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 — 29 de enero de 2024
Tutores: Dª Virginia Ahedo García y D. Jose Ignacio Santos Martín

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

Dª Virginia Ahedo García y D. Jose Ignacio Santos Martín, profesores del departamento de Ingeniería de Organización.

Exponen:

Que el alumno D. Teodoro Ricardo García Sánchez, con DNI 50450469S, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado **Aplicación de análisis de reseñas basado en modelos de lenguaje grandes mediante prompt engineering.**.

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

En Burgos, 29 de enero de 2024

Vº. Bº. del Tutor:

Firmado por AHEDO GARCIA, VIRGINIA (FIRMA) el día 14/02/2024 con un certificado emitido por AC DNIE 005

Dra. Virginia Ahedo García

Vº. Bº. del co-tutor:

SANTOS MARTIN JOSE IGNACIO - 09331511C

Firmado digitalmente por SANTOS MARTIN JOSE IGNACIO - 09331511C
Fecha: 2024.02.14 09:58:35 +01'00'

Dr. José Ignacio Santos Martín

Resumen

Las reseñas ofrecen una valiosa información sobre la percepción de los usuarios respecto a productos y servicios.

Por ello, se ha desarrollado una aplicación que utiliza LLM (mediante "prompts") para el análisis, clasificación y calificación de reseñas de usuarios.

Utilizamos el conocimiento de estos modelos preentrenados para diseñar e implementar de forma ágil una solución a este problema.

Descriptores

Análisis de sentimiento, prompt engineering, flask, python, vue, chatgpt, api rest.

Abstract

Reviews offer a valuable information about products and services user's perception.

That's why we have devoloped an application using prompts to LLMs for the sentiment análisis and classification of user's reviews.

We use this pretrained models knowlegde to design and develope an agile solution to this problem.

Keywords

Sentiment analysis, prompt engineering, flask, python, vue, chatgpt, api rest.

Índice general

Índice general iii

Índice de figuras v

Índice de tablas vi

1. Introducción 1
- 1.1. Estructura de la memoria . . . 2
- 1.2. Estructura de los anexos . . . 2

2. Objetivos del proyecto 3

3. Conceptos teóricos 5
- 3.1. Análisis de sentimiento . . . 5
- 3.2. Procesamiento del Lenguaje natural . . . 6
- 3.3. Prompt Engineering . . . 7
- 3.4. Patrones de diseño . . . 8

4. Técnicas y herramientas 11
- 4.1. Técnicas . . . 11
- 4.2. Herramientas . . . 14

5. Aspectos relevantes del desarrollo del proyecto 19
- 5.1. Metodologías . . . 19
- 5.2. Desarrollo del proyecto . . . 22
- 5.3. Elección del conjunto de datos . . . 22
- 5.4. Elección del modelo LLM . . . 24
- 5.5. Diseño del prompt . . . 26

5.6. Validación del sistema 27
5.7. Uso de herramientas 29

6. Trabajos relacionados **31**
6.1. Lingmotif [17] 31
6.2. Lexalytics [13] 32
6.3. IBM Watson Studio [10] 33
6.4. Sentinel [5] 34
6.5. Free Sentiment Analysis [22] 34

7. Conclusiones y Líneas de trabajo futuras **35**
7.1. Conclusiones 35
7.2. Líneas de trabajo futuras 36

Bibliografía **37**

Índice de figuras

5.1. Matriz de confusión Vader sentiment 28
5.2. Resultados Vader Sentiment 28
5.3. Matriz de confusión Prompt Sentiment 29
5.4. Resultados Prompt Sentiment 29
6.1. Lingmotif 31
6.2. Lexalytics análisis de texto 32
6.3. IBM Watson Studio 33

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 o negativa). Existen muchas formas de llevar a cabo este análisis (localización de palabras, afinidad léxica, métodos estadísticos o tecnicas conceptuales) [2]

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.

Utilizaremos un dataset etiquetado de opiniones de clientes de Amazon para evaluar el rendimiento del modelo. En particular, usaremos 2 medidas, por un lado la precisión que mide el ratio de verdaderos positivos entre todos los que se han clasificado como tal. y la exhaustividad ("*recall*") que mide la relación entre los verdaderos positivos y todos los positivos.

Se considera un modelo equiparable a un humano si consigue un 65 por ciento de aciertos. [21, p. 4]

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 [15]) 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.

- 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*" [24]:

  - 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 [4].

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 [16].
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 [23].

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 [24]:

- **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 [7].
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 [7]

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**: oculta 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 [7]

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 la separación de intereses (“ *'separation of concerns'*), 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. Es una forma efectiva de organizar y estructurar aplicaciones de software, lo que resulta en un código más claro, mantenible y flexible.

LLM [15]

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 [19].
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 [14]

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 una API RESTful, que es un uso bastante común de este "*framework*".

SQLAlchemy [3]

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 [9] 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 "*cuantización*" 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 [8]

"*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.

ChatGPT [20]

"*ChatGPT*" es un modelo de lenguaje desarrollado por OpenAI, basado en la arquitectura GPT (Generative Pre-trained Transformer).

La arquitectura GPT es una red neuronal de tipo transformer que ha sido preentrenada en grandes cantidades de datos textuales para aprender patrones y estructuras del lenguaje.

"*ChatGPT*" puede generar respuestas contextualmente relevantes en función de las entradas que recibe. Puede ser utilizado para una variedad de aplicaciones, desde responder preguntas y ayudar en la redacción de textos.

Es importante tener en cuenta que "*ChatGPT*" no posee conocimiento del mundo en tiempo real ni tiene la capacidad de razonamiento ni comprensión como lo haría un ser humano.

Durante la realización del trabajo fin de grado he usado "*ChatGpt*" para obtener información y sugerencias siempre contrastándolas con otras fuentes para comprobar su corrección.

Es especialmente interesante su uso para hacer resúmenes y organizar la información de forma coherente.

5. Aspectos relevantes del desarrollo del proyecto

Este apartado recoge los aspectos más interesantes del desarrollo del trabajo.

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 [18]

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** son:

- **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 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 y Github 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 idóneo para la realización de este protoripo. Una vez replanificadas las tareas en el orden adecuado, seleccioné un conjunto de datos de huggingface y un modelo LLM basado en LlamaCpp para usarlo en el notebook de collab. Al usar un modelo basado en LlamaCpp la ejecución resulta más lenta (unos 30 segundos por reseña) pero no se infiere en costes de proceso.

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

Finalmente se ha utilizado un conjunto de datos de huggingface, ya que contiente más metadatos que han resultado útiles para la organización de las pruebas (Amazon Shoe Reviews). Por otro lado se han preparado una serie de ficheros con reseñas para poder hacer la demostración. Tanto el fichero python para preparar esos fichero como los ficheros en sí, se pueden encontrar en la carpeta datasets del repositorio de código.

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 [20]. 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** h^{o}ay 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.

Después de evaluar los diferentes modelos LLM y en vista de los tiempos de respuesta se ha decidido usar el modelo de OpenAI para la aplicación final. Aunque se puede usar un modelo basado en LlamaCpp de forma opcional.

5.5. Diseño del prompt

Una vez elegido el modelo más adecuado para nuestras aplicación nos toca diseñar un "prompt" que funcione correctamente para nuestro proposito. Una de las características que tiene que tener un "prompt" es que sea preciso. [24] Tiene que especificar de forma precisa lo que esperas recibir. En nuestro caso tenemos que incluir unos delimitadores para que el modelo sea capaz de identificar cuál es el texto al que nos referimos. También tenemos que especificar cuál tiene que ser el formato de salida y los posibles valores que puede tener cada variable.

```
template_string ="""Return a JSON with the following \
                    information extracted from the review \
                    below: \
 {{ \
     ""Sentiment"": ""(positive or negative)"",\
     ""Stars"": ""Number of stars depending on the \
                 sentiment of the Review"",\
     ""Anger"": ""Is the user angry? (true or false)"", \
 }} \
Review: ‘‘‘{review}‘‘‘ \
If the information isn't present, use ""unknown"" \
as the value. \
Remember to return only the JSON.
"""
```

Por último hay que especificar qué hacer en el caso de que no encuentre la información solicitada.

Tal y como funcionan los "LLMs" la salida no es siempre correcta. Entre los problemas que nos podemos encontrar tenemos: Al ser modelos generativos, pueden devolver el texto de entrada y después seguir generando la respuesta. Salida incorrecta, en nuestro caso un JSON mal formado, ya sea por comillas por defecto o exceso, valores incorrectos, etc. En ambos casos he intentado paliar esos errores en la medida de lo posible con código adicional.

Hay que decir que los "prompts" están muy vinculados a cada tipo modelo y no funcionan en cualquier modelo por igual.

También la salida recibida no es siempre igual y por eso hay que hacer algunos ajustes para obtener el mejor resultado posible. Existe una librería llamada langchain que intenta facilitar este proceso. Para ello se definen una serie de "ResponseSchemas" para especificar cada variable de salida:

```
sentiment_schema = ResponseSchema(name="Sentiment",
description="It's the sentiment of the review
(positive or negative)")
```

De esta forma incluye la información de cada variable de salida al prompt y luego recupera su valor en la salida. Esta librería está en pleno desarrollo y en la pruebas realizadas no ha dado unos resultados satisfactorios con algunos de los modelos probados. Los mejores resultados han sido con OpenAI Aunque creo que esta será la mejor forma de construir una aplicación para interaccionar con diferentes modelos cuando su desarrollo está más avanzado.

5.6. Validación del sistema

Uno de los objetivos de este trabajo es el de comprobar si es factible usar esta aproximación para extraer los sentimientos de las reseñas de usuarios de un determinado producto o servicio. Para comprobarlo necesitamos verficar que los resultados se pueden considerar válidos. En la carpeta collab se puede encontrar un Jupyter notebook con una comparación entre nuestro sistema y VADER sentiment

Hemos seleccionado un dataset que contiene las valoraciones de los usuarios y hemos realizado 2 pruebas: Una con nuestro sistema y otra con vader.

Los resultados han sido:

VADER Sentiment

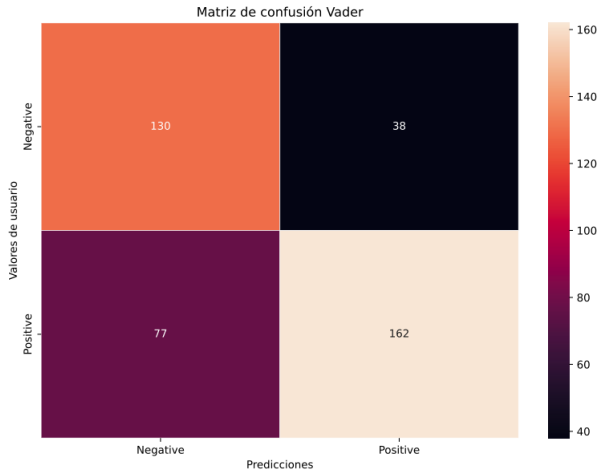

Figura 5.1: Matriz de confusión Vader sentiment

La matriz de confusión representa el número de elementos de cada clase. Como podemos ver hay 38 casos en los que el modelo ha identificado la reseña como positiva cuando en realidad es negativa. Además hay 77 casos en los que ha identificado un sentimiento positivo cuando en realidad es negativo.

```
Resultados con vader:
Precisión:  0.81
Exhaustividad:  0.6778242677824268
```

Figura 5.2: Resultados Vader Sentiment

La precisión representa la proporción de reseñas correctamente identificadas con respecto al total. Dicho de otra forma el porcentaje de reseñas relevantes. Por otra parte, la exhaustividad lo que mide es la proporción de reseñas identificadas con respecto al total de reseñas que se deberían haber clasificado en ese grupo. Como vemos en los resultados, con Vader Sentiment el 81 % de las reseñas identificadas como positivas, eran efectivamente positivas y se ha podido identificar el 61 % del total de las reseñas positivas.

Prompt sentiment

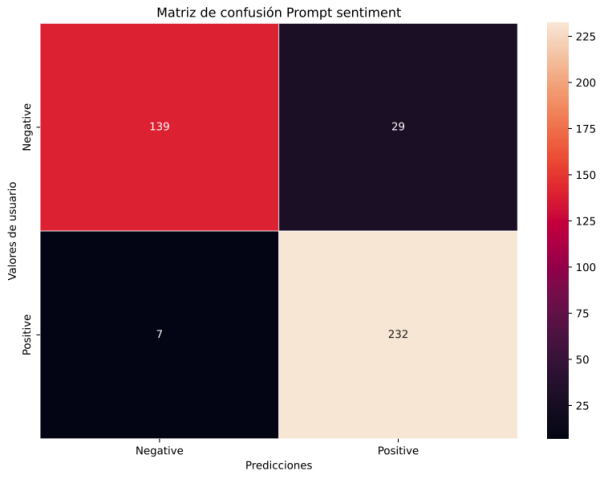

Figura 5.3: Matriz de confusión Prompt Sentiment

Con respecto a nuestro sistema ha identificado 29 reseñas como positivas cuando en realidad son negativas y 7 que identificado como negativas cuando en realidad son positivas.

```
Resultados con Prompt sentiment:
Precisión:  0.8888888888888888
Exhaustividad:  0.9707112970711297
```

Figura 5.4: Resultados Prompt Sentiment

En relación a la precisión el 89 % de las reseñas identificadas como positivas eran, efectivamente positivas. y se ha logrado clasificar correctamente el 97 % de las reseñas positivas.

En vista a estos resultados podemos afirmar que nuestro sistema funciona bastante mejor que Vader Sentiment tanto en precisión como en exhaustividad. Podemos afirmar que nuestro sistema es indestiguible de un humano clasificando reseñas según el sentimiento.

5.7. Uso de herramientas

Durante el transcurso del trabajo he usado ChatGpt tanto para pruebas de análisis de sentimiento como para elaborar el trabajo y su documentación.

ChatGpt [20] resulta una herramienta muy útil tanto para resumir, redactar y realizar borradores. También te puede ayudar a comenzar con el desarrollo de una aplicación o la configuración de una nueva librería en un proyecto existente.

También mencionar que para el desarrollo del trabajo he utilizado Visual Studio Code como editor para todo tipo de ficheros. El conjunto de extensiones disponibles hace que sea muy sencillo programar en diferentes lenguajes eficientemente. Desde el mismo editor se pueden compilar y visualizar el resultado. En concreto lo he usado para Python, Javascript, Vue.js, LaTeX e incluso Jupyter Notebooks.

6. Trabajos relacionados

6.1. Lingmotif [17]

Lingmotif es una aplicación multi-plataforma de sobremesa que analiza textos desde la perspectiva del Análisis de Sentimiento.
Básicamente, es capaz de determinar la orientación semántica (si es positivo o negativo y en qué grado) de un texto o conjunto de textos, mediante **la detección de expresiones lingüísticas** que indican una determinada polaridad.

A diferencia de la mayoría del software existente, Lingmotif no es un sólo un clasificador, ya que no se limita a clasificar un texto como positivo o negativo, sino que además ofrece una serie de datos cuantitativos, una visualización del "perfil de sentimiento" del texto o textos (incluyendo series temporales), y un detallado análisis cualitativo del texto en sí, en el que se muestran los segmentos textuales identificados.
Estas funcionalidades lo convierten en un herramienta única, y sus aplicaciones van más allá de las que normalmente ofrece este tipo de software. Lingmotif ofrece los resultados de sus análisis en archivos con formato HTML, con la versatilidad y fácil manejo que esto supone.

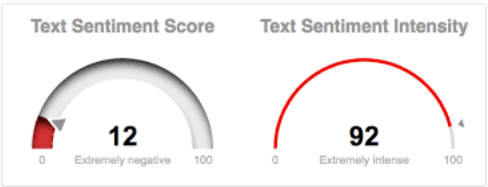

Figura 6.1: Lingmotif

6.2. Lexalytics [13]

Lexalytics es una empresa que se especializa en tecnologías de análisis de texto y procesamiento del lenguaje natural (PLN).
Ofrece soluciones para extraer información de datos de texto no estructurados, como contenido de redes sociales, comentarios de clientes, reseñas y más. Los productos de Lexalytics están diseñados para ayudar a las empresas a analizar y comprender el sentimiento, los temas y las tendencias dentro de grandes volúmenes de información textual. Uno de sus productos destacados es la Plataforma de Inteligencia Lexalytics, que incluye varias herramientas y funciones para el análisis de texto. Tienen un API (llamado Semantria)

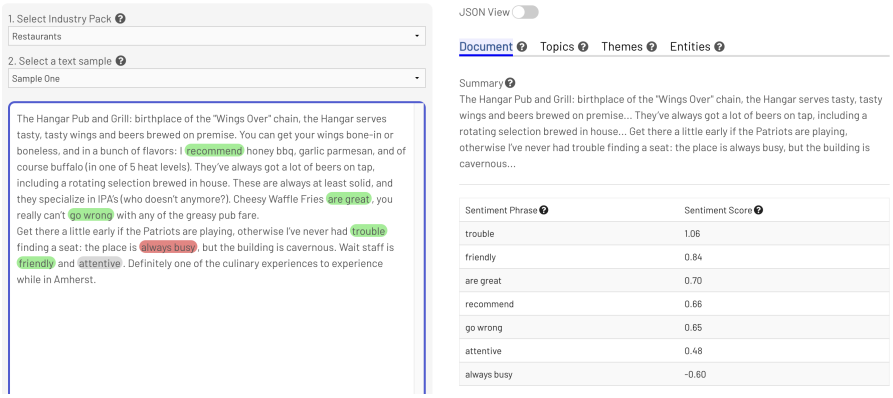

Figura 6.2: Lexalytics análisis de texto

que se usa para análisis de texto y sentimiento. Admite varios idiomas, lo que las hace adecuadas para empresas con presencia global.
Estas herramientas y servicios pueden ser valiosos para empresas que buscan dar sentido a la gran cantidad de texto no estructurado que generan o encuentran en el curso de sus operaciones.
Se pueden aplicar en áreas como la gestión de la experiencia del cliente, el monitoreo de redes sociales, la investigación de mercado y otros campos donde comprender la información textual es crucial.

6.3. IBM Watson Studio [10]

IBM Watson Studio es una plataforma de inteligencia artificial (IA) desarrollada por IBM que proporciona servicios y herramientas para el procesamiento del lenguaje natural y otros. Lleva el nombre del fundador de IBM (Thomas J. Watson) También utiliza técnicas de aprendizaje automático para extraer patrones a partir de grandes conjuntos de datos y realizar análisis predictivos.
Tiene capacidades para analizar y comprender imágenes y videos, lo que puede ser útil en aplicaciones como el reconocimiento facial y la interpretación de contenido visual.
Se ha utilizado en aplicaciones como juegos de preguntas y respuestas, demostrando su capacidad para competir y ganar contra humanos en contextos complejos. Puede automatizar tareas y procesos empresariales, mejorando la eficiencia operativa. Analiza datos en tiempo real para proporcionar información instantánea y tomar decisiones basadas en datos. IBM Watson se ha aplicado en diversos campos obtenido muy buenos resultados. Permite crear un cuenta gratuita (limitada) para poder probarla.

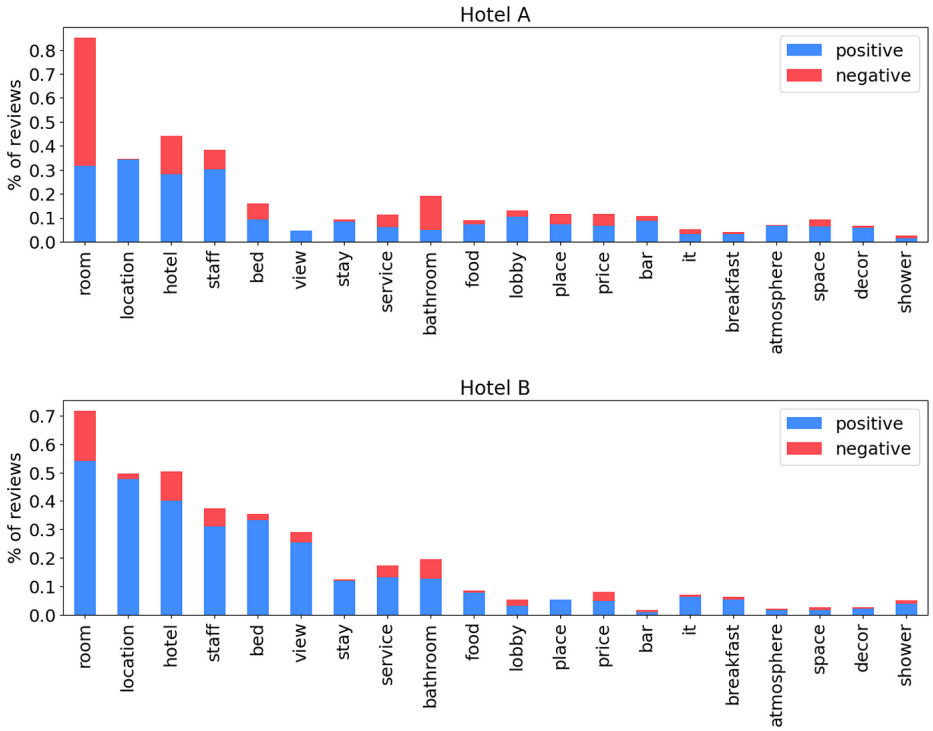

Figura 6.3: IBM Watson Studio

6.4. Sentinel [5]

Sentinel 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.

6.5. Free Sentiment Analysis [22]

Es una herramienta gratuita que permite realizar un analisis de sentimiento de cualquier texto escrito en Inglés. La herramienta califica el sentimiento con un número entre -100 y 100 dependiendo del sentimiento detectado en el texto. Sólo hay que pegar el texto en el cuadro de texto y pulsar el botón “Analyze text”

Para su funcionamiento utiliza algoritmos de linguística computacional y minería de texto. El modelo se ha entrenado usando el “American National Corpus” con lo que sólo funciona con textos en inglés americano y escritos después de 1990. No se conoce su funcionamiento interno pero por la descripción parece que ha usado un corpus anotado y usa las coincidencias de palabras para dar una nota al texto dependiendo de las palabras encontradas.

7. Conclusiones y Líneas de trabajo futuras

7.1. Conclusiones

El uso de modelos grandes de lenguaje está cada vez más extendido en muchos sectores tecnológicos.
En estos momentos hay una explosión de proyectos y usos que no eramos capaces de imaginarnos hace un años.
En este trabajo se han combinado estas técnicas con otras más tradicionales para conseguir una aplicación útil con muchas oportunidades de mejora y ampliación.
Lo que he intentado es establecer una plataforma que sea flexible para poderla ampliar y escalar tanto como se necesite.
En el camino he aprendido desde el uso de LLMs, python, web services, seguridad (mediante jwts), despliegue continuo con docker, etc. El resultado creo que es una aplicación sencilla pero con mucho potencial y suficientemente flexible como para servir de base de otros desarrollos más complejos.

7.2. Líneas de trabajo futuras

Nuevos modelos

Actualmente la aplicación soporta el uso de OpenAI y Hugging face debido a las limitaciones de ambas, sería adecuado ampliar las opciones. Una de las posible opciones sería instalar un modelo en local pero se descartó debido al pobre rendimiento y la excesiva cantidad de recursos necesarios. La opción más viable sería incluir un gestor de colas de mensajes (tipo RabbitMQ) para procesar las distintas tareas. De esta forma se podría distribuir el trabajo en diferentes ordenadores con mejores prestaciones (incluyendo tarjetas gráficas).

Nuevas fuentes de datos

La aplicación en el momento de presentación al tribunal sólo permite procesar reseñas desde un fichero (ya sea Csv o Json). Sería interesante agregar nuevas fuentes de datos como servicios web o similares.

Más opciones de análisis

Sería muy interesante darle a usuario la posibilidad de modificar el prompt para permitir extraer más información útil de las reseñas.

GridFS

Ahora mismo al importar un fichero de reseñas éste se almacena en directorio temporal en el contenedor docker.
Podría ser interesante utilizar una solución más robusta como puede ser GridFS. Aunque una de las ventajas de usarlo en un directorio temporal es que estos ficheros desaparecen al reiniciar.

Bibliografía

[1] Meta AI. Llama: open and efficient foundation language models. https://research.facebook.com/publications/llama-open-and-efficient-foundation-language-models/.

[2] Kristian Bannister. Entendiendo el análisis de sentimiento. https://www.brandwatch.com/es/2015/02/analisis-de-sentimiento/, 2015.

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

[4] Anna Bedrava. Guía del análisis de opiniones. https://awario.com/es/blog/sentiment-analysis/, 2022.

[5] Zoe Calvo. Sentinel: trabajo fin de grado. https://github.com/ZoeCalvo/Sentinel, 2020.

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

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

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

[9] Grigori Gerganov. Llama.cpp. https://github.com/ggerganov/llama.cpp.

[10] IBM. Ibm watson studio [computer software]. https://www.ibm.com/products/watson-studio, 2024.

[11] Julian McAuley Jianmo Ni, Jiacheng Li. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. https://cseweb.ucsd.edu/ jmcauley/pdfs/emnlp19a.pdf.

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

[13] Lexalytics. Lexalytics[computer software]. https://www.lexalytics.com/, 2024.

[14] Mark Lutz. *Programming python*. O'Reilly Media, Inc., 2001.

[15] 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.

[16] Antonio Moreno. ¿qué es el procesamiento del lenguaje natural? https://www.iic.uam.es/inteligencia/que-es-procesamiento-del-lenguaje-natural/, 2018.

[17] A. Moreno-Ortiz. Lingmotif 1.0 [computer software]. http://tecnolengua.uma.es/lingmotif, 2016.

[18] NetApp. ¿qué es devops? https://www.netapp.com/es/devops-solutions/what-is-devops/, 2023.

[19] nlptown. Modelo especializado en analizar sentimientos en comentarios de usuario. https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment, 2023.

[20] OpenAI. Chat gpt. https://chat.openai.com/, 2023.

[21] He Saif, Fernandez and Alani. Evaluation datasets for twitter sentiment analysis. https://ceur-ws.org/Vol-1096/paper1.pdf, 2013.

[22] Daniel Soper. Free sentiment analizer. https://www.danielsoper.com/sentimentanalysis/default.aspx, 2020.

[23] Marta Solano Tatché. Prompt engineering: todo lo que debes saber. https://www.eaeprogramas.es/blog/negocio/tecnologia/prompt-engineering-todo-lo-que-debes-saber, 2023.

[24] 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. https://arxiv.org/abs/2302.11382v1, 2023.