UNIVERSIDAD DE OVIEDO
ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN
(EPI)

Copilot inteligente para consultas LINQ/SQL

Autor:
Puga Lojo, Francisco Gabriel

Tutor (Mecalux):
Moldón Redondo, Daniel

Tutor (EPI):
Costa Cortez, Nahuel Alejandro

Memoria entregada en cumplimiento con los requisitos indicados por la asignatura Prácticas de Empresa del grado Ingeniería Informática en Tecnologías de la Información

28 de mayo de 2024

UNIVERSIDAD DE OVIEDO

Resumen

Escuela Politécnica de Ingeniería de Gijón (EPI)

Ingeniería Informática en Tecnologías de la Información

Copilot inteligente para consultas LINQ/SQL

por Puga Lojo, Francisco Gabriel

El manejo de bases de datos es fundamental para la gestión de información en las empresas hoy en día. En un mundo donde la información es clave, poder procesar datos importantes es vital para mantener y hacer crecer una empresa.

Sin embargo, trabajar con bases de datos puede ser complicado y requiere una formación específica que muchas personas fuera del ámbito informatico no tienen ni tiempo para aprender.

Mecalux es una de las compañías líderes en tecnología intralogística a nivel mundial. Es puntera en automatización de almacenes y desarrollo de software. En estas prácticas, he contribuido a desarrollar un asistente que ayuda a generar código **LINQ** SQL y que también explica las consultas generadas.

El objetivo de este asistente es facilitar el trabajo de los empleados, permitiéndoles crear y entender consultas SQL sin necesidad de conocimientos técnicos profundos, y simplificando las tareas para quienes sí los tienen. Esto agiliza los procesos internos y mejora la eficiencia en la toma de decisiones, lo cual es crucial en el entorno empresarial actual.

Índice general

Lista de Abreviaturas

MSS	**M**ecalux **S**oftware **S**olutions
IT	**I**nformation **T**echnlogies
I+D+I	**I**nvestigación, **D**esarrollo e **I**nnovación
IA	**I**nteligencia **A**rtificial
LLM	**L**arge **L**anguage **M**odel
LoRA	**L**ow **R**ank **A**daptation
CUDA	**C**ompute **U**nified **D**evice **A**rchitecture
POO	**P**rogramación **O**rientada a **O**bjetos
RAG	**R**etrieval **A**ugmented **G**eneration

Capítulo 1

Introducción

El alumno y la empresa

Mecalux es una empresa reconocida internacionalmente en el sector de soluciones de almacenamiento. Desde su fundación en 1966, ha desarrollado un amplio portafolio que abarca una variedad de sistemas de almacenamiento, como estanterías, almacenes automatizados y soluciones de software para la gestión de almacenes.

Con una presencia global, **Mecalux** opera en numerosos países, gestionando un gran volumen de operaciones y personal especializado. Las prácticas fueron realizadas de manera presencial en las **oficinas de MSS en Gijón.** [1]

NOTA

Aunque las prácticas fueron presenciales, **Mecalux** tiene una metodología de teletrabajo muy arraigada de la cual muchos empleados de IT siguen beneficiándose.

Es notable cómo esta flexibilidad está bien integrada en su rutina diaria, y parte de la comunicación con el equipo ha sido de de esta manera.

Durante estas prácticas de empresa, tuve la oportunidad de formar parte del equipo de Data Analytics en la división de MSS. En el próximo capítulo se detallarán las actividades llevadas a cabo por este equipo, junto con sus metas y objetivos.

[1] Mecalux Software Solutions es la división de Mecalux dedicada enteramente al desarrollo de software para almacenes y logística.

Capítulo 2

Data Analytics

Innovación y solución de problemas

2.1. Contexto

He tenido la fortuna de trabajar dentro del departamento de Data Analytics, el cual se encarga de recopilar, limpiar e interpretar conjuntos de datos para responder a preguntas o resolver problemas. Es un equipo muy multidisciplinar, y me sorprendió positivamente que, aunque la mayoría de sus miembros no provienen de estudios de Informática (sino de áreas como Física, Matemáticas, etc.), todos poseen una gran capacidad para reflexionar y resolver problemas de manera lógica, metódica y, sobre todo, en equipo.

Además de su labor como analistas de datos, este equipo también cumple la función de resolver inconvenientes que puedan surgir en otros departamentos y se encarga de investigar y poner a prueba nuevas soluciones antes de su implementación en el entorno de producción. Se podría decir que realizan funciones similares a las de I+D+I, ya que investigan, desarrollan y prueban nuevas soluciones para mejorar los procesos y resolver problemas dentro de la empresa.

Durante mis prácticas, mi labor ha sido una combinación de investigación y desarrollo de software, lo cual ha sido posible gracias a la naturaleza y dinámica de este equipo.

2.2. Metodología de Trabajo

Como se mencionó en la introducción, parte del equipo realiza teletrabajo, algunos de manera ocasional y otros de forma regular. De hecho, hay un integrante que ni siquiera reside en Asturias. El equipo realiza reuniones diarias (dailys), a las cuales tuve el placer de asistir en la última etapa de mis prácticas. En estas reuniones, los miembros explican el progreso y las tareas realizadas el día anterior. Estas reuniones se llevan a cabo en salas especialmente equipadas en el edificio, las cuales cuentan con cámaras y proyectores para que los empleados que están teletrabajando puedan participar activamente en las reuniones.

2.2.1. Dinámica de trabajo durante las prácticas

El departamento estaba enfocado en sus proyectos, por lo que mi equipo de trabajo habitual se redujo a otro alumno de prácticas de la carrera de Ingeniería Informática del Software, y nuestro tutor, perteneciente a Data Analytics, que se encargaba de que fuésemos por el buen camino y con el que hacíamos también nuestras propias reuniones diarias para ver el progreso de nuestro trabajo.

Mi compañero y yo teníamos horarios diferentes, por lo que coincidíamos en las oficinas durante un tiempo limitado, lo que conllevó a la necesidad de coordinarnos lo mejor posible y así poder tener los objetivos claros y aprovechar al máximo el tiempo que teníamos juntos para desarrollar, estructurar nuestras tareas y debatir los problemas a los que nos enfrentaríamos.

Al entrar antes a la oficina, me encargaba de hablar con el tutor para ver si había nuevas opciones que explorar o investigar, por donde seguir si habíamos conseguido los resultados esperados, o en caso contrario comunicar los inconvenientes, y coordinar la planificación para el día.

De esta manera, cuando mi compañero llegaba, era todo más fácil para los tres, el tutor no necesitaba volver a explicarlo todo y yo podía comunicarle de forma efectiva que teníamos que hacer, las novedades, y como nos podríamos distribuir el trabajo.

Por otra parte, cuando yo me marchaba mi compañero seguía desarrollando y trabajando en el proyecto, por lo que me comunicaba sus avances por la plataforma de Teams para que cuando llegase al día siguiente supiese de manera más rápida que era lo que se había hecho.

Nuestra comunicación fue en todo momento muy buena, nos sentábamos en mesas próximas por lo que nos acercábamos a hablar sobre como realizar el trabajo frecuentemente, y cuando no queríamos molestar utilizábamos el chat de Teams.

Cabe resaltar que aunque no trabajase con todos los integrantes del departamento de Data Analytics directamente al no pertenecer al proyecto en sí, muchos de ellos me ayudaron a lo largo de las prácticas y hubo una comunicación cercana y efectiva independientemente de ello.

Capítulo 3

MSSCopilot

Generación de Código Automática

3.1. Objetivo de las prácticas

El objetivo principal de las prácticas fue desarrollar una herramienta que permitiera a los usuarios generar código de manera automática. Esta herramienta, denominada MSSCopilot, debía ser capaz de generar código en **C#** / **LINQ** a partir de la base de código fuente existente en la compañía.

EJEMPLO

Un empleado le dice al MSSCopilot que desea obtener una lista con todos los Warehouses, el Copilot respondería con un código como el siguiente:

```
1     using (var context = new MyContext())
2     {
3         var result = context.Warehouse.Select(x => x);
4     }
```

Durante el desarrollo del programa se elaboraron más funcionalidades de las previstas inicialmente como es la explicación de código, y otras más que se explicarán con mayor profundidad posteriormente.

El estado del proyecto al comenzar las prácticas estaba en una etapa inicial, lo que permitió un entendimiento más profundo del mismo y facilitó la integración de nuevas ideas y funcionalidades desde el principio.

3.2. Desarrollo

Definir las tecnologías utilizadas en la realización del MSSCopilot resulta complicado porque como se ha comentado en la sección 2.1, aunque se trate de desarrollar un producto de software, ha sido un proyecto en el que, por la naturaleza de la IA, que sigue siendo un sector de tecnologías emergentes, las directrices a seguir no son tan claras. Debido a esto, gran parte del proyecto ha consistido en un proceso de investigación en el que se han utilizado tecnologías y técnicas que, con el tiempo, se han descartado en favor de otras opciones mejores a medida que se identificaban.

Hay mucho trabajo realizado que he omitido porque la memoria de prácticas no pretende ser una memoria técnica, sin embargo, es importante tener en cuenta que estas tecnologías y explicaciones son solo una parte del panorama completo, y el proyecto ha implicado una exploración más amplia de diversas herramientas y enfoques en el campo de la inteligencia artificial.

3.2.1. Conocimientos previos

Los **LLM** son un tipo de modelo de inteligencia artificial diseñado para comprender y generar texto, como es el famoso **GPT** , para este proyecto se ha utilizado **Ollama** .

En la descripción de las prácticas una de las tareas fundamentales era diseñar y entrenar el sistema a partir de la base de código fuente existente en la compañía. Para entrenar un **LLM** con información nueva hay dos opciones:

- Ajustar el modelo preentrenado utilizando un cojunto de datos más pequeño y específico, proceso conocido como **Fine-tuning.**
- Convertir palabras, frases o textos completos en vectores numéricos. Estos **Embeddings** capturan el significado y las relaciones semánticas del texto. Se pueden convertir las preguntas del usuario en embeddings y compararlos con los embeddings de la información que tenemos almacenada en una BDD, y si ambos tienen x similitud darle esa información al **LLM** en tiempo real para que responda con nuestra información.

En resumen, en la primera opción ponemos al modelo a estudiar los conocimientos nuevos generando así un nuevo modelo, y en la segunda opción es como si el modelo para cada pregunta, consultase en un diccionario/libro con respuestas, buscando la que más se parezca a la pregunta del usuario.

Durante el transcurso de las prácticas he tenido que realizar el programa utilizando ambas opciones.

3.2.2. Embeddings

Inicialmente se optó por utilizar Embeddings, usando **SQLite** para la BDD, y **C#** por compatibilidad con toda la infraestructura de la empresa, permitiendo el poder integrar a futuro esta utilidad de asistente de código a todos los servicios que ofrecen.

La base de datos contenía queries correctamente escritas en **C#** junto con su respectiva descripción, y el embedding asociado, esto se descartó posteriormente ya que no era viable para la empresa ponerse a generar queries con descripciones para cada casuística, y en versiones posteriores se optaría por pasarle el contenido de las tablas. Para generar la BDD se creó una clase que lee archivos de código fuente **C#** de la empresa que se filtran para extraer el contenido. Con las tablas el procedimiento fue el mismo, gracias al uso de **Entity Framework** .

Dicha comparación que se realiza entre Embeddings de la pregunta del usuario y la BDD se tuvo que implementar en código, se implementó tanto la similitud coseno como la euclídea:

Para calcular la similitud coseno entre dos vectores de embeddings se han de realizar dos pasos:

Sean **A** y **B** los vectores con embeddings cuya similitud se desea calcular. El primer paso consiste en obtener el valor del coseno del ángulo que forman ambos vectores.

Sea θ el ángulo que forman los dos vectores. Su coseno se puede calcular de la siguiente manera:

$$cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Dado que se da que $cos(\theta) \in [-1, 1]$, el segundo paso ha de ser que el resultado pase a encontrarse en el rango $[0, 1]$, con el fin de simplificar el cambio de un algoritmo de similitud a otro. De esta manera, el valor de la similitud entre los vectores **A** y **B** sería el siguiente:

$$S_C(A, B) = cos(\theta) \cdot 0.5 + 0.5$$

Sean **A** y **B** los vectores con embeddings cuya distancia se desea calcular. Calcular la distancia entre ambos consiste en calcular la norma euclídea de su diferencia:

$$d(A, B) = |A - B\| = \sqrt{(A_1 - B_1)^2 + (A_2 - B_2)^2 + ... + (A_n - B_n)^2}$$

En este caso, no es necesario modificar el rango, ya que se da que $d(A, B) \in [0, \infty)$. No obstante, es necesario realizar un segundo paso. El valor que se ha obtenido no es la similitud entre los vectores, sino su distancia. Por tanto, realizar una comparación con el valor obtenido directamente resultaría en resultados incorrectos, ya que a mayor valor de $d(A, B)$, más distintos son los vectores.

Por tanto, para calcular la similitud entre los vectores en base a la distancia euclídea, es necesario multiplicar el resultado por -1. Así, la similitud entre los vectores se calcularía de la siguiente manera:

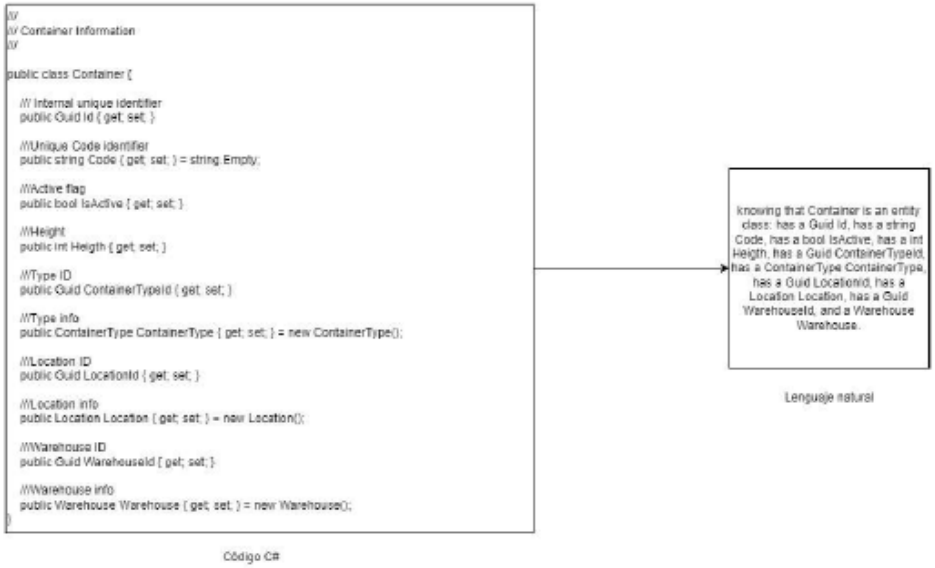
$$S_E(A, B) = -d(A, B)$$

Se optó por utilizar la distancia euclídea.

Con estas tecnologías y métodos se creó la primera versión funcional del Copilot, de la que sugieron múltiples versiones y mejoras que se aplicaron en cada una de ellas.

Algunas de las mejoras que se fueron aplicando tras prueba y error:

- Se realizaron pruebas con diferentes modelos de **Ollama** y se optó por el codellama-instruct, por su capacidad para generar código correctamente y también explicarlo. Nos dimos cuenta que aún codellama siendo un modelo especializado en código, su variante Instruct le confería la capacidad de explicar código y por ende tenía un procesamiento del lenguaje natural superior que otras versiones de codellama no disponía. Probamos convirtiendo las tablas de la BDD al lenguaje natural y tanto la comprensión del modelo como la similitud de los embeddings mejoró sustancialmente.



- Se optó por dejar de lado **SQLite** para utilizar **PostgreSQL** junto con un módulo lllamado pgvector especializado en operaciones vectoriales que ya contaba con operaciones de distancia euclídea, por lo que no había necesidad de seguir usando código propio en **C#** , lo cual disminuyó enormemente los tiempos de respuesta.
- Se hicieron benchmarks y comparaciones entre el modo normal de funcionamiento del modelo en el que guarda el historial de todo lo que se ha escrito para poder responder a preguntas anteriores, y el modo stateless en el que no guarda información entre preguntas.
- Se implementó un mecanismo para detectar que la pregunta del usuario está pidiendo una explicación, preparando al modelo para que responda correctamente.
- Se diseñaron métodos para mejorar el tiempo de respuesta del Copilot, ya que se pretendía utilizar el hardware de los equipos de los trabajadores, por lo que tenía que ser capaz de responder en un tiempo aceptable incluso con hardware modesto.

- Se creó un menú inicial a la hora de arrancar el programa(se ejecuta en una terminal), en el que te permite escoger el modelo **LLM** que se desee, y un submenú con todas las diferentes configuraciones creadas y tecnologías utilizadas a lo largo de las prácticas, como puede ser utilizar el Copilot en modo stateless o no, usar **PostgreSQL** o **SQLite** , o utilizar Embeddings o no.

3.2.3. Fine-tuning

Una vez que la versión con Embeddings era funcional, y respondía en unos tiempos aceptables, se optó por probar el Fine-tuning para comparar los tiempos de respuesta.

Esta decisión era algo que personalmente desde el principio descartaba porque requiere un gasto computacional enorme y se requieren de muchísimos datos para que el Fine-tuning pueda tener un impacto significativo en el modelo, cosa en la que el tutor de prácticas estuvo de acuerdo por lo que esta etapa estaba inicialmente destinada a estimar cuanto podría llegar a costar hacerlo, y como hacerlo.

Investigando sobre el Fine-tuning descubrí un método llamado LoRA para poder hacer Fine-tuning sin tener que preentrenar el modelo **LLM** entero, facilita la adaptación de modelos de lenguaje preentrenados a nuevas tareas sin un reinicio completo del entrenamiento, digamos que es como acoplar al modelo preexistente, en este caso Codellama, otro 'modelo' LoRA en tiempo de ejecución.

Esto viene muy bien a **Mecalux** no solo porque se ahorraría los costes de un Fine-tuning clásico, sino que si su base de datos crece, no es necesario volver a entrenar de cero el modelo, varios adaptadores LoRA pueden ser aplicados a un mismo modelo, y la generación de estos adaptadores es muy poco costosa computacionalmente.

Para esta nueva etapa el código en **C#** no cambió demasiado, y el tiempo fue destinado en como generar estos adaptadores. Llamasharp que es la biblioteca **C#** /.NET usada para ejecutar **LLM** era incapaz de utilizar estos adaptadores correctamente, por lo que sabiendo que está basado en el proyecto original de **Ollama** llama.cpp, escrito en **C** /**C++** y **Python** , se probó a utilizar los adaptadores generados en **C++** desde un Ubuntu, y se confirmó que allí si funcionaba. Aunque nos encontramos con ese bug el cual fue notificado, tuvimos la suerte de poder utilizar uno de los programas en **Python** que vienen dentro del proyecto de llama.cpp que permite fusionar el modelo base con el adaptador LoRA y generar un modelo nuevo, que a efectos prácticos es lo mismo.

Como el proceso de generar adaptadores y combinarlos era una tarea muy tediosa y que requería de cierta complejidad y conocimientos realicé un programa en shell script para facilitar todo el proceso y de paso generar métricas. Por ejemplo la cantidad requerida de RAM para generar los adaptadores era muy elevada y sobrepasaba con creces la de los portátiles de trabajo de la empresa, por tanto el script se encarga de generar memoria swap en este caso, evitando un fallo de segmentación.

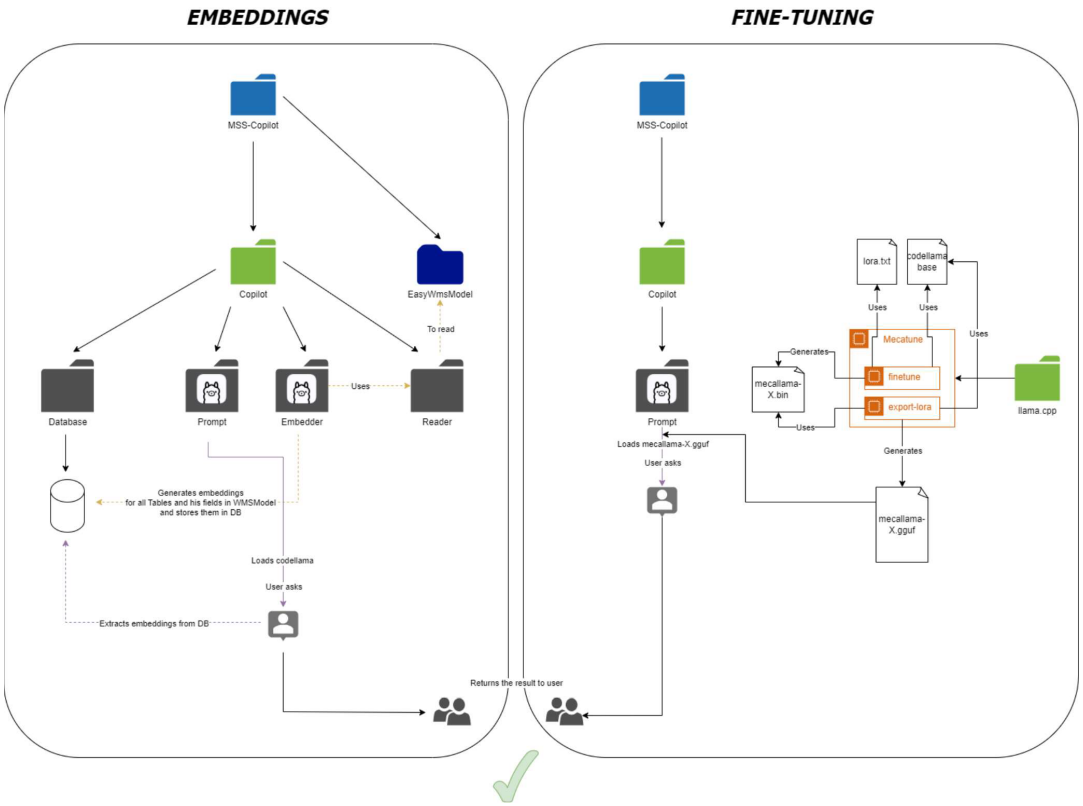
3.2.4. Embeddings vs Fine-tuning

Al finalizar las prácticas se hicieron los benchmarks pertinentes y en cuanto a tiempos de ejecución ambos eran similares, en parte porque se dedicó mucho esfuerzo en optimizar al máximo la opción embeddings, pero en la resolución de las preguntas y en coherencia tanto en las explicaciones como en la generación de código ganaba los Embeddings.

La explicación a esto además de lo previamente mencionado es que el adaptador necesita de un fichero con la información que se quiere aprender, y no había mucha información en cuanto a como de redactar dicho fichero, se probó a escribirlo en lenguaje natural y después siguiendo directrices que se usaron para entrenar a codellama, pero no se hicieron unas pruebas exhaustivas por no disponer de más tiempo.

Es probable que afinando la manera de elaborar ese fichero las comparativas cambien a favor del Fine-tuning usando LoRA.

3.2.5. Esquema general del proyecto



3.3. Relación de las tareas desarrolladas con los conocimientos adquiridos en los estudios universitarios

Al reflexionar sobre mis estudios, identifico varias asignaturas y temas que han sido particularmente relevantes y valiosos para mi formación. Estos incluyen:

- **Sistemas Inteligentes e Inteligencia de Negocio:** Cursé Sistemas Inteligentes en el 2020-2021 e Inteligencia de Negocio el año posterior, así que no puedo ser muy específico en cuanto al temario dado pero creo que son dos asignaturas que aunque hayan pasado 2-3 años desde que las he cursado me han permitido realizar las prácticas con un contexto general sobre la inteligencia artificial, conceptos como el procesamiento del lenguaje natural, o el entendimiento de como funciona un **LLM** por debajo, y conceptos como tokens, training data, sobreajuste, subajuste, etc..
- **Bases de Datos y Sistemas de Información:** Bases de datos ha sido una asignatura esencial ya no solo por el hecho de almacenar la información de los Embeddings sino que en el contexto del programa, que es un asistente de sentencias SQL, ha sido esencial para poder probar luego la IA y saber que las queries que me daba eran correctas o no, además de que en esta asignatura aprendí **PostgreSQL** . Destaco también la asignatura de Sistemas de Información porque en ella utilicé **SQLite** , y fue la primera toma de contacto que tuve entre backend y BDD.
- **Metodología de la Programación y Tecnologías y Paradigmas de la Programación:** Del caso de Metodología de la Programación recalcar la importancia de haber aprendido polimorfismo, y de Paradigmas de la Programación los patrones de diseño, cabe recalcar que prácticamente todo el conocimiento backend de la carrera ha sido en **Java** , y que **C#** se utiliza únicamente en las prácticas de Seguridad y muy muy superficialmente, pero con los conocimientos de estas asignaturas no ha sido nada complicado hacer una abstracción de todos estos conocimientos de programación y aplicarlos en **C#** , la adaptación fue bastante más amena de la que pensaba en un principio.
- **Programación Concurrente y Paralela:** Si bien he utilizado paralelización de tareas ha sido de una manera muy básica, ya que las librerías se encargaban de hacerlo todo más fácil, pero los conocimientos teóricos han sido útiles de igual manera, y recalcar sobre todo los contenidos aprendidos respecto a **CUDA** , el modelo **LLM** se puede ejecutar tanto en CPU como GPU, y toda la problemática que implica programar utilizando **CUDA** se me hizo muy amena gracias a esta asignatura, sin ella se me hubiese hecho muy cuesta arriba este aspecto.

- **Configuración y Evaluación de Sistemas:** Me ha parecido una asignatura fundamental y que he recordado muchas veces a la hora de realizar las prácticas, he tenido que hacer muchísimos benchmarks y pruebas de rendimiento frente a muchas casuísticas y hacer muchas gráficas que gracias a esta asignatura considero que he hecho correctamente. Me gustaría comentar que gracias al paradigma de esta asignatura me percaté que el programa daba unos tiempos de respuesta anormales y que eran debidos a dos factores principales, uno que la capacidad de cómputo era tal que el procesador llegaba al estrangulamiento térmico y capaba sus capacidades para no prenderse fuego, y que la discrepancia entre múltiples equipos del departamento se debía a que muchos de estos eran portátiles y su potencia se veía enormemente reducida cuando no estaban enchufados a la corriente independientemente de que se hubiese configurado Windows para que esto no pasase.

- **Sistemas Distribuidos y Algoritmia:** Son dos asignaturas que en lo que es conocimientos no me han servido especialmente, quizás si Algoritmia de manera más subconsciente, pero son asignaturas en las que utilicé muchísimo **C** y **C++** , lo cual me vino bien de cara a la parte de Fine-tuning.

- **Sistemas Operativos:** Considero que es otra de las asignaturas fundamentales, ya que nos enseñan a manejarnos bien en un sistema basado en Linux, crear shell-scripts, y hacernos descubrir este aspecto de la informática y la importancia de la consola, que de no ser por esta asignatura, probablemente no tendría los conocimientos que tengo hoy en día.

Capítulo 4

Lecciones aprendidas y conclusiones

Durante el desarrollo de mis prácticas, he adquirido tanto habilidades blandas como habilidades técnicas.

Nunca había utilizado **C#** , pero confiaba en la capacidad de abstraer mis conocimientos de otros lenguajes POO, y considero que no tengo nociones muy avanzadas en cuanto a inteligencia artificial respecta, pero si interés por la temática, ya que tenía pensado realizar por mi cuenta algo bastante parecido a lo que se requería en la oferta, por lo que me aventuré a hacerme cargo de las responsabilidades que se requerían para las prácticas, y estoy muy contento con la decisión.

He aprendido muchísimo más del campo de la inteligencia artificial, y he disfrutado mucho de toda la labor de investigación que realizamos para poder ofrecer soluciones al proyecto, conceptos como LoRA, RAG, o el descubrimiento de modelos **LLM** emergentes que usan 2 bits para los pesos permitiendo ejecutar estos modelos en dispositivos de recursos limitados como podrían ser dispositivos móviles, entre otros muchos más conocimientos adquiridos.

He experimentado una gran satisfacción durante mis prácticas, ya que he tenido la oportunidad de trabajar con una variedad de herramientas, tecnologías y librerías (pgAdmin, **SQLite** , **CUDA** , pgvector, llamasharp, llama.cpp), así como con diferentes lenguajes de programación (**C** , **C++** , **C#** , **Python** , Shell, PL/pgSQL) y sistemas operativos (Windows, WSL). Esta diversidad ha sido muy gratificante y ha complementado mi deseo de adquirir un conocimiento amplio y una visión general de múltiples áreas.

Los problemas que tuve con la librería de llamasharp me hizo crear mis primeras issues en GitHub lo cual junto a el uso de Git para el proyecto de MSSCopilot me acercó más a los conocimientos necesarios y buenas prácticas para el desarrollo de código colaborativo.

La necesidad de comunicarme con mi compañero de prácticas de manera efectiva y aprovechar al máximo las virtudes de cada uno siendo consciente de nuestras flaquezas y diferencias me ha servido mucho como crecimiento personal a la hora de trabajar en equipo, un poco de liderazgo, y sobre todo de adaptabilidad y resolución de problemas.

En general ha sido una experiencia en la que aunque la tutela del tutor ha sido excelente y ha estado presente y preocupado por nosotros en todo momento a pesar de sus responsabilidades, ha habido bastante autonomía y he sentido que se le ha dado mucha importancia a la capacidad de resolución de problemas y el pensamiento crítico lo cual valoro muchísimo.

Todo el departamento de Data Analytics me trató excelentemente y he tenido un trato muy cercano, en general con todo el personal de la empresa pero especialmente con ellos. Ha sido muy enriquecedor en todos los sentidos ver como opera una empresa de tal importancia como es **Mecalux** .