UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

IAGScore

Presentado por Pedro Antonio Abellaneda Canales
en Universidad de Burgos — 12 de mayo de 2025
Tutor: Raúl Marticorena Sánchez

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. nombre tutor, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. Pedro Antonio Abellaneda Canales, con DNI dni, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

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, 12 de mayo de 2025

Vº. Bº. del Tutor:

D. nombre tutor

Resumen

Actualmente, los avances relacionados con la inteligencia artificial han supuesto una revolución en la forma en la que interactuamos con la tecnología, destacando en este sentido los modelos de lenguaje a gran escala (LLM). La irrupción de estas tecnologías ha supuesto un punto de inflexión para la automatización de tareas complejas, generación de contenido, de código o búsqueda de información.

En este contexto, este Trabajo Fin de Grado propone el desarrollo de una herramienta basada en el uso de modelos de lenguaje a gran escala para optimizar la evaluación de tareas de programación realizadas por alumnos. La propuesta inicial consiste en dotar a los usuarios de herramientas para la importación de rúbricas personalizadas en formato Markdown, diseño de prompts personalizados en texto plano, configuración personalizada para determinados modelos automatizando varias de estas tareas a través de una aplicación.

Descriptores

Django, python, PostgreSQL, LLM, Markdown, correción de tareas ...

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

keywords separated by commas.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

Descripción del contenido del trabajo y del estructura de la memoria y del resto de materiales entregados.

2. Objetivos del proyecto

A continuación se detallan los objetivos que motivan la realización de este proyecto:

2.1. Objetivos generales

- Desarrollar un sistema *software* que permita automatizar la evaluación de tareas de programación en cualquier lenguaje.
- Investigar la posibilidad de hacer uso de diferentes modelos de lenguaje avanzado para la evaluación de estas tareas.
- Comprobar la posibilidad de delegar totalmente en estos modelos para la realización de estas tareas.
- Implementar operaciones e interacción con la aplicación:
    - Permitir a los usuarios crear una cuenta.
    - Permitir al usuario importar documentos en formato *Markdown* con los contenidos de sus propias rúbricas.
    - Permitir a los usuarios diseñar o definir los *prompts* con los que interactuarán con los modelos.
    - Permitir a los usuarios configurar diferentes parámetros del modelo que evaluará las tareas.
    - Permitir a los usuarios ejecutar el modelo para que realice la evaluación.
    - Permitir a los usuarios consultar el reaultado de la ejecución de la evaluación.

- Permitir a los usuarios consultar datos sobre el modelo utilizado.

2.2. Objetivos técnicos

- Crear una aplicación web utilizando python y el *framework* **Django**
- Utilizar la arquitectura MVT(Model - View - Template) de Django.
- Utilizar una base de datos **PostgreSQL** para el almacenamiento de datos.
- Utilizar **Ollama** para la gestión de los modelos de lenguaje.
- Utilizar **Celery** y **Redis** para la gestión de tareas asíncronas.
- Utilizar **Docker** para la creación de contenedores y facilitar el despliegue del proyecto.
- Hacer uso de **Sphinx** para la generación de la documentación del proyecto.
- Gestionar el proyecto utilizando la herramienta de gestión de proyectos **Zube** para la planificación y seguimiento del proyecto.
- Realizar test con una cobertuta que garanticen la calidad del producto final.
- Hacer uso del sistema de control de versiones **GIT** distribuido junto con la plataforma **Github**.
- Aplicar la metodología ágil **Scrum** para el desarrollo del proyecto.
- Utilizar herramientas de integración continua **CI/CD**, como las integradas en **Github actions**, comprobando el despliege del proyecto, ejecución de test e integrando test de cobertura de código con **coverage**, control de calidad de código integrando **SonarCloud** con Github actions.
- Utilizar **SonarCloud** para la gestión de la calidad del código y su integración con **Github actions**

2.3. Objetivos personales

- Utilizar el *framework* **Django** para el desarrollo de aplicaciones web.
- Investigar el comportamiento de los modelos con diferentes configuraciones para conocer la viabilidad del proyecto.
- Conocer y utilizar **Docker** para la creación de contenedores y facilitar el despliegue del proyecto.
- Aprender a utilizar herramientas de control de versiones como **GIT** y plataformas como **Github**.
- Aprender a utilizar herramientas de integración continua como **Github actions**.
- Aprender a utilizar herramientas de gestión de proyectos como **Zube**.
- Aprender a utilizar herramientas de calidad de código como **SonarCloud**.
- Aprender a utilizar herramientas de documentación como **Sphinx**.

3. Conceptos teóricos

En este apartado se describen los componentes principales que hacen posible la evaluación automática de tareas mediante modelos de lenguaje de gran tamaño (LLM).

En primer lugar, se explica en qué consisten los LLM y cómo su arquitectura basada en transformadores permite comprender y generar texto de forma avanzada. A continuación, se introduce LangChain como un *framework* que facilita la integración de estos modelos en aplicaciones complejas, y se analiza el uso de Ollama como herramienta para ejecutar los modelos localmente, garantizando privacidad y control sobre los datos.

Además, se detalla el papel de Redis y Celery, que permiten gestionar tareas de forma asíncrona y eficiente dentro del sistema, especialmente útil cuando el procesamiento de las tareas requiere tiempo o recursos considerables.

Finalmente, se menciona el concepto de *prompt engineering*, una técnica que permite mejorar la calidad de las respuestas generadas por los modelos optimizando la forma en que se les formula la entrada, aspecto que puede tener impacto directo en la efectividad del sistema desarrollado.

3.1. Modelos de lenguaje grandes (LLM)

Los modelos de lenguaje grandes (LLM, por sus siglas en inglés) son sistemas de inteligencia artificial diseñados para modelar y procesar el lenguaje humano. Estos modelos forman parte del campo del *procesamiento de lenguaje natural* (PLN), una rama de la inteligencia artificial que estudia cómo las máquinas pueden comprender, interpretar y generar texto o habla en lenguaje humano.

Se denominan 'grandes' porque están entrenados con volúmenes masivos de texto, y tienen una arquitectura muy grande capaz de capturar matices lingüísticos complejos que los modelos más pequeños no pueden.

La gran mayoría de LLMs actuales se basan en la arquitectura Transformer, propuesta por primera vez en el artículo "Attention Is All You Need" [5]. contruyendo una arquitectura basada en mecanismos de atención, sin recurrir a redes recurrentes o convolucionales.

El mecanismo de atención permite al modelo ponderar la importancia relativa de cada palabra en la entrada, considerando su relación con todas las demás en la secuencia. Esto le permite capturar dependencias a largo plazo y relaciones semánticas complejas, incluso entre palabras que están muy separadas en el texto.

3.2. Ollama

Ollama es una herramienta que permite ejecutar modelos de lenguaje grandes de forma local, facilitando así la utilización de LLMs sin necesidad de conectarse a servicios en la nube. Esto representa una ventaja significativa en términos de privacidad, eficiencia y control sobre el entorno de ejecución.

Además de simplificar la instalación y gestión de modelos como LLaMA, Mistral o Phi-4, Ollama ofrece una interfaz sencilla para ejecutar estos modelos desde la línea de comandos o integrarlos en aplicaciones a través de bibliotecas específicas como LangChain.

En este proyecto, Ollama ha sido clave para ejecutar LLMs directamente en el entorno local evitando el uso de servicios externos.

Una de las características más útiles de Ollama es que permite configurar distintos parámetros del modelo, lo que facilita su adaptación a diferentes necesidades o contextos. En particular, en este proyecto se permite configurar los siguientes parámetros:

- **Temperatura**: Controla la aleatoriedad de las respuestas generadas. Un valor bajo produce respuestas más predecibles, mientras que un valor alto genera respuestas más creativas y variadas.
- **Top-p**: También conocido como muestreo de núcleo, este parámetro ajusta la probabilidad acumulativa de las palabras seleccionadas. Un valor bajo limita la selección a las palabras más probables, mientras que un valor alto permite una mayor diversidad en las respuestas.

- **Top-k**: Este parámetro limita el número de palabras candidatas a considerar durante la generación de texto. Un valor bajo restringe la selección a las palabras más probables, mientras que un valor alto permite una mayor variedad en las respuestas generadas.

Además, Ollama facilita la gestión de versiones de los modelos, permitiendo a los usuarios probar diferentes versiones de un mismo modelo o incluso modelos alternativos sin complicaciones.

3.3. LangChain

LangChain es un *framework* diseñado para facilitar la creación de aplicaciones que integran modelos de lenguaje con diversas fuentes de datos, herramientas externas y flujos de trabajo. Aunque su potencial es muy amplio, en este proyecto se ha utilizado de forma básica para conectar modelos de lenguaje locales, gestionados con Ollama, e integrarlos de manera sencilla dentro del entorno de desarrollo en Python.

En concreto, se ha utilizado el módulo `OllamaLLM` de `langchain_ollama`, que permite invocar modelos como LLaMA o Mistral localmente sin necesidad de APIs externas. Esta integración facilita el acceso a modelos avanzados sin comprometer la privacidad de los datos ni depender de servicios en la nube.

LangChain ofrece también funcionalidades más avanzadas como el manejo de memoria conversacional, agentes que interactúan con múltiples herramientas, y la orquestación de cadenas de prompts, aunque estas características no han sido necesarias en el alcance actual del proyecto.

3.4. Gestión de tareas asíncronas con Celery y Redis

En este proyecto se ha utilizado **Celery**, una librería de Python para la ejecución de tareas en segundo plano y procesamiento distribuido, con el objetivo de gestionar de forma eficiente las correcciones automáticas enviadas por los estudiantes. Su integración permite desacoplar el proceso de corrección del flujo principal de la aplicación, evitando bloqueos o tiempos de espera innecesarios en la interfaz de usuario.

Esto resulta especialmente útil en entornos donde se espera una alta concurrencia de usuarios o cuando el procesamiento de texto puede requerir tiempos de cómputo significativos, como es el caso del uso de LLMs.

Para la gestión de mensajes entre la aplicación y Celery se ha empleado **Redis**, un sistema de almacenamiento en memoria que actúa como *message broker*. Redis permite almacenar las tareas en una cola, que posteriormente son recogidas y ejecutadas por los *workers* de Celery.

El flujo básico de trabajo es el siguiente:

1. El usuario envía una tarea desde la aplicación.
2. Esta tarea se encola en Redis como un mensaje.
3. Celery detecta la nueva tarea en la cola y la ejecuta, utilizando un modelo de lenguaje (LLM) gestionado por Ollama.
4. Una vez completada la corrección, la respuesta se almacena para ser presentada al usuario.

Este enfoque ofrece varias ventajas:

- Permite procesar múltiples tareas en paralelo.
- Mejora la escalabilidad del sistema.
- Reduce la carga directa sobre el servidor principal.
- Aumenta la capacidad de respuesta de la interfaz.

3.5. Prompt engineering

El *prompt engineering* es una técnica dentro del campo del procesamiento de lenguaje natural que consiste en diseñar cuidadosamente las instrucciones que se proporcionan a un modelo de lenguaje (LLM) para obtener respuestas más precisas, relevantes o útiles.

Dado que los LLM no tienen comprensión real del contexto, el modo en que se les formula una entrada (prompt) influye significativamente en la calidad y pertinencia de la respuesta. Por ello, estructurar correctamente el mensaje de entrada se convierte en un factor clave para guiar el comportamiento del modelo.

Aunque en este proyecto no se ha aplicado de forma intensiva, se han realizado pruebas con distintas formulaciones de entrada para verificar cómo pequeños cambios en el prompt afectan la salida del modelo. Por ejemplo,

el uso de frases más explícitas o el establecimiento claro del rol del modelo (e.g.,"Actúa como profesor universitario") puede mejorar la coherencia y adecuación de las respuestas al evaluar tareas [2].

Existen técnicas avanzadas dentro del *prompt engineering*, como:

- **Few-shot prompting**: Se incluyen uno o varios ejemplos dentro del prompt para que el modelo aprenda el formato o estilo deseado.
- **Zero-shot prompting**: El modelo genera una respuesta sin ejemplos previos, confiando solo en la instrucción.
- **Chain-of-thought prompting**: Se induce al modelo a razonar paso a paso para tareas complejas.
- **Role prompting**: Se establece un rol específico para el modelo, como "profesor universitario".

Aunque estas estrategias no se han utilizado en profundidad en este trabajo, su conocimiento es relevante para posibles mejoras futuras del sistema, especialmente si se busca afinar el comportamiento del modelo en escenarios educativos específicos.

Subsubsecciones

Y subsecciones.

3.6. Referencias

Las referencias se incluyen en el texto usando cite [6]. Para citar webs, artículos o libros [3], si se desean citar más de uno en el mismo lugar [1, 3].

3.7. Imágenes

Se pueden incluir imágenes con los comandos standard de LaTeX, pero esta plantilla dispone de comandos propios como por ejemplo el siguiente:

Figura 3.1: Autómata para una expresión vacía

3.8. Listas de items

Existen tres posibilidades:

- primer item.
- segundo item.

1. primer item.
2. segundo item.

Primer item más información sobre el primer item.

Segundo item más información sobre el segundo item.

-

3.9. Tablas

Igualmente se pueden usar los comandos específicos de LaTeXo bien usar alguno de los comandos de la plantilla.

Herramientas	App AngularJS	API REST	BD	Memoria
HTML5	X			
CSS3	X			
BOOTSTRAP	X			
JavaScript	X			
AngularJS	X			
Bower	X			
PHP		X		
Karma + Jasmine	X			
Slim framework		X		
Idiorm		X		
Composer		X		
JSON	X	X		
PhpStorm	X	X		
MySQL			X	
PhpMyAdmin			X	
Git + BitBucket	X	X	X	X
MikTEX				X
TEXMaker				X
Astah				X
Balsamiq Mockups	X			
VersionOne	X	X	X	X

Tabla 3.1: Herramientas y tecnologías utilizadas en cada parte del proyecto

4. Técnicas y herramientas

4.1. Metodologías

Scrum

Scrum es un marco de trabajo ágil, líder en el desarrollo de software y la gestión de proyectos, que se distingue por su flexibilidad. Organiza el trabajo en sprints, iteraciones cortas de una a cuatro semanas, cada una entregando un incremento funcional del producto. Al final de cada sprint, se evalúa el resultado para detectar mejoras y ajustar prioridades según las necesidades del proyecto. Los requisitos, que pueden cambiar durante el desarrollo, se recogen en la pila del producto, mientras que las tareas específicas de cada iteración se detallan en la pila del sprint. Este enfoque incremental permite solapar fases del proyecto, optimizando tiempos y facilitando la adaptación a nuevos requerimientos. Scrum promueve la calidad a través del aprendizaje continuo del equipo, generando entregas tempranas de productos utilizables y asegurando una mejora constante alineada con las necesidades del usuario [8].

Kanban

Kanban es un método ágil que se centra en la visualización del flujo de trabajo y la gestión del trabajo en curso. Utiliza un tablero representando las tareas en una serie de tarjetas, permitiendo a los miembros del equipo ver el progreso y las prioridades del trabajo. Kanban Se basa en principios como la limitación del trabajo en curso, la mejora continua y la adaptación a los cambios, es flexible y se puede aplicar a diferentes tipos de proyectos y equipos, lo que lo convierte en una herramienta valiosa para la gestión de proyectos en entornos ágiles [7].

Scrumban

Scrumban es una metodología híbrida que combina elementos de Scrum y Kanban. Se basa en la estructura de Scrum, con sprints y roles definidos, pero incorpora la flexibilidad de Kanban en la gestión del flujo de trabajo. Esta metodología es especialmente útil para equipos que buscan una transición suave entre Scrum y Kanban, o para aquellos que desean aprovechar lo mejor de ambos enfoques. Scrumban se centra en la mejora continua, la visualización del trabajo y la colaboración del equipo, lo que lo convierte en una opción popular para la gestión de proyectos ágiles [4].

5. Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, comentados por los autores del mismo. Debe incluir desde la exposición del ciclo de vida utilizado, hasta los detalles de mayor relevancia de las fases de análisis, diseño e implementación. Se busca que no sea una mera operación de copiar y pegar diagramas y extractos del código fuente, sino que realmente se justifiquen los caminos de solución que se han tomado, especialmente aquellos que no sean triviales. Puede ser el lugar más adecuado para documentar los aspectos más interesantes del diseño y de la implementación, con un mayor hincapié en aspectos tales como el tipo de arquitectura elegido, los índices de las tablas de la base de datos, normalización y desnormalización, distribución en ficheros3, reglas de negocio dentro de las bases de datos (EDVHV GH GDWRV DFWLYDV), aspectos de desarrollo relacionados con el WWW... Este apartado, debe convertirse en el resumen de la experiencia práctica del proyecto, y por sí mismo justifica que la memoria se convierta en un documento útil, fuente de referencia para los autores, los tutores y futuros alumnos.

6. Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

7. Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Bibliografía

[1] Zachary J Bortolot and Randolph H Wynne. Estimating forest biomass using small footprint lidar data: An individual tree-based approach that incorporates training data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 59(6):342–360, 2005.

[2] Prompt Engineering Guide. Guía de prompting para modelos de lenguaje. https://www.promptingguide.ai/es, 2025. [Internet; Accedido el 9 de mayo de 2025].

[3] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[4] Anna Pérez, OBS Business School. La metodología scrumban: cuándo y por qué utilizarla. https://www.obsbusiness.school/blog/la-metodologia-scrumban-cuando-y-por-que-utilizarla, 2014. [Internet; Accedido el 12 de mayo de 2025].

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[6] Wikipedia. Latex — wikipedia, la enciclopedia libre. https://es.wikipedia.org/w/index.php?title=LaTeX&oldid=84209252, 2015. [Internet; descargado 30-septiembre-2015].

[7] Wikipedia. Kanban (desarrollo) — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Kanban_(desarrollo), 2025. [Internet; Accedido el 12 de mayo de 2025].

[8] Wikipedia. Scrum (desarrollo de software) — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software), 2025. [Internet; Accedido el 12 de mayo de 2025].