UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Aplicación de análisis de reseñas basado en modelos de lenguaje grandes mediante prompt engineering.

Presentado por Teodoro Ricardo García Sánchez
en Universidad de Burgos — 2 de enero de 2024
Tutor: Jose Ignacio Santos Martín y Virginia Ahedo García

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

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

Expone:

Que el alumno D. Teodoro Ricardo García Sánchez, 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, 2 de enero de 2024

Vº. Bº. del Tutor:

D. nombre tutor

Vº. Bº. del co-tutor:

D. nombre co-tutor

Resumen

En este primer apartado se hace una **breve** presentación del tema que se aborda en el proyecto.

Descriptores

Palabras separadas por comas que identifiquen el contenido del proyecto Ej: servidor web, buscador de vuelos, android ...

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

El análisis de sentimientos intenta descubrir la actitud de un usuario con respecto a algún tema. Esta puede ser una reseña, un estado afectivo o emocional. Este de análisis se denomina minería de opinión.

En principio el objetivo del análisis de sentimientos es clasificar la polaridad del texto. (positiva, negativa o neutra). Existen muchas formas de llevar a cabo este análisis (localización de palabras, afinidad léxica, métodos estadísticos o tecnicas conceptuales)

En este trabajo vamos a evaluar la utilización del "Prompt Engineering.ªsociado a un modelo grande de lenguaje (LLM) para ese análisis de sentimiento. Evaluaremos su viabilidad y sus resultados.

Usaremos dos medidas para evaluar el modelo, por un lado la precisión (número de elementos entre el número total) y la exhaustividad (recall) que mide la cantidad de elementos que ha sido capaz de identificar (se calcula dividiendo el número de aciertos entre el número total que hemos clasificado de esa forma).

Se considera un modelo equiparable a un humano si consigue un 70 por ciento de aciertos.

1.1. Estructura de la memoria

Introducción: Descripción del tema sobre el que trata el trabajo y su situación. Incluye su estructura (memoria y anexos)

Objetivos del trabajo: Este apartado explica de forma concisa cuales son los objetivos que se persiguen con la realización del proyecto. Se

pueden distinguir entre los objetivos marcados por los requisitos del software a construir y los objetivos de carácter técnico que plantea a la hora de llevar a la práctica el proyecto.

Conceptos teóricos: Exposición de conceptos que facilitan la comprensión del proyecto.

Técnicas y herramientas: Listado de metodologías y herramientas que han sido utilizadas para llevar a cabo el proyecto.

Aspectos relevantes: Muestra aspectos a destacar durante la realización del proyecto.

Trabajos relacionados: Estado del arte en el ámbito del "sentiment analysis" y trabajos similares.

Conclusiones y líneas de trabajo futuras: Conclusiones obtenidas al final del proyecto y posibles ideas futuras.

1.2. Estructura de los anexos

Plan de proyecto software: Planificación temporal y viabilidad económica y legal.

Especificación de requisitos: Objetivos y y requisitos establecidos al comienzo del proyecto.

Especificación de diseño: Recoge los diseños de datos, procedimental, arquitectónico y de interfaces.

Manual del programador: Explica los conceptos más técnicos del proyecto como su instalación, la organización de carpetas y la ejecución.

Manual de usuario: Es la guía de cómo utilizar la aplicación paso a paso.

Sostenibilización curricular: Reflexión personal sobre los aspectos de la sostenibilidad que se abordan en el trabajo.

2. Objetivos del proyecto

El objetivo del trabajo propuesto es desarrollar una aplicación basada en inteligencia artificial (IA) utilizando modelos de lenguaje grandes (LLM [17]) para el análisis, clasificación y calificación de reseñas de usuarios en portales de recomendación y tiendas de comercio electrónico. Los objetivos específicos son los siguientes:

- Crear una aplicación que utilice modelos de lenguaje grandes (como ChatGPT, Bard o Llama) de manera efectiva para analizar y comprender las reseñas de usuarios.
- Diseñar un sistema basado en "prompts", definiendo las tareas, entradas y salidas de la respuesta a una instrucción, para lograr una solución ágil sin necesidad de nuevos entrenamientos.
- Identificar y categorizar aspectos específicos del producto o servicio:
  - Implementar un sistema que pueda identificar y categorizar los aspectos específicos mencionados por los usuarios en sus reseñas.
  - Extraer información relevante sobre productos o servicios, identificando características específicas que generan opiniones.
- Relacionar opiniones y emociones con calificaciones:
  - Desarrollar un mecanismo que pueda relacionar las opiniones expresadas en las reseñas con las calificaciones otorgadas por los usuarios.
  - Analizar las emociones asociadas a las opiniones para comprender mejor la percepción de los usuarios respecto a productos y servicios.

- Implementar técnicas de "*prompt engineering*" [22]:
  - Utilizar técnicas de "*prompt engineering*" para optimizar la interacción con el modelo de lenguaje, mejorando la calidad y relevancia de las respuestas generadas.
  - Facilitar un desarrollo e implementación rápida y de bajo código para la solución propuesta.

El objetivo general es aprovechar la capacidad de los LLM para comprender el lenguaje natural y aplicarlos de manera efectiva en el análisis de reseñas de usuarios, proporcionando una solución ágil y eficiente para extraer información valiosa sobre la percepción de los usuarios en portales de recomendación y tiendas en línea.

3. Conceptos teóricos

Para una mejor compensión del trabajo realizado se recomienda conocer los conceptos teóricos mencionados en esta sección.

3.1. Análisis de sentimiento

El análisis de sentimiento es un campo de estudio en el procesamiento del lenguaje natural (PLN) que se centra en determinar y extraer la polaridad emocional asociada con un conjunto de datos de texto.

El objetivo principal es identificar y evaluar la actitud emocional expresada en un fragmento de texto, ya sea positiva, negativa o neutral.

El análisis de sentimiento implica varios pasos:

- **Preprocesamiento del Texto**: Antes de realizar el análisis de sentimiento, es necesario realizar una serie de tareas de preprocesamiento del texto, que incluyen la eliminación de palabras irrelevantes (stop words), lematización (reducción de palabras a su forma base), y la normalización del texto.
- **Representación del Texto**: El texto debe representarse de una manera que pueda ser procesada por algoritmos de aprendizaje automático. Esto a menudo implica convertir el texto en vectores numéricos utilizando técnicas como la bolsa de palabras (bag-of-words) o modelos de incrustación de palabras (word embeddings).
- **Modelado**: Se utilizan modelos de aprendizaje automático, como máquinas de soporte vectorial (SVM), redes neuronales, o algoritmos

de clasificación, para entrenar un modelo sobre los datos de texto anotados con etiquetas de sentimiento. Durante el entrenamiento, el modelo aprende patrones y características asociadas con la polaridad del sentimiento.

- **Clasificación**: Una vez entrenado, el modelo se utiliza para predecir la polaridad del sentimiento de nuevos textos. El resultado típicamente indica si el texto es positivo, negativo o neutral.

El análisis de sentimiento se aplica en una variedad de campos, desde la minería de opiniones en redes sociales hasta la evaluación de comentarios de clientes en línea.

Puede ser utilizado para comprender la percepción pública de productos, servicios o eventos, así como para tomar decisiones informadas en marketing, atención al cliente y otras áreas empresariales.

3.2. Procesamiento del Lenguaje natural

Es una rama de la inteligencia artificial que se ocupa de la interacción entre las computadoras y el lenguaje humano.
El objetivo principal del PLN es permitir que las máquinas comprendan, interpreten y generen texto de manera similar a como lo haría un ser humano.
Dentro del procesamiento del lenguaje natural hay una serie de conceptos que es importante conocer:

- **Tokenización**: Es el proceso de dividir un texto en unidades más pequeñas llamadas "tokens". Un token puede ser una palabra, una frase o incluso un carácter, dependiendo de la granularidad deseada.

- **Análisis morfológico**: Implica comprender la estructura y la forma de las palabras en un texto.
  Esto puede incluir la descomposición de las palabras en raíces y afijos para comprender su significado.

- **Análisis sintáctico**: Se refiere a analizar la estructura gramatical de las oraciones para comprender la relación entre las palabras.
  Esto implica identificar partes del discurso, como sustantivos, verbos, adjetivos, etc., y cómo se combinan para formar oraciones gramaticalmente correctas.

- **Análisis semántico**: Busca comprender el significado de las palabras y cómo se combinan para formar significados más complejos.
  Involucra la interpretación del contexto y la comprensión de las relaciones entre las palabras.

- **Análisis pragmático**: Considera el uso del lenguaje en situaciones específicas y comprende el significado más allá de las palabras.
  Incluye la interpretación de intenciones, tono y contexto cultural.

- **Reconocimiento de entidades**: Consiste en identificar y clasificar entidades como nombres de personas, organizaciones, fechas, ubicaciones, etc., dentro de un texto.

- **Desambiguación**: Aborda la resolución de ambigüedades en el lenguaje.
  Por ejemplo, entender a qué se refiere un pronombre o resolver el significado de una palabra con múltiples interpretaciones.

- **Generación de lenguaje**: Implica la creación de texto humano legible por parte de las máquinas.
  Puede ser tan simple como completar una oración o tan complejo como la creación de contenido original.

El procesamiento del lenguaje natural se utiliza en una variedad de aplicaciones, como motores de búsqueda, traducción automática, chatbots, resumen automático de textos, análisis de sentimientos y muchas otras áreas donde se requiere comprensión y generación de texto.

3.3. Prompt Engineering

El "*Prompt Engineering*" se refiere a la práctica de diseñar de manera estratégica las instrucciones que se proporcionan a modelos de lenguaje, como los "*Large Language Models*", para obtener resultados específicos.

En otras palabras, implica la formulación precisa de las solicitudes al modelo con el objetivo de influir en la salida de manera deseada.

Esta técnica es especialmente relevante en el contexto de modelos generativos de lenguaje, como GPT-3, donde la calidad y la relevancia de las respuestas pueden depender en gran medida de cómo se presenta la información al modelo.

El "*Prompt Engineering*" se utiliza para optimizar la interacción con estos modelos y obtener resultados que sean más útiles para una tarea particular.

Algunas estrategias comunes de "*Prompt Engineering*" incluyen [22]:

- **Ajuste del Contexto Inicial**: Proporcionar un contexto inicial específico que oriente al modelo hacia la tarea deseada.
- **Modificación de la Formulación**: Experimentar con la redacción de las instrucciones para obtener respuestas más precisas o informativas.
- **Control de la Temperatura**: En modelos de lenguaje como GPT-3, se puede ajustar la "temperatura"para influir en la aleatoriedad de las respuestas generadas.
- **Incorporación de Ejemplos**: Proporcionar ejemplos específicos dentro de la solicitud para guiar al modelo hacia comportamientos deseados.

El "*Prompt Engineering*" es una parte importante de trabajar con modelos de lenguaje generativos para adaptar su salida a las necesidades específicas de una tarea o aplicación. La investigación continua en esta área busca mejorar la eficacia y la coherencia de la interacción con estos modelos.

3.4. Patrones de diseño

Los patrones de diseño en el ámbito del desarrollo de software son soluciones probadas para problemas recurrentes que surgen durante el diseño y la implementación de software. Estos patrones representan las mejores prácticas y experiencias recopiladas por desarrolladores a lo largo del tiempo y proporcionan soluciones estructuradas y reutilizables para situaciones comunes en el desarrollo de software. Los patrones de diseño pueden clasificarse en varias categorías, y algunos de los más conocidos son:

- **Patrones de Creación**:
  - **Singleton**: Garantiza que una clase tenga solo una instancia y proporciona un punto global de acceso a ella.
  - **Factory Method**: Define una interfaz para crear un objeto, pero deja que las subclases alteren el tipo de objetos que se crearán.
- **Patrones de Estructura**:

    - **Adapter**: Permite que la interfaz de una clase sea utilizada como otra interfaz.
    - **Decorator**: Añade funcionalidades a un objeto dinámicamente.

- **Patrones de Comportamiento**:
    - **Observer**: Define una dependencia uno a muchos entre objetos para que, cuando un objeto cambie de estado, todos sus dependientes sean notificados y actualizados automáticamente.
    - **Strategy**: Define una familia de algoritmos, encapsula cada uno de ellos y los hace intercambiables.

- **Patrones de Arquitectura**:
    - **MVC (Modelo-Vista-Controlador)**: Divide una aplicación en tres componentes interconectados, separando la representación de la información, la lógica de negocio y la gestión de la interfaz de usuario.
    - **MVP (Modelo-Vista-Presentador)**: Similar al MVC, pero con una separación más clara entre la lógica de presentación y la lógica de negocio.

La aplicación de patrones de diseño puede llevar a un código más flexible, mantenible y escalable. Sin embargo, es importante aplicarlos con prudencia y adaptarlos según las necesidades específicas del proyecto, ya que una implementación incorrecta puede llevar a complicaciones innecesarias. Los patrones de diseño son herramientas poderosas, pero no son una solución única para todos los problemas.

4. Técnicas y herramientas

En esta parte de la memoria voy a presentar algunas de las técnicas de desarrollo que he utilizado para el desarrollo del proyecto, así como la información relevante de cada una de ellas.

4.1. Técnicas

ORM [8]

Un ORM (Object-Relational Mapping) es una técnica de programación que se utiliza en el desarrollo de software para conectar objetos en un programa con tablas en una base de datos relacional.
Su propósito principal es permitir a los desarrolladores trabajar con datos de una base de datos relacional utilizando objetos y clases en lugar de escribir consultas SQL directamente. Un ORM mapea las estructuras de datos en una base de datos a objetos en un lenguaje de programación, lo que facilita la interacción con la base de datos y el manejo de datos de una manera más orientada a objetos. Las ventajas clave de usar un "*ORM*" incluyen:

- **Abstracción de la base de datos**: O^{o}culta los detalles de la base de datos subyacente, permitiendo a los desarrolladores interactuar con los datos en términos de objetos y clases en lugar de preocuparse por la sintaxis SQL específica de la base de datos.

- **Productividad**: Reduce la cantidad de código SQL que los desarrolladores deben escribir, lo que ahorra tiempo y esfuerzo en el desarrollo de aplicaciones.

- **Portabilidad**: Puede ser diseñado para ser independiente de la base de datos, lo que facilita la migración de una base de datos a otra sin cambiar el código de la aplicación.
- **Mantenibilidad**: Al trabajar con objetos y clases en lugar de consultas SQL directas, el código puede ser más fácil de mantener y entender.
- **Seguridad**: Suelen ofrecer características de seguridad integradas, como la prevención de inyecciones SQL, lo que reduce la vulnerabilidad de las aplicaciones a ataques.

Ejemplos de ORMs populares incluyen Hibernate para Java, Entity Framework para .NET, SQLAlchemy para Python, Sequelize para Node.js y Django ORM para aplicaciones web Django en Python.
Cada uno de estos ORMs proporciona una forma de mapear objetos a tablas de base de datos y facilita la manipulación de datos en aplicaciones de software de manera más eficiente y mantenible.

MVC [8]

MVC es un patrón arquitectónico comúnmente utilizado en el desarrollo de aplicaciones de software, especialmente en aplicaciones web.
Las siglas "*MVC*" representa "*Model-View-Controller*", y se divide en tres componentes principales que desempeñan roles específicos en la organización y estructura de una aplicación. Estos son:

- **Model (Modelo)**: El modelo representa la lógica empresarial y los datos subyacentes de la aplicación.
  En otras palabras, el modelo se encarga de manejar la manipulación y el acceso a los datos, así como de realizar cálculos y procesos necesarios.
  El modelo no tiene conocimiento de la interfaz de usuario y no se preocupa por cómo se muestran los datos.
  Es independiente de la vista y el controlador.
- **View (Vista)**: La vista se encarga de la presentación y la interfaz de usuario.
  Su función principal es mostrar los datos al usuario de una manera comprensible y atractiva.
  La vista no realiza operaciones comerciales o de lógica, sino que simplemente muestra la información proporcionada por el modelo.
  En sistemas web, esto a menudo se traduce en la generación de páginas

HTML o interfaces de usuario dinámicas. item **Controller (Controlador)**: El controlador actúa como un intermediario entre el modelo y la vista.

Recibe las solicitudes del usuario a través de la vista, procesa esas solicitudes, realiza las operaciones necesarias en el modelo y luego actualiza la vista para mostrar los resultados.
El controlador maneja la lógica de interacción y toma decisiones sobre qué acción realizar en función de la entrada del usuario.
La arquitectura MVC promueve una clara separación de preocupaciones, lo que facilita la modularidad y la escalabilidad de las aplicaciones. Esto permite a los desarrolladores trabajar de manera más eficiente, ya que pueden centrarse en una capa (modelo, vista o controlador) sin preocuparse demasiado por las otras capas. Además, el MVC es ampliamente utilizado en marcos de desarrollo web como Ruby on Rails, Django (Python), Laravel (PHP), y ASP.NET MVC (CSharp), entre otros. En resumen, el patrón MVC es una forma efectiva de organizar y estructurar aplicaciones de software, lo que resulta en un código más claro, mantenible y flexible.

LLM [17]

Un LLM (Modelo de Lenguaje Grande) es un tipo de modelo de inteligencia artificial diseñado para procesar y generar texto en lenguaje natural a gran escala.
Estos modelos están entrenados en enormes cantidades de datos de texto y están diseñados para comprender y generar texto en varios idiomas, lo que les permite llevar a cabo tareas de procesamiento de lenguaje natural de manera efectiva.
Algunos ejemplos son GPT-3, GPT-4 y BERT [18].
Características clave:

- **Tamaño y complejidad**: Los modelos de lenguaje grandes contienen miles de millones o incluso decenas de miles de millones de parámetros, lo que les permite capturar una gran cantidad de conocimiento lingüístico y patrones de lenguaje.
- **Pre-entrenamiento y afinación**: Estos modelos se entrenan primero en grandes corpus de texto, lo que les permite aprender el lenguaje natural y su estructura. Luego, se pueden afinar o ajustar para tareas específicas, como traducción automática, resumen de texto, generación de texto creativo, entre otros.

- **Generación de texto**: Son capaces de generar texto coherente y contextualmente relevante.
  Pueden responder preguntas, completar oraciones, generar contenido original y realizar tareas de procesamiento de lenguaje natural.

- **Amplia aplicabilidad**: Estos modelos se utilizan en una amplia gama de aplicaciones, que van desde asistentes virtuales y chatbots hasta motores de búsqueda mejorados, análisis de sentimientos, resumen automático de texto, traducción automática y más.

Plantean desafíos éticos y de seguridad, ya que su capacidad para generar texto implica la necesidad de prevenir el uso indebido, la desinformación y el sesgo en el contenido generado.
Estos modelos han sido desarrollados y perfeccionados por empresas de tecnología y organizaciones de investigación en inteligencia artificial, y han demostrado un gran potencial en una variedad de aplicaciones.
A medida que continúan evolucionando, su influencia en la forma en que interactuamos con la tecnología y generamos contenido en línea también está en constante crecimiento.

4.2. Herramientas

Python [16]

Es un lenguaje de programación de alto nivel, interpretado y generalmente considerado como fácil de aprender y de utilizar.
Python destaca por su sintaxis clara y fácil de entender, lo que lo convierte en un lenguaje ideal tanto para principiantes como para programadores experimentados.
Es compatible con una amplia gama de sistemas operativos, lo que lo hace muy portable.

Ofrece una extensa biblioteca estándar que cubre una variedad de tareas, desde manipulación de archivos y redes hasta desarrollo web y matemáticas, lo que acelera el desarrollo de aplicaciones.
Cuenta con una comunidad de desarrollo muy activa y una gran cantidad de bibliotecas y marcos de trabajo de código abierto disponibles, lo que facilita la creación de aplicaciones complejas.
Es orientado a objetos, lo que significa que se enfoca en la creación de objetos que pueden contener datos y funciones.

Se utiliza en una amplia variedad de campos, como desarrollo web, análisis de datos, inteligencia artificial, automatización de tareas, desarrollo de juegos y más.

Flask [6]

Flask es un framework de desarrollo web en Python que se utiliza para crear aplicaciones web de manera sencilla y rápida.
Está dentro de lo que se denomina "microframework" por su enfoque minimalista, proporciona las funcionalidades esenciales pero permite agregar las bibliotecas necesarias según los requisitos.
No impone estructuras rígidas ni componentes innecesarios pero incorpora funcionalidades (como el motor de plantillas Jinja2 o el sistema de rutas) que hacen más sencillo el desarrollo de aplicaciones.
En este proyecto lo utilizamos para realizar un API RESTful, que es un uso bastante común de este framework.

SqlAlchemy [2]

SQLAlchemy es una biblioteca de Python que proporciona un conjunto de herramientas y un ORM(4.1) que permite a los desarrolladores interactuar con bases de datos relacionales de una manera más sencilla y orientada a objetos.

Ofrece una amplia gama de métodos y operadores para realizar consultas en la base de datos, lo que facilita la construcción de consultas complejas de manera programática.

Gestiona las transacciones de manera transparente, lo que asegura la integridad de los datos al realizar operaciones de lectura y escritura.
Se pueden definir relaciones entre objetos y tablas de base de datos de manera sencilla, lo que simplifica la representación de datos relacionados.
Proporciona herramientas para realizar migraciones de bases de datos, lo que facilita la actualización de la estructura de la base de datos a medida que evoluciona la aplicación.
Permite la creación de extensiones y complementos personalizados para adaptarse a necesidades específicas.
Se utiliza principalmente en aplicaciones web, especialmente en el desarrollo de aplicaciones basadas en frameworks como Flask y Django.
Permite a los desarrolladores trabajar con bases de datos de manera más eficiente y mantener un código más limpio y legible al proporcionar una

capa de abstracción entre la aplicación y la base de datos subyacente.
Esto también facilita la portabilidad de la aplicación entre diferentes sistemas de gestión de bases de datos.

LlamaCpp [1]

La división de inteligencia artificial de Meta, llamada Meta AI, presentó LLaMA, un LLM (4.1) de 65.000 millones de parámetros que permite disfrutar de un motor de IA conversacional muy parecido a ChatGPT.
Este modelo estaba inicialmente disponible para desarrolladores e investigadores que justificaran su uso.
Grigori Gerganov publicó en Github [10] un pequeño desarrollo llamado llama.cpp, un proyecto que permite poder usar el modelo LLaMA en un portátil o un PC convencional.
Eso se logra gracias a la llamada çuantización"que reduce el tamaño de los modelos de Facebook para hacerlos "manejables"por equipos más modestos a nivel de hardware.

PostgreSql [12]

PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto. Admite consultas complejas y procedimientos almacenados. Proporciona mecanismos para garantizar la integridad de los datos almacenados, como restricciones de clave primaria y foránea, así como comprobaciones de restricciones.
Es escalable y se puede utilizar en aplicaciones de todos los tamaños, desde aplicaciones pequeñas hasta sistemas empresariales de alto rendimiento.
Admite la replicación y puede configurarse para lograr alta disponibilidad mediante la implementación de réplicas y clústeres. Soporta autenticación, autorización y cifrado de datos.
PostgreSQL es una opción popular tanto en la comunidad de código abierto como en empresas que buscan una base de datos de alto rendimiento y confiabilidad para sus aplicaciones.
Su licencia de código abierto permite su uso y distribución sin costos de licencia, lo que lo convierte en una opción atractiva para una amplia variedad de proyectos.

Docker [9]

Docker es una plataforma que permite empaquetar una aplicación y todas sus dependencias en un contenedor.
Este contenedor es como una unidad autónoma que puede ejecutarse de manera consistente en cualquier entorno que admita Docker. Además, Docker facilita la gestión y la implementación de estas aplicaciones empaquetadas. Al utilizar Docker, los desarrolladores pueden estar seguros de que la aplicación se ejecutará de la misma manera en todas partes, desde el entorno de desarrollo hasta el de producción. Esto elimina problemas relacionados con las diferencias entre configuraciones y facilita las pruebas. Se pueden iniciar y detener rápidamente, lo que permite una implementación más rápida de nuevas versiones o características.
Esto es crucial en el *desarrollo ágil*, donde la velocidad de respuesta a los cambios es esencial.

Además encapsula todas las dependencias de la aplicación en el contenedor, evitando conflictos con otras aplicaciones o componentes del sistema. Esto simplifica la gestión de dependencias y reduce los posibles problemas de compatibilidad.
También facilita la escalabilidad horizontal, lo que significa que puedes ejecutar múltiples instancias de contenedores de la misma aplicación para gestionar cargas de trabajo más pesadas de manera eficiente.

Los contenedores Docker pueden compartirse fácilmente, lo que facilita la colaboración entre equipos de desarrollo y operaciones. Todos trabajan con la misma configuración, lo que reduce la posibilidad de errores causados por diferencias en los entornos.

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.

5.1. Metodologías

Metodologías ágiles(SCRUM)

Scrum es un marco de trabajo ágil utilizado comúnmente en el desarrollo de software, aunque se ha extendido a otras áreas.
Proporciona un enfoque estructurado para la gestión de proyectos que se centra en la entrega iterativa y incremental de productos.

Scrum se basa en los principios del manifiesto ágil y busca mejorar la eficiencia, la flexibilidad y la colaboración en equipos de desarrollo.

Roles

- **Scrum Master**: Facilita el proceso Scrum, elimina obstáculos y ayuda al equipo a alcanzar sus objetivos.
- **Product Owner**: Representa las necesidades del cliente y define las características del producto.
- **Equipo de Desarrollo**: Profesionales que trabajan en la entrega del producto.

Eventos

- **Sprint**: Un periodo de tiempo fijo (generalmente de 2 a 4 semanas) en el que se entrega un incremento de producto.
- **Reunión de Planificación del Sprint**: Al inicio de cada sprint, el equipo planifica el trabajo que se realizará durante ese periodo.
- **Revisión del Sprint**: Al final de cada sprint, el equipo presenta el trabajo completado al Product Owner y a otras partes interesadas.
- **Retrospectiva del Sprint**: Una sesión al final de cada sprint donde el equipo revisa su desempeño y busca formas de mejorar.

Artefactos

- **Product Backlog**: Una lista priorizada de todas las funcionalidades, cambios y mejoras propuestas para el producto.
- **Sprint Backlog**: La lista de tareas que el equipo se compromete a completar durante un sprint.
- **Incremento**: El producto funcional y potencialmente entregable al final de cada sprint.

Scrum promueve la transparencia, la inspección y la adaptación, lo que significa que los equipos pueden ajustar su enfoque y estrategia en función de los cambios en los requisitos del cliente o en las circunstancias del proyecto.

Este marco de trabajo es especialmente útil en entornos donde los requisitos pueden cambiar con frecuencia y se valora la capacidad de respuesta y flexibilidad del equipo de desarrollo.

Para este proyecto se ha seguido la metodología **SCRUM** configurando los sprints con una duración de 2 semanas, para la planificación se ha utilizado la herramienta Zube. Se puede encontrar más detalles sobre el proceso seguido en los anexos de este documento.

DevOps

DevOps es una cultura, filosofía y conjunto de prácticas que se centran en la colaboración estrecha y la integración continua entre los equipos de desarrollo (Dev) y operaciones (Ops) en el ciclo de vida del desarrollo de software.
El objetivo principal de **DevOps** es acelerar la entrega de software, mejorar la calidad y la confiabilidad, y permitir una respuesta rápida a los cambios y a las necesidades de los usuarios.
DevOps promueve la automatización, la comunicación eficaz y la colaboración entre los equipos, lo que permite un flujo de trabajo más eficiente en todo el proceso de desarrollo y entrega de software.
Algunos de los principios y prácticas clave de **DevOps** incluyen:

- **Automatización**: La automatización de tareas repetitivas y procesos manuales acelera la entrega y minimiza los errores. Esto incluye la automatización de pruebas, implementaciones, aprovisionamiento de infraestructura y monitoreo.

- **Integración Continua (CI)**: Los cambios de código se integran regularmente en un repositorio compartido, se prueban automáticamente y se implementan en un entorno de desarrollo o de prueba. Esto asegura que el código esté siempre en un estado funcional.

- **Entrega Continua (CD)**: La entrega continua extiende la integración continua al permitir la entrega automática de cambios a entornos de prueba o producción después de la integración y las pruebas exitosas.

- **Monitoreo y Retroalimentación Continua**: El monitoreo constante de aplicaciones y sistemas permite detectar problemas en tiempo real y proporciona información valiosa para mejorar la calidad y la eficiencia.

- **Colaboración y Comunicación**: La comunicación efectiva entre los equipos de desarrollo y operaciones es esencial. La colaboración se fomenta mediante reuniones regulares, herramientas compartidas y un entendimiento mutuo de las metas y responsabilidades.
- **Infraestructura como Código (IaC)**: La infraestructura se administra y despliega como código, lo que facilita la creación y el mantenimiento de entornos de desarrollo y producción consistentes y escalables.
- **Cultura de Mejora Continua**: DevOps promueve una cultura en la que se aprende de los errores y se busca constantemente la mejora en todos los aspectos del desarrollo y la operación de software.
- **Seguridad**: La seguridad es un aspecto crítico de **DevOps**. Los principios de seguridad deben estar integrados en todas las etapas del ciclo de vida del desarrollo de software.

La implementación exitosa de **DevOps** puede llevar a una mayor velocidad de entrega, una mayor calidad del software, una mayor eficiencia operativa y una mayor capacidad de respuesta a los cambios del mercado y las necesidades del cliente. Esta metodología se ha vuelto esencial en el desarrollo de software moderno, especialmente en entornos ágiles y de entrega continua.

Para el publicar del código de este trabajo se ha utilizado Docker junto con Portainer para el despliegue continuo.

5.2. Desarrollo del proyecto

El principal escollo que me he encontrado para la realización de este proyecto es la elección del modelo LLM adecuado.
Por un lado tenemos el servicio de OpenAI. Este servicio es quizá el más estable y que mejores resultados obtiene.
Sin embargo este servicio es de pago (aunque propocionan un saldo gratuito al crear una cuenta, no permite realizar más de 3 llamadas al minuto). Además tenemos el tiempo de proceso para cada reseña (unos 3 segundos). Existe otros LLMs disponibles como por ejemplo Llama2 (Meta), Bard(Google).
Con el modelo de Meta existe la posibilidad de ejecutarlo en un entorno local mediante diferentes librerías (LlamaCpp).
Otros modelos que se pueden ejecutar en local son GPT4All. Una herramienta muy útil para elegir el modelo adecuado es LM Studio, permite descargar y probar distintos modelos, además ofrece una API compatible con OpenAI.

Para la ejecución en local es muy recomendable el uso de una tarjeta gráfica para optimizar la ejecución del modelo.

Al planificar el segundo sprint incluí una tarea para crear un prototipo en collab, sin embargo esta tarea tenía unas dependencias que hacían imposible su realización. Lo primero que tenía que hacer era seleccionar un dataset para poder realizar las pruebas. Y además necesitaba seleccionar el modelo LLM idoneo para la realización de este protoripo.

5.3. Elección del conjunto de datos

Elegir un conjunto de datos adecuado es crucial para el análisis de sentimientos. En este proyecto nos insteresa realizar un análisis de sentimientos en reseñas de Amazon por lo que se ha buscado con esa premisa.
Esto reduce la búsqueda pero aun así hay que encontrar un conjunto de datos que tenga una serie de características que lo haga ideal para su uso en el proyecto.
Por un lado, como veremos más adelante, el análisis de sentimiento tiene un coste, ya sea económico (OpenAI), o en tiempo (LlamaCpp). Esto implica que tenemos que elegir un dataset de un tamaño adecuado. Por un lado, tiene que ser suficientemente grande para que los resultados sean significativos y por otro lado lo suficientemente pequeño como para que el coste sea adecuado al proyecto. El conjunto de datos debe contener información suficiente como para poder realizar comparaciones bajo distintas circustancias de estudio, por usuario, por producto, etc.
También sería interesante que el conjunto de datos sea represativo de la población, para ello lo ideal sería obtener el conjunto de datos de diferentes orígenes.
En nuestro caso el origen es siempre Amazon pero a la hora de seleccionar los datos intentaremos que estos sean lo más variados posibles.
Además hay tener en cuenta la disponibilidad, en algunos casos Amazon ha decidido anular la licencia de uso con lo que esos conjuntos de datos no se podrían usar.
En base a estas premisas se han estudiado datasets de diferentes orígenes:

Universidad de California [11]

La Universidad de California proporciona un conjunto de datos con reseñas de Amazon. Los más llamativo de este conjunto de datos es su tamaño (230 millones de reseñas) y la cantidad de metadatos incluidos.

También propocionan ejemplos de uso y de código así como un cuaderno en Colab.

Amazon Reviews Dataset

Kaggle

Kaggle es una plataforma en línea que ofrece una variedad de recursos relacionados con la ciencia de datos y el aprendizaje automático. Fue fundada en 2010 y adquirida por Google en 2017. Kaggle proporciona un entorno donde los científicos de datos, los investigadores y los entusiastas del aprendizaje automático pueden colaborar, compartir conocimientos, participar en competiciones de ciencia de datos y acceder a conjuntos de datos. Uno de los conjuntos de datos evaluados han sido:

- Amazon Reviews for Sentiment Analysis

  Este conjuntos de datos tiene un tamaño adecuado para nuestro proposito pero le faltan metadatos y no nos permitiría desarrollar toda la funcionalidad esperada en la aplicación.

- Amazon Reviews

Hugging Face

Hugging Face es una empresa y plataforma en línea que se centra en la creación y distribución de modelos de lenguaje de inteligencia artificial, así como en herramientas y recursos relacionados con el procesamiento del lenguaje natural (NLP). La compañía es conocida por su biblioteca de modelos preentrenados, así como por su contribución a la comunidad de aprendizaje automático y NLP. Existen muchos conjuntos de datos que podemos usar para nuestro proyecto, por ejemplo:

- Amazon Shoe Reviews
- Amazon Video Games Review
- Amazon Reviews for Sentiment Analysis

5.4. Elección del modelo LLM

Para la elección del modelo a usar en este trabajo, primero examinamos el modelo más sencillo de utilizar. Este es el modelo de OpenAI [19]. En

concreto el modelo **ChatGpt 3.5 Turbo**, este modelo es muy estable, rápido y económico. El problema de usar este modelo, a pesar de ser económico, es su coste. Para evitar ese problema y poder utilizar un modelo sin limitaciones, he buscado otras alternativas. Entre estas se encuentran:

- **Microsoft Azure Language Models** Microsoft Azure ofrece varios servicios de procesamiento del lenguaje natural, incluidos modelos de lenguaje como parte de su oferta de servicios cognitivos.
- **Google Cloud Natural Language API** Google Cloud proporciona una API de procesamiento del lenguaje natural que incluye funciones avanzadas como análisis de sentimientos, extracción de entidades y más.
- **Hugging Face Transformers** Hugging Face es una plataforma que ofrece acceso a una amplia variedad de modelos de lenguaje preentrenados. Los modelos GPT y otros están disponibles a través de su biblioteca Transformers.
- **Facebook AI** Facebook AI Research (FAIR) trabaja en el desarrollo de modelos de lenguaje y herramientas relacionadas con la inteligencia artificial. Aunque no tienen un modelo específico comparable a GPT, han contribuido significativamente al campo.
- **Rasa** Rasa es una plataforma de código abierto para construir asistentes conversacionales. Permite a los desarrolladores crear chatbots personalizados y asistentes virtuales.
- **Chatbot Frameworks** Hay varios marcos y bibliotecas de código abierto que permiten a los desarrolladores construir sus propios chatbots, como ChatterBot, Botpress, y Microsoft Bot Framework.
- **Dialogflow** Dialogflow, propiedad de Google, es una plataforma de desarrollo de chatbots y asistentes virtuales que utiliza tecnologías de procesamiento del lenguaje natural.

Investigando sobre las diferentes alternativas nos hemos encontrado con dificultades a la hora de elegir la idónea. Por un lado, están los modelos que también son de pago como son “Microsoft Azure Language Models” y Google Cloud Natural Language API. Ambas soluciones ofrecen periodos de prueba limitados. Por otro lado están las plataformas para asistentes conversacionales como Dialogflow, Rasa, etc que no ofrecen un API para poder usarlas en un programa. Por último nos queda **Facebook API**, en

este caso su modelo Llama2 está disponible para todo tipo de usos. Existen una colección de modelos que van desde 7b a 70b (7b significa que se han usado 7.000 millones de tokens para su entrenamiento y 70b 70.000 millones). Aún así el modelo más pequeño necesita unos recursos muy elevados para su funcionamiento. Aquí es dónde entra en juego la cuantificación.
Aplicando la cuantificación, los modelos ocupan menos espacio y son más rápidos pero también más imprecisos. En la plataforma Hugging Face existen multitud de modelos de diferentes tamaños y tipos de cuantificación. En este punto nos quedaba elegir un modelo que se pueda ejecutar de una manera razonable en equipos personales pero manteniendo unos niveles razonables de precisión.
Así entra en juego el último de los elementos para poder elegir un modelo y es un método sencillo para poder evaluar los modelos.
Para ello he usado una aplicación llamada LM Studio.
Es una aplicación gratuita que permite descargar modelos de Hugging Face y probarlos mediante un chat o mediante una API.
Además permite aprovechar una tarjeta gráfica en el caso de tenerla para poder usar el modelo aprovechando sus capacidades de computación en paralelo.

6. Trabajos relacionados

6.1. BERT

BERT [5], que significa "*Bidirectional Encoder Representations from Transformers*" (Representaciones de Codificador Bidireccional a partir de Transformadores), es un modelo de lenguaje desarrollado por Google en 2018.
BERT es parte de la familia de modelos de lenguaje basados en transformers y es ampliamente conocido por su capacidad para comprender el contexto de las palabras en un texto, lo que lo convierte en una herramienta poderosa en el análisis de sentimiento y muchas otras tareas de procesamiento del lenguaje natural (NLP).
Lo que hace que BERT sea particularmente efectivo en el análisis de sentimiento es su capacidad para entender la semántica y el significado contextual de las palabras en un texto. A diferencia de los modelos de lenguaje unidireccionales anteriores, BERT es bidireccional, lo que significa que procesa el texto de izquierda a derecha y de derecha a izquierda en dos pasadas separadas.
Esto le permite capturar el contexto de una palabra en función de las palabras que la rodean en todas las direcciones, lo que mejora significativamente su comprensión del lenguaje natural.
En el análisis de sentimiento, BERT se utiliza para evaluar el tono o polaridad del texto, es decir, determinar si el texto tiene un sentimiento positivo, negativo o neutral.
Para hacerlo, se entrena a BERT [18] en grandes conjuntos de datos etiquetados con sentimientos, lo que le permite aprender a identificar patrones y contexto asociados con emociones específicas en el lenguaje.
BERT ha sido un avance significativo en el campo del procesamiento del

lenguaje natural y ha mejorado considerablemente el rendimiento en una amplia gama de aplicaciones, incluido el análisis de sentimiento, la traducción automática, la respuesta a preguntas y muchas otras tareas de NLP.

6.2. Transformers

Los ´´*Transformers*" son una arquitectura de modelos de lenguaje y procesamiento del lenguaje natural (NLP) que se ha convertido en un avance revolucionario en este campo desde su introducción en 2017.
Fueron desarrollados por Vaswani et al. en el artículo ´´*Attention Is All You Need*" [20] y han tenido un gran impacto en tareas de NLP, incluyendo la traducción automática, el análisis de sentimiento, la generación de texto, etc.
La característica principal de los ´´*Transformers*" es su mecanismo de atención, que permite capturar las relaciones entre las palabras en un texto de manera más efectiva que las arquitecturas de modelos de lenguaje anteriores. El mecanismo de atención permite a un modelo dar más peso a ciertas partes del texto cuando procesa una palabra o token en particular, lo que le permite comprender mejor el contexto y las dependencias entre las palabras. Consisten en múltiples capas apiladas de unidades de atención, lo que les permite aprender representaciones jerárquicas de los datos.
A diferencia de otros de modelos de lenguaje, no utilizan capas convolucionales, lo que los hace altamente paralelizables y eficientes para el entrenamiento. Se utilizan en muchas aplicaciones de procesamiento del lenguaje natural, como traducción automática, generación de texto, análisis de sentimiento [7], respuesta a preguntas, resumen de texto, etc. Ejemplos de modelos Transformer populares incluyen BERT [5], GPT [19] (Generative Pre-trained Transformer), RoBERTa [14], y muchos otros.
Estos modelos han demostrado un rendimiento sobresaliente en diversas tareas de procesamiento del lenguaje natural y han contribuido significativamente al avance en este campo.

6.3. NLTK

NLTK [3] es una biblioteca de código abierto en el lenguaje de programación Python que proporciona herramientas, recursos y bibliotecas para trabajar con el procesamiento de lenguaje natural (NLP).
Fue desarrollada originalmente por Steven Bird y Edward Loper en la Universidad de Pensilvania. NLTK es ampliamente utilizado en la comunidad

de NLP y es una herramienta esencial para investigadores, estudiantes y desarrolladores que trabajan en tareas relacionadas con el procesamiento del lenguaje natural.
NLTK proporciona una variedad de herramientas para tokenizar, segmentar, etiquetar y analizar texto. Incluye una amplia gama de recursos lingüísticos, como corpus (conjuntos de datos de texto etiquetado), léxicos y otros datos que son útiles para la investigación y desarrollo en NLP.
Ofrece módulos y funciones para llevar a cabo tareas como análisis sintáctico, análisis semántico, desambiguación de sentidos, análisis de sentimiento y más.
NLTK se integra con otras bibliotecas y recursos, lo que facilita la conexión con motores de búsqueda, bases de datos y otras herramientas de procesamiento del lenguaje natural.
NLTK también incluye funcionalidades para el aprendizaje automático en NLP, lo que permite a los usuarios desarrollar modelos de lenguaje y clasificadores.
NLTK es muy útil para quienes deseen aprender sobre procesamiento de lenguaje natural, ya que proporciona una base sólida y muchas herramientas prácticas.
Además, es una excelente opción para prototipar y desarrollar soluciones de NLP en Python [13].

6.4. Scikit-learn

Scikit-learn es una biblioteca de aprendizaje automático en Python que se utiliza ampliamente en una variedad de aplicaciones, incluido el análisis de sentimiento.
Sin embargo, en el contexto específico del análisis de sentimiento, scikit-learn se utiliza más como una herramienta para la construcción de modelos de clasificación y evaluación de rendimiento que para el procesamiento del lenguaje natural en sí.
En el análisis de sentimiento, scikit-learn [15] se utiliza para extracción de textos, construcción de modelos (clasificación, SVM, Naive Bayes, Regresión, etc) y evaluación de rendimiento (precisión, recall, F1-score y matriz de confusión).

En resumen, scikit-learn es una herramienta valiosa en el análisis de sentimiento, ya que facilita la construcción y evaluación de modelos de clasificación para determinar el sentimiento en textos.
Sin embargo, para tareas de procesamiento del lenguaje natural más complejas, como la comprensión del contexto y la semántica del texto, a menudo

se combinan otras bibliotecas y modelos NLP, como spaCy, transformers (como BERT o GPT), y NLTK, con scikit-learn para lograr u n rendimiento óptimo en el análisis de sentimiento.

6.5. Sentinel

Sentinel [4] es una aplicación web desarrollada como TFG del grado de ingeniería informática de la Universidad de Burgos que realiza un análisis de sentimientos de textos extraídos de las redes sociales Twitter e Instagram.
La aplicación busca los tweets relacionados con la palabra introducida en el caso de Twitter, o los comentarios que le han escrito a la cuenta de Instagram que se ha buscado.
Después analiza el sentimiento que hay en ellos, los puntúa con valores entre 0 y 1 y se almacenan los resultados.
Estos resultados se muestran al usuario en gráficos y tablas para hacer la experiencia más visual.
Además se le ofrece la opción de calcular series temporales a partir de los resultados, y se da una predicción de los valores futuros.

6.6. NLP

NLP [21], o Procesamiento del Lenguaje Natural (por sus siglas en inglés, Natural Language Processing), es un campo de la inteligencia artificial y la informática que se enfoca en la interacción entre las computadoras y el lenguaje humano.
Su objetivo es permitir que las máquinas comprendan, interpreten y generen texto y lenguaje humano de manera similar a como lo hacen los seres humanos.
Algunos puntos clave sobre NLP son:

Comunicación entre humanos y máquinas: NLP se centra en facilitar la comunicación efectiva entre humanos y computadoras a través del lenguaje natural. Esto incluye la comprensión del texto escrito y hablado, así como la generación de texto legible y coherente por parte de las máquinas.

Tareas en NLP: NLP abarca una amplia gama de tareas y aplicaciones, que incluyen la traducción automática, el análisis de sentimiento, el análisis de texto, la extracción de información, la respuesta a preguntas,

el resumen de texto, el procesamiento de voz, el procesamiento de chatbots y mucho más.

Desafíos en NLP: El procesamiento del lenguaje natural es un campo desafiante debido a la complejidad del lenguaje humano, que incluye ambigüedades, variabilidad, coloquialismos y otros factores. La comprensión del contexto, la semántica y la cultura son aspectos fundamentales.

Herramientas y tecnologías: En NLP, se utilizan una variedad de herramientas y tecnologías, como bibliotecas de código abierto (como NLTK, spaCy y scikit-learn en Python), modelos de lenguaje (como BERT y GPT-3), algoritmos de aprendizaje automático y redes neuronales, para abordar tareas específicas.

Aplicaciones en la vida cotidiana: NLP tiene un impacto significativo en la vida cotidiana, ya que se utiliza en motores de búsqueda, asistentes virtuales (como Siri y Alexa), traducción automática, análisis de redes sociales, detección de spam, corrección ortográfica, entre muchas otras aplicaciones.

Avances recientes: En los últimos años, los modelos de lenguaje basados en transformers, como BERT y GPT-3, han logrado avances significativos en tareas de NLP, mejorando la capacidad de las máquinas para entender y generar texto de manera más precisa y coherente.

El procesamiento del lenguaje natural es un campo en constante evolución con un gran potencial para transformar la forma en que interactuamos con la tecnología y procesamos grandes cantidades de datos de texto.
A medida que se desarrollan nuevas técnicas y modelos, el NLP seguirá siendo una área de investigación y desarrollo crucial en la inteligencia artificial.

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.

7.1. Conclusiones

7.2. Líneas de trabajo futuras

Nuevos modelos

Nuevas fuentes de datos

Web service

Más opciones de análisis

Configuración del prompt

GridFS

Bibliografía

[1] Meta AI. Llama: open and efficient foundation language models.

[2] Michael Bayer, A Brown, and G Wilson. Sqlalchemy. *The architecture of open source applications*, 2:291–314, 2012.

[3] Edward Loper Bird, Steven and Ewan Klein. *Natural Language Processing with Python*. O'Reilly Media Inc., 2009.

[4] Zoe Calvo. Tfg. TFG Universidad de Burgos, 2020.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[6] Gareth Dwyer, Shalabh Aggarwal, and Jack Stouffer. *Flask: building python web services*. Packt Publishing, 2017.

[7] Lino Alberto Urdaneta Fernández. Análisis de sentimientos en español con seis líneas de código. Medium, 2021.

[8] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2012.

[9] Javier Garzás. ¿qué es docker. https://www.javiergarzas.com/2015/07/que-es-docker-sencillo.html, 2015.

[10] Grigori Gerganov. Llama.cpp.

[11] Julian McAuley Jianmo Ni, Jiacheng Li. Justifying recommendations using distantly-labeled reviews and fine-grained aspects.

[12] Salahaldin Juba, Achim Vannahme, and Andrey Volkov. *Learning PostgreSQL*. Packt Publishing Ltd, 2015.

[13] Angelica Landazabal. Realiza un análisis de sentimiento en 3 pasos con python. Platzi, 2020.

[14] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[15] Perfecto Vidal Lloret. Análisis de sentimientos con scikit-learn. http://elcodigoperfecto.blog, 2023.

[16] Mark Lutz. *Programming python*. .º'Reilly Media, Inc.", 2001.

[17] Suvir Mirchandani, Fei Xia, Pete Florence, Brian Ichter, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. Large language models as general pattern machines. *arXiv preprint arXiv:2307.04721*, 2023.

[18] nlptown. Modelo especializado en analizar sentimientos en comentarios de usuario. Hugging Face, 2023.

[19] OpenAI. Chat gpt, 2023.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[21] Mrinal Walia. Los 5 mejores (y desconocidos) proyectos de análisis de sentimientos en github para proyectos de nlp, 2021.

[22] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. A prompt pattern catalog to enhance prompt engineering with chatgpt, 2023.