08/2025	20/08/25
Realización del frontal	20 horas	21/08/2025	25/08/25
FINALIZACIÓN DEL PROYECTO			
Finalización de la documentación y entrega del TFM	25 horas	26/08/25	30/08/25

Tabla 2.1: Planificación inicial del proyecto

2.5. Plan de Riesgos

Para poder realizar un proyecto de cualquier índole es necesario elaborar un plan de riesgos. Estos riesgos deben de referirse a problemas futuros, los cuales podrían suceder durante el desarrollo del proyecto, y debe documentarse tanto la causa como el efecto que producirá el riesgo.

Para elaborar de manera correcta el plan de riesgos se han tenido en cuenta los siguientes puntos:

- La probabilidad de que se produzca el riesgo. Será medida a través de los valores: bajo, medio y alto.
- El impacto que tendrá el riesgo. Si llega a aparecer será medido a través de los valores: bajo, medio y alto.
- Plan de mitigación: conjunto de acciones que se deben realizar para reducir la probabilidad de que un riesgo ocurra.

- Plan de contingencia: conjunto de acciones que se deben realizar una vez el riesgo se ha ocurrido para reducir su impacto sobre el proyecto.

La Tabla 2.2 muestra el Riesgo 1, el cual es la estimación incorrecta del tiempo en el desarrollo del proyecto.

R1	**Estimación incorrecta del tiempo**
Descripción	El desarrollo del proyecto no se está llevando según las estimaciones iniciales de tiempo.
Probabilidad	Baja
Impacto	Medio
Plan de mitigación	▪ Realizar una nueva estimación, para determinar el alcance real del proyecto. ▪ Priorizar las tareas críticas.
Plan de contingencia	▪ Definir el alcance del proyecto, acorde al tiempo disponible.

Tabla 2.2: Riesgo 1

La Tabla 2.3 muestra el Riesgo 2, el cual es la perdida de tiempo de trabajo por ausencia del alumno, mientras que la Tabla 2.4 muestra el Riesgo 3, el cual representa el riesgo de ausencia del tutor.

R2	**Ausencia del alumno por motivos de trabajo u enfermedad**
Descripción	El estudiante no puede realizar el proyecto debido a otras responsabilidades laborales, familiares u enfermedad del mismo
Probabilidad	Alta
Impacto	Medio
Plan de mitigación	▪ Comunicar a los tutores el posible retraso
Plan de contingencia	▪ En función del tiempo perdido, redistribuir las horas o añadir un margen de tiempo a la planificación.

Tabla 2.3: Riesgo 2

R3	**Ausencia de los tutores por motivos de trabajo u enfermedad**
Descripción	El estudiante no puede realizar avances el proyecto por dependencia de contacto con los tutores del proyecto.
Probabilidad	Media
Impacto	Medio
Plan de mitigación	▪ Realizar una comunicación vía email sobre la situación personal del tutor para que el alumno tenga constancia de ello
Plan de contingencia	▪ Cancelar o aplazar las reuniones que se iban a tener con el alumno sobre el avance del proyecto. ▪ Buscar un medio de comunicación alternativo, tales como reuniones por videoconferencia, para realizar las tutorías en el caso de que el tutor lo considere necesario.

Tabla 2.4: Riesgo 3

La Tabla 2.5 muestra el Riesgo 4, el cual es la modificación de requisitos.

R4	**Cambio en los requisitos del proyecto**
Descripción	Se produce una modificación de los requisitos acordados al inicio del proyecto.
Probabilidad	Baja
Impacto	Alto
Plan de mitigación	▪ Mantener informado al tutor del proyecto de problemas que surjan y el avance del proyecto de manera recurrente.
Plan de contingencia	▪ Añadir un margen a la planificación del proyecto.

Tabla 2.5: Riesgo 4

La Tabla 2.6 muestra el Riesgo 5, el cual es la avería del ordenador de trabajo o caída de los servicios utilizados.

2.6. Planificación final del proyecto

Tras haber pasado por todas las fases descritas en la planificación inicial como se puede apreciar en la Tabla 2.1, algunos de los riesgos anteriormente mencionados se han producido, concretamente:

- El estudiante Víctor Hernando Aragón, a la vez que realiza este proyecto realiza un trabajo a tiempo completo por lo que a lo largo de la duración del mismo esta activo el riesgo 2 (Tabla 2.3).
- El 11 de julio se produjo un conflicto en las librerías empleadas en el proyecto en la máquina virtual empleada para el desarrollo del mismo lo que activo el el riesgo número 5 (Tabla 2.6).
- El 19 de septiembre se produjo una caida de la máquina virtual empleada para el desarrollo del proyecto que activo el el riesgo número 5 (Tabla 2.6).

R5	**Fallos en las aplicaciones o hardware para desarrollar el proyecto**
Descripción	Se produce algún problema con el hardware o con los servicios software empleados.
Probabilidad	Media
Impacto	Alto
Plan de mitigación	▪ Mantener los repositorios donde se almacena el proyecto lo más actualizados posible. ▪ Tener un copia local del proyecto en nuestro ordenador.
Plan de contingencia	▪ En el caso del ordenador se nos estropeara tendríamos que intentar emplear otro, si es posible. ▪ Si alguno de los servicios software empleados dejara de funcionar, lo ideal sería hacer los cambios en local y esperar a que el servicio vuelva a estar disponible.

Tabla 2.6: Riesgo 5

- El 20 de noviembre se produjo una falta de almacenamiento en la máquina virtual, que impedía realizar diferentes pruebas lo que activo el riesgo número 5 (Tabla 2.6).

- A finales de noviembre, los tutores Alma María Pisabarro Marron y Carlos Enrique Vivaracho Pascual no pudieron reunirse en una de las reuniones semanales sobre el avance del proyecto, esto activaría el riesgo número 3 (Tabla 2.4).

- El 30 de Noviembre se produjo una caída de la máquina virtual empleada para el desarrollo del proyecto que activo el el riesgo número 5 (Tabla 2.6).

A continuación en la Tabla 2.7 muestra las fechas establecidas para cada fase del tfm después de la aparición de los riesgos y como afectaron la aplicación de los planes de contingencia al desarrollo del proyecto.

Tarea	Duración	Comienzo	Fin
INVESTIGACIÓN			
Investigación	120 horas	25/06/25	14/08/25
ANALISIS			
Análisis de los requisitos del proyecto	20 horas	15/08/25	20/08/25
DISEÑO, IMPLEMENTACIÓN Y PRUEBAS			
Implementación del modelo base	20 horas	21/08/2025	01/09/25
Implementación del framework para entrenar el modelo	37 horas	02/09/2025	11/10/25
Realización de pruebas sobre el modelo reentrenado	62 horas	12/10/25	08/12/25
Realización del frontal	20 horas	09/12/25	15/12/25
FINALIZACIÓN DEL PROYECTO			
Finalización de la documentación y entrega del TFM	35 horas	16/12/25	ESCRIBIR FECHA

Tabla 2.7: Planificación inicial del proyecto

Como podemos apreciar de las 225 horas planeadas al inicio del proyecto se han elevado a un total de 314 horas las fases con mayor incremento han sido la de investigación y de pruebas del modelo esto se debe principalmente a los siguientes motivos:

- La fase de investigación se alargó mas de lo esperado debido a que fue difícil encontrar un framework que permitiera realizar las tareas de re entrenamiento de modelos de manera sencilla realizando múltiples investigaciones de diferentes proyectos y alternativas tanto de implementación solo con librerías básicas o frameworks hasta dar con el correcto. El cual se emplearía en la implementación final.
- La fase de realización de pruebas sobre el modelo reentrenado se ha alargado debido a que ha resultado difícil conseguir un buen resultado

del modelo reentrenado, debido al problema de la falta de datos para el entrenamiento así como que el modelo inicial no fue el adecuado a la hora de hacer la inferencia.

- A todo esto se suma la realización del proyecto junto con el trabajo del alumno a jornada completa lo que no ha permitido ser del todo consistente con los tiempos estipulados al inicio del proyecto.

La Figura **??** muestra el diagrama de Gantt final del proyecto, donde se pueden apreciar la duración de cada una de las etapas una vez finalizado el proyecto. En el diagrama se pueden observar las tareas principales del proyecto.

AÑADIR GANTT FINAL

2.7. Plan de Presupuesto

En todos los proyectos existe un presupuesto, en el cual se muestran los gastos del proyecto, teniendo en cuenta las herramientas hardware y software utilizadas, así como el personal que ha trabajado en este proyecto.

Presupuesto Software

El proyecto se ha realizado en su mayoría con herramientas software gratuitas.

En el caso de que el proyecto se estuviera desarrollando en una empresa esta debía de asumir el coste empresarial de unsloth herramienta empleada para almacenar el modelo, su coste supone un total de 50 € al mes que durando el proyecto alrededor de 7 meses supondría un coste total de 350 € a fecha de 10/12/2025

Presupuesto Hardware

El hardware son los dispositivos que han sido empleados para la realización del proyecto, los cuales son:

- Maquina virtual linux: 70 € mes 490 € en total
- Monitor Acer: tiene un precio de 150 €
- Tarjeta grafica NVIDIA A40: 7104,52 €

Presupuesto recursos humanos e infraestructura

En cuanto al coste de personal, considerando al alumno un desarrollador Mid-Serior de tres años, el sueldo oscila alrededor de los 33.000 € anuales en contratos a jornada completa, es decir, 40 horas a la semana, o unas 160 horas al mes, aplicando un IRPF del 17,11 % sobre el salario bruto. El sueldo mensual equivalente serían 1.800,94 € en 14 pagas netos al mes. Como tal el gasto a la empresa por horas partiendo del total bruto en 7 meses son 19.250 €

Sobre el presupuesto de infraestructura, teniendo en cuenta que el trabajo se ha realizado en la vivienda personal del alumno, podemos considerar el coste de internet como gasto. Teniendo en cuenta que la tarifa son 44 € al mes y se ha llevado a cabo el proyecto durante 7 meses el coste total es de 308 €

Presupuesto total

En la Tabla 2.8 procedemos a sumar las cantidades de cada uno de los apartados anteriores para obtener el total.

Concepto	**Coste**
Presupuesto software	350 €
Presupuesto hardware	7.744,22 €
Presupuesto recursos humanos e infraestructura	19.558 €
Coste total	**27.652,22 €**

Tabla 2.8: Coste total del proyecto

Conceptos teóricos

En este capítulo nos centraremos especialmente en la terminología y conceptos teóricos necesarios para el entendimiento del proyecto, todos los conceptos explicados a continuación han tenido que ser investigados, estudiados y comprendidos por el alumno, de tal manera de poder llegar así a la solución planteada.

3.1. Terminología empleada

- Inteligencia Artificial (IA): Según la comisión europea son sistemas de software (y posiblemente también de hardware) diseñados por humanos que, ante un objetivo complejo, actúan en la dimensión física o digital percibiendo su entorno, a través de la adquisición e interpretación de datos estructurados o no estructurados y razonando sobre el conocimiento, procesando la información derivada de estos datos y decidiendo las mejores acciones para lograr el objetivo dado. [6]

- Modelo extenso de lenguaje (LLM): se trata de modelos que son entrenados con grandes cantidades de texto proveniente de artículos, dentro de estos podemos diferenciar entre los LLM tradicionales, aquellos que han sido entrenados unicamente con texto como por ejemplo GPT-3, LLaMA y los LLM Multimodales que han sido entrenados con otras fuentes de información que no es solo texto, como imágenes, video y audio.

- Parámetros del modelo: matrices numéricas que ajustan cómo el modelo transforma los datos de entrada en salida, a mayor numero de

parámetros mayor capacidad de aprendizaje y compresión de patrones, pero también supone un mayor consumo de memoria y coste computacional

- Tokens: Los tokens son palabras, juegos de caracteres o combinaciones de palabras y puntuación que generan los modelos de lenguaje grandes (LLM) cuando descomponen el texto. La tokenización es el primer paso del entrenamiento.

- Dataset de entrenamiento: conjunto de datos con cierta estructura que es capaz de entender el modelo con el fin de reentrenarlo para la tarea especifica de los datos. .

3.2. Formas de adaptación de un modelo al contexto

Este apartado pretende recoger las diferentes formas de conseguir que el modelo seleccionado emplee el conjunto de datos basado en un conjunto de programas java. Existen diferentes formas de entrenar a nuestro modelo entre las cuales podemos destacar son RAG (Retrieval-Augmented Generation), LoRA/QLoRA (Low-Rank Adaptation/ Quantum Low-Rank Adaptation) y el fine-tuning completo (ajuste fino entrenando todos los parámetros del modelo). A continuación, para cada técnica se explica como funciona, las herramientas típicas para implementarla, requisitos de hardware, ventajas, desventajas y escenarios recomendados de uso.

RAG (Retrieval-Augmented Generation)

RAG (Generación Aumentada con Recuperación de información) es un método que no altera los parámetros del modelo base, sino que lo potencia proporcionándole información adicional relevante extraída de una base de conocimiento externa en cada consulta. RAG extiende las capacidades del LLM hacia un dominio específico o base de conocimiento interna sin necesidad de reentrenar el modelo. Es una forma rentable de mantener las respuestas del modelo actualizadas, precisas y enfocadas al contexto deseado, porque aprovecha información fuera de los datos originales de entrenamiento del modelo.[2]

Funcionamiento

Con RAG, el proceso de inferencia del modelo se complementa con un buscador que permite recuperar la información actuando antes de la generación de la respuesta. Cuando el usuario realiza una consulta o prompt, este componente de búsqueda utiliza la entrada para recuperar datos relevantes de una fuente externa (por ejemplo, un conjunto de documentos, una base de código o una base de conocimientos)[2].

Los datos externos (textos, fragmentos de código, documentación, etc.) suelen haber sido previamente procesados con modelos de embeddings para convertirlos en representaciones numéricas, almacenándose en una base de datos vectorial que permite búsquedas semánticas eficientes[2].

Una vez obtenidos los documentos o fragmentos más relevantes mediante la búsqueda (por similitud vectorial, coincidencia semántica, etc.), se aumenta la entrada original del usuario añadiendo ese contenido recuperado al prompt. Finalmente, el modelo de lenguaje genera su respuesta utilizando tanto su conocimiento entrenado como la información específica proporcionada en el prompt [2].

En esencia, RAG inyecta conocimiento puntual de la base externa en la entrada del modelo para guiar la generación de una respuesta más informada. Diagrama conceptual de RAG: el modelo de lenguaje (LLM) recibe la consulta del usuario junto con datos relevantes recuperados de una base de conocimiento externa, utilizando ambos para generar una respuesta precisa. El flujo típico es:

1. Convertir la pregunta del usuario en una consulta semántica
2. Buscar en la base de conocimiento vectorial los documentos o fragmentos más relacionados
3. Añadir esos fragmentos al prompt antes de la generación

De este modo, no se modifica el modelo original, sino que se le proporciona contexto adicional en tiempo de inferencia.

Requisitos Hardware

En comparación de los otros metodos de adaptación de modelos que veremos mas adelante se trata realmente de un costo computacional bajo, ya que como hemos visto realmente no realizamos un entrenamiento del

modelo sino que realizamos la búsqueda del prompt inicialmente en la base de conocimiento, por lo que el hardware principalmente será necesario para almacenar el modelo original y mantener la estructura de busqueda sobre la base de conocimiento y puede ejecutarse en un entorno doméstico siempre que el modelo por ejemplo no tenga muchos parámetros

Ventajas

- No altera el modelo base: No hay que reentrenar el LLM, lo que evita costos de cómputo elevados y riesgos de degradar el modelo. Se aprovecha el conocimiento general del modelo y sólo se le inyecta información específica cuando hace falta.

- Información actualizada y específica: Permite al modelo acceder a datos más recientes o especializados que no estaban en su entrenamiento original, así como modificar la base de conocimiento para mantener actualizadas las respuestas que de el LLM.

- Control y trazabilidad: podemos controlar la fuente de la información que el modelo usa, imitándola a bases de conocimiento específicas.

Desventajas

- Dependencia del motor de búsqueda: La calidad de las respuestas depende en gran medida de qué tan bien funcione el módulo de búsqueda. Si la base de datos vectorial no devuelve los fragmentos relevantes o está incompleta, el modelo puede seguir alucinando o dando información incorrecta. Requiere por tanto un buen diseño de la base de conocimiento y afinamiento del proceso de búsqueda.

- Complejidad del sistema: Comparado con usar sólo un LLM fine-tuneado, RAG introduce componentes adicionales, lo cual puede ser más complejo de implementar y mantener. Hay más partes móviles que pueden fallar o añadir latencia (por ejemplo, la búsqueda vectorial añade pasos extra en cada consulta).

- No estamos entrenando el modelo: realmente no estamos adaptando el modelo a nuestras necesidades, simplemente el prompt que nos llega por parte del usuario lo consultamos en la base de conocimiento y este luego se pasa al modelo.

Casos de uso

- Información dinámica o actualizable: Si el conocimiento del dominio cambia con frecuencia (nuevas versiones de software, noticias al día, datos en tiempo real), RAG permite mantener las respuestas actualizadas simplemente actualizando la base de conocimiento, sin tener que re-entrenar el modelo constantemente.
- Recursos de cómputo limitados: Cuando no se cuenta con GPUs potentes ni tiempo para entrenamiento, RAG ofrece una forma de adaptar un modelo grande a un contexto específico aprovechando la inferencia solamente. Por ejemplo, empresas que quieran personalizar un chatbot con sus documentos internos, pero no puedan costear entrenar de nuevo un LLM, pueden usar RAG para lograrlo de forma eficiente.
- Cuando el modelo necesita contexto para un caso de uso específico: si ya empleamos de base un buen modelo para la tarea concreta, como por ejemplo un modelo el cual esta centrado en código y le damos una base de conocimiento mas concreta entonces obtendremos unas buenas respuestas y encima adaptadas al contexto específico.

LoRA / QLoRA (Low-Rank Adaptation)

LoRA (Low-Rank Adaptation) es una técnica de ajuste fino eficiente en parámetros que permite adaptar grandes modelos con un costo computacional muy bajo en comparación con el fine-tuning tradicional. La idea central es congelar los pesos originales del modelo y añadir un pequeño conjunto de parámetros entrenables adicionales que aprenden la tarea o el dominio nuevos. Estos parámetros adicionales toman la forma de adaptadores de rango bajo (low-rank), de ahí el nombre LoRA. Durante el entrenamiento, únicamente se ajustan estos adaptadores (que representan un porcentaje muy pequeño del total de parámetros), manteniendo intactos los pesos originales. Al término, el modelo utiliza los pesos originales más las pequeñas modificaciones aprendidas para generar respuestas especializadas. QLoRA es una variante de LoRA que combina esta idea con cuantización a 4 bits del modelo base para minimizar aún más el uso de memoria. En QLoRA, el modelo preentrenado se carga en formato cuantizado de 4 bits (en lugar de 16 o 32 bits típicos), se mantiene congelado, y se entrenan igualmente adaptadores LoRA encima. Esto permite afinar modelos extraordinariamente grandes usando hardware relativamente accesible, sin pérdida significativa de calidad.[5]

Funcionamiento

En LoRA estándar, las matrices de pesos de determinadas capas del modelo se complementan con unas matrices de bajo rango que se inicializan de forma que inicialmente no afecten la salida. Durante el entrenamiento, en lugar de actualizar la gran matriz de pesos original W (que permanece congelada), se entrenan dos matrices mucho más pequeñas, A y B, cuya multiplicación produce una aproximación de bajo rango de las posibles actualizaciones a W. En concreto, W se mantiene fijo y la actualización efectiva es $W + \Delta W$, donde $\Delta W = A \times B$ con A de dimensión (n × r) y B de (r × m) siendo r el rango bajo elegido (mucho menor que n o m). De esta forma, el número de parámetros ajustables es minúsculo comparado con W (aproximadamente $(n + m)r$, en lugar de nm). Al finalizar el entrenamiento, el modelo puede usar $W + \Delta W$ como pesos efectivos.[5]

Para resumir, esto significa que sólo se aprenden las diferencias necesarias para la nueva tarea o dominio, sin alterar los conocimientos originales del modelo. En QLoRA, se agrega un paso previo: cuantizar el modelo base a 4 bits.[13] El modelo preentrenado en 16 bits (FP16) se convierte a un formato de 4 bits (normalmente NF4, Normalized Float 4, óptimo para pesos distribuidos normalmente). Luego, ese modelo 4-bit se congela y se incorporan los adaptadores LoRA como en el método original.

Durante el entrenamiento, se propagan los gradientes a través del modelo cuantizado congelado hacia los adaptadores, actualizando sólo los pesos de LoRA. La cuantización reduce drásticamente la memoria necesaria para almacenar el modelo, y al no actualizar los pesos cuantizados, se evita degradar su valor original con operaciones de entrenamiento. Para el cálculo de gradientes, QLoRA de hecho decuantiza temporalmente los pesos a 16 bits en el pase hacia atrás, pero sólo calcula gradientes para los parámetros LoRA[13]

En suma, QLoRA conserva las ventajas de LoRA (pocos parámetros a entrenar) sumándole una compresión del modelo base, logrando una eficiencia excepcional sin prácticamente sacrificar rendimiento (según estudios, QLoRA alcanza resultados a la par del fine-tuning completo en 16 bits en muchos casos).[13]

Requisitos Hardware

LoRA y especialmente QLoRA se diseñaron para reducir radicalmente los requerimientos de memoria y cómputo al adaptar grandes modelos. En la práctica, esto significa que se puede fine-tunear un modelo grande en

una sola GPU de gama prosumidor o incluso en GPU de menor VRAM. Por ejemplo, se ha reportado que full fine-tuning de un modelo de 7B parámetros en FP16 requeriría del orden de 40–60 GB de VRAM, mientras que con LoRA ese mismo modelo puede entrenarse con 8 GB de VRAM, haciendo posible su uso en entornos domésticos.[4]

Ventajas

- Extremadamente eficiente en recursos: La principal ventaja es la drástica reducción en uso de VRAM y cómputo durante el entrenamiento. Sólo se ajusta una fracción ínfima de los parámetros del modelo, ademas de que obtiene resultados similares a un finetuning completo.
- Conserva el conocimiento original: Dado que el modelo base no se modifica, no hay riesgo de forgetting o de dañar las capacidades generales del LLM. El adaptador aprende sobre el dominio , pero el conocimiento original del modelo (sintaxis general, gramática, conocimientos del mundo) permanece intacto.
- Distribución y despliegue sencillos: Relacionado con lo anterior, compartir o desplegar un modelo adaptado con LoRA es más fácil, ya que sólo se comparten los pesos de los adaptadores (p. ej. vía HuggingFace Hub) que son archivos pequeños, y el destinatario los aplica al modelo base público. Esto facilita la colaboración y la iteración entre equipos, sin intercambiar enormes ficheros de modelo.

Desventajas

- Complejidad de integración en inferencia: Para usar un modelo LoRA, se necesita cargar tanto el modelo base como los pesos del adaptador añadiendo una capa extra de complegidad operacional a tener un modelo completamente ajustado.
- No entrena todo el potencial: Si bien LoRA adapta bien el modelo, no permite ajustar todos los matices ya que muchos parámetros quedan sin modificación, por lo que en conjuntos de datos grandes es mas recomendable realizar un entrenamiento completo de todos los parámetros, ya que LoRA busca cambios en un subespacio limitado.
- Limitaciones de QLoRA específicas: En QLoRA, al usar cuantización 4-bit, actualmente no es posible realizar entrenamiento 4-bit en CPU, debe ser en GPU con CUDA.

Casos de uso

- Cuando los recursos son limitados pero se necesita especialización: LoRA/QLoRA es la técnica de elección cuando no se dispone de múltiples GPUs de gran memoria o clusters para entrenamiento.
- Proyectos de investigación o prototipos rápidos: Si se desea experimentar con adaptar modelos a distintas tareas de forma ágil.
- Múltiples dominios o tareas en un mismo modelo base: Cuando se quiera mantener un solo modelo base y tener variaciones especializadas para distintos casos como un mismo LLM base adaptado por separado a código Java, a generación de documentación técnica.
- Cuando se requiere conservar el modelo original intacto: Si por algún motivo necesitamos que el modelo pueda volver a su estado original o mantener la opción de usarlo en tareas generales, LoRA es ideal porque no modifica los pesos base.

Fine-tuning completo del modelo

El fine-tuning completo es el enfoque tradicional en el que se entrenan todos los parámetros del modelo de lenguaje en el conjunto de datos específico, ajustando así completamente el modelo a la nueva tarea o dominio. En este caso no se introduce arquitectura adicional: simplemente se toma el modelo preentrenado existente y se continúa entrenando con los datos proporcionados (normalmente con una tasa de aprendizaje baja para no destruir lo ya aprendido). Al finalizar, todos los pesos del modelo pueden haberse actualizado para reflejar los patrones del nuevo conjunto de datos. Este método convierte esencialmente al modelo generalista en un modelo especializado aprendiendo directamente de los datos específicos. Las principales desventajas son que es costoso a nivel de recursos y tiene sus riesgos de cara al modelo final.

Funcionamiento

Es similar a cuando se entrena el modelo original pero partiendo de los pesos ya aprendidos para los parámetros en pre-entrenamiento. Se formula un objetivo y se usa el algoritmo de optimización para ajustar los gradientes de todos los parámetros en cada paso, minimizando la función de pérdida en el conjunto de datos específico. Durante este proceso, el modelo puede adaptar todos sus niveles de representación: desde capas inferiores (aprendiendo

algunas sintaxis específicas del código Java del conjunto de datos que estamos usando) hasta las superiores (quizás adoptando cierto estilo en las respuestas

Tras el fine-tuning, se obtiene un nuevo modelo (un nuevo conjunto completo de pesos) especializado en los datos proporcionados. En nuestro caso específico si afinamos completamente un modelo con un con un conjunto de datos de código Java, el resultado es un modelo cuyo conocimiento del lenguaje Java y estará embebido en todos sus parámetros.

Requisitos Hardware

El fine-tuning completo de LLMs es notoriamente intensivo en recursos. Se necesita suficiente VRAM (memoria de GPU) para alojar el modelo completo y sus gradientes y los estados del optimizador, además de las activaciones durante el pase hacia adelante. Esto se acumula rápidamente: por ejemplo, un modelo de 7 mil millones de parámetros ocupa unos 14 GB en FP16 solo para los pesos; el optimizador requiere de mas recursos, y las activaciones/gradientes de una pasada pueden requerir otros 10 GB, llevando el total a on the order of 40–60 GB de VRAM para entrenar un modelo de 7B[4]

Ventajas

- Modelo autónomo especializado: El resultado del fine-tuning completo es un modelo especializado independiente, que no necesita componentes adicionales ni contexto externo para desempeñarse en el dominio. Toda la información relevante del dataset queda embebida en sus pesos.

- Adaptación de todos los pesos de los parámetros: Al actualizar todos los parámetros, el modelo tiene máxima capacidad de absorber el nuevo conocimiento o adaptar sus habilidades. No está limitado por un subespacio de ajustes (como en LoRA). Esto puede traducirse en un rendimiento ligeramente superior en la tarea específica cuando se dispone de suficientes datos.

- Personalización profunda del comportamiento: Más allá de conocimiento actual, al fine-tunear completo es posible cambiar comportamientos intrínsecos del modelo.

Desventajas

- Coste computacional alto: es de los tres métodos el que mayor conste computacional tiene tanto en hardware como en tiempo
- Riesgo de sobreajuste y forgetting: Al actualizarlo todo, existe el riesgo de sobreajustar al conjunto de datos específico, especialmente si este no es muy grande. El modelo puede terminar "memorizando.[en] exceso patrones o hasta ejemplos enteros del dataset o incluso el modelo puede perder parte de su conocimiento general al sobrescribirlo con la nueva información.
- Necesidad de gran cantidad de datos de calidad: Para aprovechar un fine-tuning completo sin arruinar el modelo, se suele requerir un conjunto de datos amplio y bien adaptado al dominio.

Alternativas de diferentes IA´s

En este capitulo se centra principalmente en la investigación realizada para encontrar un modelo adecuado para la tarea del proyecto partiendo de un analisis inicial del problema hasta los posibles lenguajes candidatos.

4.1. Descripción del problema

Uno de los dos problemas a la hora de empezar a desarrollar la idea del TFM es encontrar el mejor modelo LLM que se adapte al caso de uso, teniendo en cuenta las características del problema:

Usuarios

Los usuarios que emplearán este modelo de IA con el fin de obtener respuesta a preguntas realizadas tanto en lenguaje natural como enviando código Java serán alumnos de 1º de carrera de la Universidad de Valladolid (UVA), por lo tanto, estamos hablando de personas que están empezando su etapa como ingenieros informáticos así como empezando con la programación, ya que existirán alumnos que, como fue mi caso, no han programado todavía antes de entrar a la carrera.

Asignatura

El modelo de IA está destinado concretamente para la asignatura de Fundamentos de Programación, que se imparte en el primer cuatrimestre del primer año de ingeniería informática. en la cual se enseñan las bases de la programación estructurada en el lenguaje de programación Java. Por lo que el modelo que tendremos que elegir será un modelo que esté preparado

o entrenado de base con un conjunto de datos de texto o información relacionada con los lenguajes de programación y específicamente con Java.

Tamaño del modelo

El modelo es necesario que no sea muy grande, ya que la tarea que desempeñará está bastante acotada por el contenido de la asignatura, ya que este modelo de IA que se escoja usará también datos obtenidos de programas Java facilitados por la Universidad de Valladolid con el fin de que las respuestas que devuelva el modelo estén basadas en los contenidos que se imparten en esta asignatura.

Pero si es importante que a pesar de que el tamaño del modelo este acotado se opte por un modelo el cual la principal fuente de datos con los que ha sido entrenado sea codigo, en conncreto interesaría sobretodo con código Java

Coste del modelo

Debido a la naturaleza del proyecto y su uso universitario, alternativas de pago como la API de ChatGPT, Claude, Sonet, Gemini, entre otros LLM's de pago que quedan descartados, por lo que se optará por buscar LLM's que sean open source y gratuitos ademas de que las licencias que emplean permitan su uso y explotación.

4.2. Búsqueda de modelos

Tras realizar una búsqueda con las necesidades del proyecto se han elegido los siguientes candidatos:

DeepSeek-Coder

Se trata de un modelo entrenado con un total de 2 billones de tokens, en el que de esos tokens alrededor del 87 % es código y el resto son tokens en lenguaje natural en los idiomas inglés y chino. Ademas de que se nos ofrece el modelo con diferentes parámetros de 1.3B a 33B asi como un modelo base que sobretodo sirve para el autorellenado de código y otro instruct el cual esta entrenado para responder y resolver preguntas en lenguaje natural. [7].

En el propio repositorio de github nos ofrece una comparación con el resto de modelos open-source en los diferentes lenguajes de programación de una manera muy gráfica:

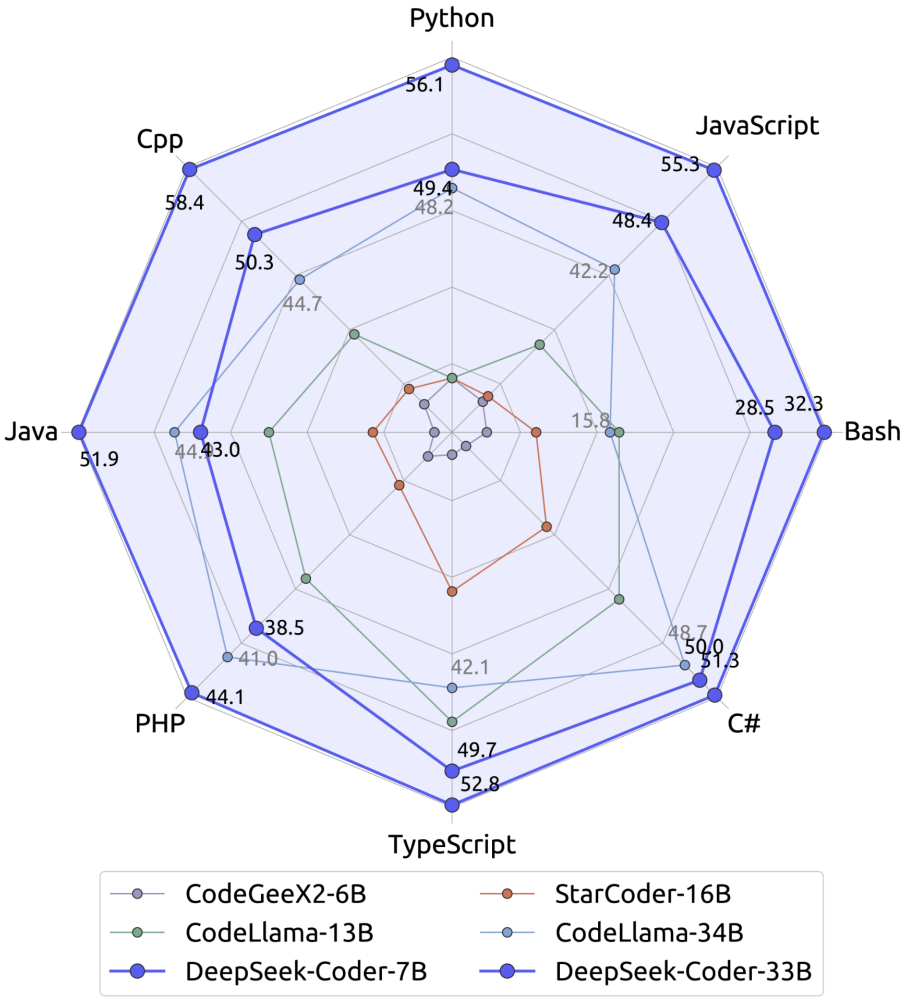

Figura 4.2: Grafico de comparación del rendimiento con respecto otros modelos open source basados en código [7]

Ventajas

- Es un modelo muy moderno, su primera versión es de noviembre de 2023.

- Como nos muestran en sus comparativas tanto el modelo con 7B y 33B parámetros tiene unos resultados muy buenos en todos los lenguajes superando a otros modelos como por ejemplo CodeLlama-34B en casi todos los lenguajes.

- Posee un modelo instruct que esta adaptado tanto para recibir código como preguntas en lenguaje natural.
- Emplea licencia MIT

Desventajas

- En los últimos meses se han realizado diferentes noticias sobre la seguridad del uso de DeepSeek con respecto a la protección de los datos sobretodo no introduciendo datos sensibles en ella ni personales.
- No dispone de tanta comunidad como otros modelos mas populares como CodeLlama o StarCoder

CodeLlama

Modelo perteneciente a meta derivado de Llama 2 ajustado para tareas de programación, posee ademas el modelo base que permite autocompletado, un modelo especializado en python y otro que permite recibir instrucciones

Ventajas

- Es un modelo muy moderno, su primera versión es de noviembre de 2023.
- Como nos muestran en sus comparativas tanto el modelo con 7B y 33B parámetros tiene unos resultados muy buenos en todos los lenguajes superando a otros modelos como por ejemplo CodeLlama-34B en casi todos los lenguajes.
- Posee un modelo instruct que esta adaptado tanto para recibir código como preguntas en lenguaje natural.

Desventajas

- Dispone de una licencia mas restrictiva sin ser MIT.

Llama 3

Modelo perteneciente a meta permite comprender y generar texto mejor que sus predecesores con un buen razonamiento en código y conversación.

Ventajas

- Es un modelo mas moderno, su fecha de lanzamiento fue el 18 de abril de 2024.
- Posee varieantes del modelo desde 8B a 70B de parametros
- Posee un modelo instruct que esta adaptado tanto para recibir código como preguntas en lenguaje natural.
- Al ser un lenguaje de meta y tener ya dos años en el mercado existen múltiples herramientas e implementaciones para el modelo que nos facilitan su implementación.
- Al no ser un modelo adaptado enteramente a código es capaz de entender mejor instrucciones en leguaje natural.

Desventajas

- Dispone de una licencia mas restrictiva sin ser MIT.
- No es un modelo especializado en código lo cual si se piden tareas de codigo complejas podría generar respuestas menos optimas que otros lenguajes.

StarCoder2

StarCoder es un modelo de lenguaje especializado en código fuente, desarrollado por el proyecto BigCode, en colaboración entre Hugging Face y ServiceNow Research. Es parte de la familia de modelos StarCoder / StarCoderBase, sucesores de SantaCoder, entrenados específicamente para tareas de desarrollo de software con múltiples lenguajes de programación y soporte para prompts en lenguaje natural.

Ventajas

- Es un modelo muy moderno, su primera versión es de noviembre de 2023.
- Como nos muestran en sus comparativas tanto el modelo con 15B y 3B parámetros tiene unos resultados buenos ademas de soportar multilenguaje.

- Los datos con los que ha sido entrenado pertencecen a repositorios de github preseleccionados.

Desventajas

- Dispone de una licencia mas restrictiva BigCode OpenRAIL-M v1.

CodeGeeX2

Modelo desarrollado por THUNLP (Tsinghua University NLP Group). Está diseñado para tareas de programación asistida por IA, con orientación multilingüe, tanto en lenguajes de programación como en lenguajes naturales.

Ventajas

- Tiene un tamaño de 6B de parametros lo que permite despliegues locales o en servidores con GPU´s estandar.
- entrenado con lenjuaes naturales incluido el español.
- Dispone de licencia MIT.
- Permite tareas de completado, explicación y depuración e código

Desventajas

- Modelo menos conocido que CodeLlama o StarCoder, si nos fiujamos en las descargas del modelo de HugginFace tiene 153 con respecto a los otros dos modelos que tienen 4.997 y 172.744 respectivamente
- No esta capacitado para autorrelleno de codigo como otros modelos.

Selección de modelo y proceso de adaptación

Para seleccionar el modelo y forma de adaptación sobre el modelo se han de realizar diferentes pruebas sobre los modelos así como ver que cantidad de datos tenemos para realizar la adaptación.

Pruebas realizadas sobre los modelos

Se han realizado diferentes pruebas en los modelos seleccionados; para ello, se ha empleado Ollama [16].

Ollama es una herramienta de código que permite ejecutar los modelos en un entorno local y además permite su uso en entornos con pocos recursos, ya que estos modelos que hemos comparado en el apartado anterior consumirían demasiada VRAM y mi ordenador no podría manejarlos.

Esto se debe gracias a que Ollama implementa su propio backEnd para la gestión de los modelos que descargamos en la herramienta, como indican en su propio repositorio de github puedes llegar a correr modelos de 7B de parámetros con solo 8GB de ram en el ordenador [17] [3]

Modelo elegido inicialmente

El modelo elegido al inicio del proyecto fue **CodeLlama** por varios motivos:

- Gran soporte y comunidad alrededor de los modelos openSource de Meta: a pesar de ser un modelo especifico para la programación este proviene de Meta 2, que ha sido un Modelo ampliamente usado.

- Permite el uso del lenguaje español tanto en los prompts como en las respuestas del propio modelo: comparándolo con otros de los modelos preseleccionados tras la realización de las pruebas en el entorno local algunos de ellos como DeepSeek si entendía el texto en español, pero las respuestas eran en inglés pudiendo dificultar para algunos alumnos que no tengan conocimientos de este idioma.

- Tamaño ofrecido del modelo: aunque otros de los modelos propuestos tienen tamaños menores de parámetros he considerado que dado que el tamaño de 7B que nos ofrece CodeLlama de parámetros es un tamaño adecuado ademas de cumplir con las características del idioma y que tenemos capacidad suficiente para correrlo, ya que en mi ordenador local que tiene 16GB de memoria RAM empleando Ollama se consigue ejecutar sin problemas.

Forma de adaptación del modelo

El modelo se adaptará siguiendo el enfoque de Lora (Low Rank Adaptation) debido a que no se dispone de suficientes ejemplos de código Java con el fin de poder reentrenar el modelo ajustando ciertas partes del mismo.

Para ello deberemos darle un conjunto de datos:

El conjunto de datos consiste en un total de 162 lineas formadas por los codigos e instrucciones suministradas por los profesores de la asignatura de Fundamentos de Programación, en estos documentos aparecen las guias de como escribir código en java según los criterios de calificación de la misma asi como de programas de ejemplo.

La idea de poder entrenar el modelo con este conjunto de datos es que aprenda de como queremos que los alumnos programen y darles respuestas que vayan asociadas a los conocimientos que se imparten en la asignatura.

Problemas que surgieron con Code-Llama tras el entrenamiento con Lora

Durante los meses de octubre y noviembre tras tener un implementación correcta del modelo y el dataset optimizado para la tarea requerida del problema surgieron diferentes problemas:

- Problemas con el formato del conjunto de datos para adaptarlo al modelo: debido a que el formato de datos que recibe Llama 2 es algo

complejo y a pesar de que el framework empleado para el entrenamiento tiene metodos para adaptar los dataset al formato que necesita el modelo concretamente no se obtuvieron buenos resultados.[15]

- Problemas de tasa de aprendizaje: la tasa de aprendizaje del modelo era muy baja para el conjunto de datos lo cual ya era un indicador de que la inferencia que realizaría con los promts suministrados no seria de calidad.
- Respuestas generadas por el modelo inconsistentes: tras el entrenamiento los resultados no eran los esperados para que el modelo tuviera un uso real.

Reemplazo del modelo

Por ello se decidió ya que la implementación del código era muy similar, realizar el entrenamiento con el modelo superior Llama 3, ya que este en las pruebas preeliminares de seleccion de modelo había obtenido unos resultados similares en los promts y su compatibilidad multilenguaje es incluso mejor que la de codeLlama. El único problema es que este modelo no esta especializado en código, pero al tratarse de un módelo el cual esta pensado para que se use en asignaturas de programación de iniciación es mas que sufuciente para lo solicitado por los tutores del proyecto.

Sorprendentemente la tasa de aprendizaje con el mismo conjunto de datos era muy superior ademas de que los resultados de los promts tanto en lenguaje natural como con código resultaron ser muy superiores a los ofrecidos por codeLlama.

Por lo que el modelo seleccionado finalmente ha sido Llama 3.

Dataset para reentrenar el modelo y frontal

6.1. Obtencion de los datos de entrenamiento

El conjunto de datos suministrado serán programas .java que habrá que adaptar para alimentar al modelo y re entrenarlo.

Estos programas han sido obtenidos principalmente de dos fuentes:

- Documentos de problemas de programación facilitados por la UVA: la profesora y cotutora del proyecto Alma Pisabarro Marrón facilitó una serie de documentos entre los que se encontraban diferentes problemas de programación vistos en la asignatura de Fundamentos de Programación(FPROG). Estos problemas consisitian en problemas de programacioón en el lenguaje java que cubrén conceptos como bucles, condicionales, recusividad, operaciones sobre matrices entre otros.
- Documento de mala praxis: Este documento facilitado por Alma Pisabarro Marrón consiste en una serie de directrices que se siguen en la asignatura de (FPROG) con el fin de que los alumnos aprendan las bases de manera correcta, la salida o uso de practicas consideradas erroneas en el documento implica que el codigo realizado por el alumno no se adecua a los criterios de la asignatura.
- Problemas generados con Inteligencia artifical: gracias a los dos anteriores podido generar mas ejemplos a traves de promts a la inteligencia artificial en concreto al modelo ChatGPT-4o que me ha permitido

generar mas ejemplos para tener un dataset con 177 promts que tratan los diferentes conceptos de la asignatura

6.2. Estructura del conjunto de entrenamiento

Tenemos principalmente dos tipos de ejemplos de promts en el conjunto de datos:

Generación de código en base a lenguaje natural

En este caso tenemos un prompt por parte del usuario que es una instrucción en lenguaje natural para que genere un código concreto mal generado apropósito.

```
{
    "instruction": "Crea una clase Java que sume dos
       enteros usando nombres no descriptivos.",
    "input": "",
    "output": "public class SumaPobre {\n  public
       static void main(String[] args) {\n    int a =
       3;\n    int b = 4;\n    int c = a + b;\n
       System.out.println(c);\n  }\n}"
}
```

Generación de código en base a lenguaje natural y un input

En este caso se le pasa como instrucción del usuario que corrija el codigo que se le pasa como input, que es normalmente el codigo del registro anterior del dataset.

Pero el output es el código corregido según el documento de malas praxis y añadiendo comentarios de lo que esta mal y que se ha corregido

```
{
    "instruction": "Corrige los nombres en la siguiente
        clase para que sean claros y descriptivos.",
    "input": "public class SumaPobre {\n  public static
        void main(String[] args) {\n    int a = 3;\n
          int b = 4;\n    int c = a + b;\n    System.
       out.println(c);\n  }\n}",
    "output": "**Problemas Encontrados:**\n. *Nombrado
       Inadecuado*: variables ‘a‘, ‘b‘ y ‘c‘ carecen de
        significado.\n. *Nombrado Inadecuado*: el
       nombre de clase ‘SumaPobre‘ no es representativo
       .\n\n**Analisis de Impacto:**\nEl codigo es
       dificil de leer y no comunica la intencion. Esto
        complica la comprension y mantenimiento.\n\n**
       Codigo Refactorizado:**\n‘‘‘java\npublic class
       SumaClara {\n  public static void main(String[]
       args) {\n    int primerSumando = 3;\n    int
       segundoSumando = 4;\n    int suma =
       primerSumando + segundoSumando;\n    System.out.
       println(\"La suma es: \" + suma);\n  }\n}\n‘‘‘\n
       \n**Explicacion:**\nSe renombraron la clase y
       las variables con nombres que reflejan su
       funcion real, mejorando legibilidad y claridad."
}
```

De esta manera conseguimos que el modelo comprenda que le estamos dando ejemplos mal resueltos y ademas a estos ejemplos mal resueltos como deben de ser solucionados, no solo generando un código correcto, sino que ademas añadiendo que es lo que estaba mal y que se ha corregido para que siga correctamente el documento de malas praxis.

Elección del frontal

En este caso para interactuar con el frontal se ha optado por una solución open source denominada openwebui[18] que se trata de una plataforma hosteada en el propio ordenador que generamos a traves de un contenedor docker que detecta automaticamente todos los modelos installados que tenemos en ollama, la interfaz es muy similar a la que usa chat-gpt ademas que implementa funciones muy interesantes de manera predeterminada como un motor de inferencia propio para usar con RAG

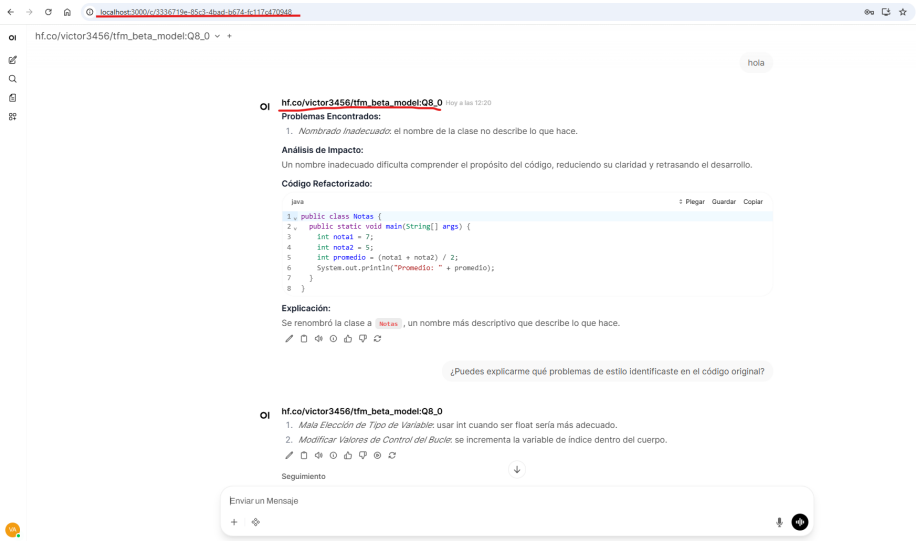

Figura 6.3: Imagen de la interfaz que emplea openwebui [18]

Como se aprecia en la imagen, tenemos que acceder a través del puerto 3000 de nuestra maquina al frontal en el nos aparecerán todos los modelos que tenemos instalados en Ollama y podremos seleccionar uno para interactuar con el.

Tecnologías utilizadas

Las tecnologías empleadas a lo largo del proyecto han sido las siguientes

8.1. Python

Python: principal lenguaje de programación en el que se ha creado el jupyter Notebook para realizar el entrenamiento del modelo de Llama3 al modelo especifico.

8.2. Unsloth

A la hora de realizar el re entrenamiento de modelos tras realizar varias pruebas e investigar diferentes alternativas me he decidido por Unsloth [20] que se trata de un framework open-source que permite realizar fine-tunning de LLMs de una manera muy eficiente.

- Totch (Pytorch): Librería enfocada en el deep learning, puede emplear tanto la CPU como la GPU para realizar las operaciones matemáticas, en el caso de que queramos usar la GPU sera necesario emplear CUDA de NVIDIA. Aunque Unsloth se ejecuta solo empleando GPU en pytorch.[19]
- Transformers: Es la librería de HuggingFace que nos permite cargar modelos ya entrenados por otras personas o modelos open-source que las empresas suben a la propia plataforma.[10]
- Datasets: librería que emplea Unsloth con el fin de el manejo de datasets masivos, con el fin de alimentar a los modelos en el re entrenamiento.[9]

8.3. Discord

Discord es una aplicación gratuita que permite la comunicación a través de texto, voz y vídeo con las personas que tienes agregadas a tu lista de amigos. También permite la creación de servidores en los cuales la gente que se encuentre en ellos puede hablar e interactuar entre ellos. Discord está disponible en Windows, macOS, Linux, iOS, iPadOS, Android y navegadores web [1].

Discord ha sido la herramienta utilizada principalmente para resolver bugs y problemas con la librería de Unsloth[20], ya que como sucede muchas veces en python que es un lenguaje muy dependiente de librerias anidadas a lo largo del desarrollo del notebook y generación del modelo reentrenado han ocurrido bugs que he tenido que comunicar por el foro del grupo de discord, en el cual los desarrolladores de la librería han ayudado a resolver.

8.4. GanttProject

GanttProject es una aplicación gratuita para la elaboración de diagramas de Gantt [8], la cual se ha utilizado para realizar los diagramas de gantt referentes a la planificación del proyecto. El principal motivo de uso es su licencia gratuita. y conocimiento de previos proyectos.

8.5. Overleaf

Overleaf es una herramienta de publicación y redacción colaborativa en línea empleada como editor LaTeX [11]. LaTex es una herramienta de composición de textos centrada en documentos científicos, ya que permite una capacidad de configuración de las formas y texto muy superiores a otros editores de texto.

8.6. Modelo de gran lenguaje

Como hemos comentado en secciones anteriores y tras compararlo con el resto de lenguajes open-source que podíamos escoger Llama 3 de 8B de parámetros, que ademas es uno de los principales motivos por los que se ha seleccionado Unsloth ya que es uno de los modelos soportados por este framework. Y que mas soporte de la comunidad tiene

8.7. Uso del modelo

El modelo se empleará a través de la interfaz creada con el proyecto de openwebui[18] ya que ofrece una cantidad de características muy completas ademas de la portabilidad que permite que se ejecute en un contenedor de docker.

Conclusiones y Líneas de trabajo futuras

Este capítulo se centra en un resumen general de la realización del proyecto así como los problemas y limitaciones encontrados durante su realización y posibles lineas de trabajo futuras para mejorar el chatbot y ampliar su funcionamiento.

9.1. Conclusiones generales

Tras la realización del proyecto se ha llegado a las siguientes conclusiones;

- La búsqueda de un modelo concreto que cumpla con las necesidades del proyecto es clave para su buen funcionamiento, no solo a nivel de que se trate de un modelo adaptado a la tarea, sino que ademas sea capaz de realizar un razonamiento correcto de los datos que se le pasan.
- El empleo del finetunning sobre un modelo es una forma muy solvente de adaptar un modelo ya correcto al contexto que necesitamos, pero encontrar las tecnologías que nos permiten adaptarlo ha sido una labor de mucho esfuerzo e investigación.
- El conseguir un conjunto de datos correcto y lo suficientemente grande para la realización del modelo ha sido un reto, ya que el modelo al contrario de lo que suele ser normalmente común en los modelos cuando se reentrenan que es buscar que ademas de lo que hace el modelo añadir un contexto especifico para el. También se ha buscado

restringir las respuestas del modelo para que no genere código que el estudiante pueda escribir y que es mas avanzado de los contenidos ofrecidos de la asignatura.

- Se han realizado una serie de pruebas sobre el modelo así como pruebas con usuarios con el fin de ver que el modelo satisfaga las necesidades planteadas.

9.2. Limitaciones encontradas

A lo largo del desarrollo de este proyecto nos hemos encontrado con algunas limitaciones principalmente podemos destacar dos:

- Un aspecto en el que el proyecto se ha visto bastante limitado es a la hora de encontrar ejemplos de código para suministrar como conjunto de datos al modelo para reentrenarlo. Ya que inicialmente se ha partido de una batería de problemas suministrados por la profesora y cotutora del proyecto Alma Pisabarro Marrón, estos problemas nos han servido para obtener los primeros datos para el dataset, el resto se han generado en base a ellos empleando inteligencia artificial. Aun así sería adecuado seguir obteniendo mas problemas de la asignatura para poder aumentar el conjunto de datos suministrado.

- El otro aspecto limitante ha sido la disposición de tiempo. Debido a que desde que empece este máster he trabajado a tiempo completo en mi trabajo como desarrollador de software, lo que ha sido un limitante en cuanto al tiempo que le he podido dedicar a este proyecto. Centrando mis esfuerzos principalmente en los fines de semana. Lo que ha causado que el proyecto se haya retrasado de la fecha prevista en las fechas establecidas originalmente.

9.3. Lineas de trabajo futuras

Al tratarse de un proyecto principalmente de investigación y que se ha podido implementar una solución funcional, surgen a partir de este proyecto múltiples lineas de mejora:

Una de ellas y para mi la mas importante es la de aumentar el conjunto de datos que se le suministran en el modelo en el dataset, con el fin de que este obtenga aun mejor el contexto y permita realizar mejores inferencias.

Otra linea de mejora ir actualizando el modelo que se emplea, en la actualidad se ha seleccionado llama 3, por la compatibilidad con la herramienta empleada para el entrenamiento Unsloth, esta herramienta se emplea mucho y tiene una gran comunidad. Por lo que seguirán actualizandose para admitir nuevos modelos open-source que vayan saliendo con el tiempo, por lo que sería muy interesante actualizar el modelo empleado dentro de la familia de Llama, actualmente ya hay versiones para Llama 4, pero desgraciadamente para modelos de 16B de parámetros, lo que los hace demasiado grandes para la tarea designada en este proyecto, pero seguro que en el futuro se sacan versiones de menor tamaño y el razonamiento del modelo seguro mejora.

Otra linea de mejora paralela es emplear el modelo original y la estructura de este proyecto para otras asignaturas de la carrera, ya que lo que uno de los principales logros de este proyecto ha sido encontrar una forma relativamente rápida y fácil de poder generar nuevos modelos adaptados al contexto, a través de las herramientas empleadas. De esta manera cada asignatura podría tener su modelo especializado y obtener respuestas adaptadas siempre al contexto de la asignatura. Lo cual ayudaría que los alumnos no usasen modelos generales como chat-gpt o gemini.

Apéndices

Apéndice A

Especificación de Requisitos

A.1. Introducción

En este apéndice se describen los requisitos generales del proyecto

A.2. Objetivos generales

A.3. Catalogo de requisitos

A.4. Especificación de requisitos

Apéndice B

Especificación de diseño

B.1. Introducción

B.2. Diseño de datos

B.3. Diseño procedimental

B.4. Diseño arquitectónico

Apéndice C

Documentación de usuario

C.1. Introducción

En este apéndice se hablará de la documentación requerida para que los usuarios puedan ejecutar el modelo generado en este proyecto en su ordenador local.

C.2. Requisitos de usuarios

El usuario debe tener conocimientos básicos sobre instalación de programas desde internet, asi como conocimientos básicos de uso del terminal de windows, linux o mac. Ya que todas herramientas son compatibles con estos sistemas operativos.

C.3. Instalación

El usuario debe de tener instalado los siguientes programas en su ordenador:

- Ollama: herramienta que nos permitirá descargar el modelo, para descargalo puede hacerse desde la siguiente url https://ollama.com/
- docker: herramienta de gestión de contenedores que nos permitirá desplegar el frontal con el que podremos ejecutar el modelo desde una interfaz web muy similar a chat-gpt. Para descargarlo lo podemos hacer desde el siguiente enlace https://www.docker.com/products/docker-desktop/

C.4. Manual del usuario

Una vez tenemos todos los programas descargados en el ordenador y tanto docker como ollama en ejecución debemos de usar el terminal, en este caso las imagenes se corresponden con el terminal de windows pero los pasos serían los mismos en el resto de sistemas operativos.

Lo primero es bajarnos del repositorio de huggingFace en el que se encuentra el modelo para ollama para ello lanzamos el siguiente comando en el terminal:

- ollama run hf.co/victor3456/tfm_beta_model:Q8_0

Esto nos permitirá instalar el modelo dentro de ollama de nuestro ordenador si ejecutamos el comando

- ollama list

Nos aparecerá algo así:

```
PS C:\Users\herna> ollama list
NAME                                ID              SIZE      MODIFIED
hf.co/victor3456/tfm_beta_model:Q8_0    34d9e848978c    8.5 GB    2 hours ago
```

Figura C.1: Foto del terminal indicando los modelos existentes en ollama [18]

Ahora debemos instalar en nuestro ordenador la imagen docker asociada ahora es importante saber si nuestro ordenador dispone de una gráfica NVIDIA o no ya que en función de ello tendremos que descargar una imagen diferente

En el caso de que nuestro ordenador disponga de una gráfica NVIDIA descargaremos la siguiente imagen que como se aprecia en el comando tiene en cuenta que podrá emplear cuda:

- docker run -d -p 3000:8080 –gpus all –add-host=host.docker.internal:host-gateway -v open-webui:/app/backend/data –name open-webui –restart always ghcr.io/open-webui/open-webui:cuda

En el caso de no disponer de ella deberemos de emplear la siguiente imagen docker

- docker run -d -p 3000:8080 –add-host=host.docker.internal:host-gateway -v open-webui:/app/backend/data –name open-webui –restart always ghcr.io/open-webui/open-webui:main

El frontal se levantará en el puerto 3000 y nos pedira agregar una cuenta, si es necesario puede crearse con datos dummie si no se quiere proporcionar una dirección de correo real.

Una vez hayamos entrado nos aparecerá algo así:

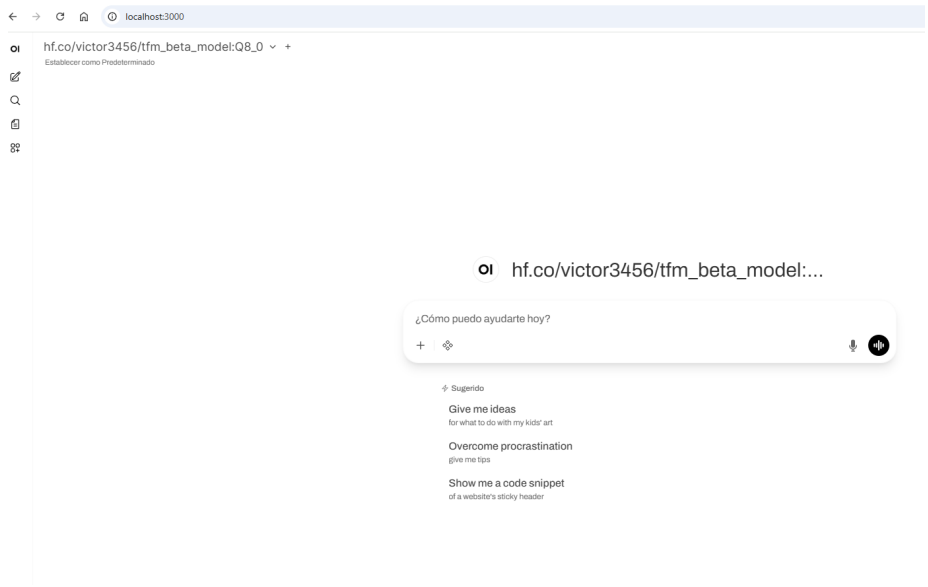

Figura C.2: Foto del frontal la primera vez que iniciamos sesión [18]

Aquí ya tendremos un frontal en el que tendremos un historial con las conversaciones que hemos tenido, siempre y cuando mantengamos el contenedor levantado.

Bibliografía

[1] Jason Citron y Stan Vishnevskiy. Web oficial de Discord. https://discord.com/. último acceso: Diciembre 2025.

[2] Amazon. ¿qué es la rag (generación aumentada por recuperación)?, 2025. [Internet; recuperado el 21 de junio de 2025].

[3] Diego B. ¿qué es ollama? principales características y modelos, 2025. [Internet; recuperado el 5 de julio de 2025].

[4] Sebastian Buzdugan. What's the best gpu for fine-tuning llms? a no-nonsense guide, 2025. [Internet; recuperado el 21 de junio de 2025].

[5] CloudFlare. ¿qué es la adaptación de bajo rango (lora)?, 2025. [Internet; recuperado el 21 de junio de 2025].

[6] Gobierno de España. Definicion ia según la comisión europea, 2023. [Internet; recuperado el 7 de junio de 2025].

[7] DeepSeek-coder. Deepseek-coder repositorio, 2025. [Internet; recuperado el 13 de junio de 2025].

[8] ganttProyect. ganttProyect. https://www.ganttproject.biz/. Último acceso: Diciembre 2025.

[9] huggingFace. datasets. https://huggingface.co/docs/datasets/index. Último acceso: Diciembre 2025.

[10] huggingFace. transformers. https://huggingface.co/docs/transformers/index. Último acceso: Diciembre 2025.

[11] John Hammersley and John Lees-Miller. Web oficial de Overleaf. https://es.overleaf.com/. último acceso: Diciembre 2025.

[12] Lucidchart. Waterfall. La historia detrás del error. https://www.lucidchart.com/blog/es/pros-y-contras-de-la-metodologia-de-cascada. Último acceso: Octubre 2025.

[13] Younes B Tim Dettmers Artidoro Pagnoni Sylvain Gugger Sourab Mangrulkar. Making llms even more accessible with bitsandbytes, 4-bit quantization and qlora, 2025. [Internet; recuperado el 21 de junio de 2025].

[14] Martín Alaimo. Modelo de cascada: Ventajas y desventajas. https://medium.com/@kleer.la/waterfall-la-historia-detr%C3%A1s-del-error-39f589a12115. Último acceso: Octubre 2025.

[15] Meta. formatoLlama2. https://www.llama.com/docs/model-cards-and-prompt-formats/meta-llama-2/. Último acceso: Diciembre 2025.

[16] Ollama. Ollama, 2025. [Internet; recuperado el 5 de julio de 2025].

[17] Ollama. Ollama, 2025. [Internet; recuperado el 5 de julio de 2025].

[18] openwebui. openwebui. https://docs.openwebui.com/. Último acceso: Diciembre 2025.

[19] pytorch. pytorch. https://pytorch.org/. Último acceso: Diciembre 2025.

[20] Unsloth. Unsloth, 2025. [Internet; recuperado el 25 de septiembre de 2025].