Prototipo 4 - Análisis de sentimientos de imágenes

Andrés Felipe Patarroyo Muñoz - 506221032
Santiago Jair Torres Rivera - 506221074

***Abstract*—En este documento se presenta un proyecto que busca analizar y clasificar emociones en textos, organizando comentarios en categorías como Muy Negativo, Negativo, Neutro, Positivo y Muy Positivo. Implementa una versión funcional que clasifica comentarios en tres categorías principales: positivos, neutrales y negativos. El proyecto se desarrolló en dos partes, se utilizó Python y FastAPI para el backend, y HTML, CSS, JavaScript, JQuery y Ajax para el frontend. Los beneficios incluyen la capacidad de filtrar y seleccionar comentarios según su tono emocional, lo que es útil para mejorar la atención al cliente, analizar opiniones en redes sociales o monitorear la retroalimentación en productos y servicios. Proporcionando una plataforma accesible y eficiente para el análisis de emociones, este proyecto puede ayudar a las organizaciones a tomar decisiones informadas basadas en el sentimiento de los usuarios.**

Keywords: Análisis de emociones, clasificación de comentarios, categorización emocional, FastAPI, Python, HTML, CSS, JavaScript, JQuery, Ajax, NLTK, Redes sociales, Opiniones de usuarios, tokenización.

I. Introducción

Hoy en día la tecnología sigue ganado importancia y cada vez es más usada en el mundo laboral, por tanto, la comprensión de las emociones detrás de los mensajes de texto se ha vuelto esencial en numerosos campos. Este proyecto se centra en desarrollar una solución para analizar emociones en textos digitales, utilizando técnicas avanzadas de procesamiento de lenguaje natural (PLN).

Utilizando FastAPI de Python como base, creamos una API flexible y eficiente que permite realizar análisis de sentimientos, recuperar comentarios aleatorios y filtrar comentarios por tipo de emoción. Una función destacada es ”emotionsText-detect”, que utiliza un modelo pre-entrenado basado en BERT para analizar el sentimiento de cada oración por separado, ofreciendo una comprensión detallada del tono emocional de un texto.

En el frontend, se diseñó una interfaz intuitiva que permite a los usuarios interactuar fácilmente con la API. Desde un formulario para ingresar texto hasta opciones para buscar comentarios aleatorios y filtrarlos por emoción, el frontend garantiza una experiencia de usuario fluida.

A lo largo del documento, se explorará cómo esta solución puede aplicarse en diferentes contextos, como la monitorización de redes sociales, la atención al cliente y el análisis de opiniones en línea, destacando su valor en la comprensión y análisis de emociones expresadas en textos digitales.

II. Implementación

Entorno virtual y librerias

Desde la fase inicial del proyecto, se estableció un proceso para configurar un entorno virtual y gestionar las bibliotecas esenciales y necesarias para el funcionamiento del proyecto. Se creó un entorno virtual de Python, un espacio aislado dedicado exclusivamente para el proyecto en cuestión. Esto se logró mediante el uso de la herramienta virtualenv y siguiendo una serie de comandos estándar en la terminal. Una vez el entorno virtual activo, se procedió a la instalación de las bibliotecas requeridas, tal como se especifica en el archivo de requisitos ”requirements.txt”. Este proceso, se realizó en la terminal con el comando ”pip install -r requirements.txt”, y se garantizó la instalación de las bibliotecas principales, incluyendo fastapi, nltk, tensorflow, torch, transformers, tf-keras y uvicorn. Finalmente, una vez que todas las bibliotecas necesarias fueron instaladas con éxito, se procedió al lanzamiento del proyecto utilizando el servidor uvicorn. Esta acción permitió la puesta en marcha de la aplicación, que estuvo disponible en la dirección URL: http://127.0.0.1:3000, teniendo en cuenta el puerto seleccionado durante la configuración.

Backend

Se utilizó FastAPI como framework de Python para crear APIs de manera rápida y sencilla. En él, se montan rutas estáticas para archivos como CSS o JavaScript, se definen plantillas para renderizar páginas HTML y se añade un middleware para gestionar el acceso de origen cruzado (CORS). Además, se definen distintas rutas (endpoints) para manejar diferentes solicitudes HTTP. Por ejemplo, una ruta para mostrar la página principal (”/”), otra para analizar un texto enviado mediante una solicitud POST (”/analizarTexto”), otra para obtener con una petición GET un comentario aleatorio (”/comentarioAleatorio”) y finalmente, otra para filtrar los comentarios con una petición GET según su tipo de emociones y que devuelve los resultados (”/comentarios”). La función ”emotionsText-detect” analiza texto haciendo uso de otra función cuyo código define una función para analizar el sentimiento de un texto utilizando un clasificador de sentimientos pre-entrenado basado en BERT. La función recibe un texto como entrada y utiliza la biblioteca Hugging Face Transformers para dividir el texto en oraciones y luego analizar el sentimiento de cada oración individualmente. Luego, calcula un promedio de las puntuaciones de sentimiento de todas las oraciones para obtener un puntaje general de sentimiento para el texto. Finalmente, devuelve el puntaje general y la etiqueta de sentimiento correspondiente. Este enfoque permite una comprensión más detallada del sentimiento en textos largos, ya que considera el sentimiento de cada oración por separado. La integración de un modelo pre-entrenado que realiza el procesamiento de lenguaje natural (NLP) para detectar emociones.

Frontend

Para el desarrollo del frontend del proyecto, se desarrolló una página que presenta un formulario con un área de texto para ingresar el texto que se va a analizar, un botón para buscar un comentario aleatorio y un botón para enviar el texto ingresado para su análisis en el backend, además, se incluyeron tres botones adicionales para filtrar y mostrar comentarios positivos, neutrales o negativos provenientes del consumo de un archivo JSON. Finalmente, al momento de filtrar y mostrar comentarios, se agregan en forma de tarjeta y al final todos los comentarios se encuentra un botón para eliminar los comentarios filtrados, y como dato adicional, se muestra la cantidad de comentarios que se encontraron según el tipo de comentario solicitado.

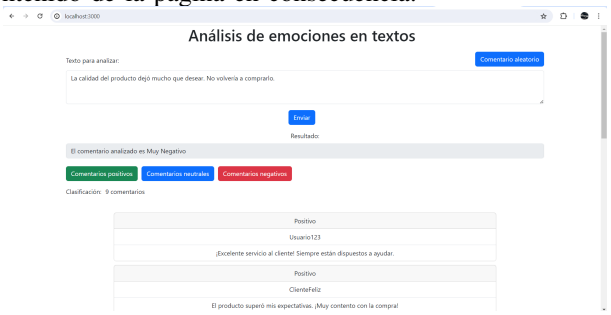
El código JavaScript, JQuery y Ajax maneja las interacciones del usuario con la página. Se utilizan eventos de clic para enviar solicitudes POST y GET a la API del servidor local para analizar el texto ingresado, buscar un comentario aleatorio y obtener comentarios filtrados según su tipo de emoción. Los resultados de estas solicitudes se muestran dinámicamente en la página web utilizando la manipulación del DOM.

III. Resultados

El desarrollo del proyecto se centró en la implementación de una solución robusta y eficiente para analizar emociones en textos, abordando tanto el frontend como el backend de la aplicación. En el backend, se optó por utilizar FastAPI, un framework de Python que permite crear APIs de manera rápida y sencilla. Con FastAPI, se definieron rutas para manejar distintas solicitudes HTTP, desde mostrar la página principal hasta analizar texto, obtener comentarios aleatorios y filtrar comentarios según su tipo de emoción.

Uno de los aspectos destacados del backend fue la función ”emotionsText-detect”, la cual utiliza un clasificador de sentimientos pre-entrenado basado en BERT para analizar el texto ingresado. Este enfoque permite una comprensión más detallada del sentimiento en textos largos, al considerar el sentimiento de cada oración por separado. Además, la integración de un modelo pre-entrenado agrega precisión al proceso de análisis de sentimientos.

En cuanto al frontend, se diseñó una página web intuitiva que permite al usuario ingresar texto para su análisis, buscar comentarios aleatorios y filtrar comentarios según su tipo de emoción. La implementación de JavaScript, JQuery y Ajax permitió gestionar de forma dinámica las interacciones del usuario, enviando solicitudes al backend y actualizando el contenido de la página en consecuencia.



IV. Conclusión

Este proyecto demuestra la eficacia de tecnologías modernas como FastAPI y modelos de procesamiento de lenguaje natural (PLN) para el análisis y clasificación de emociones en textos. La implementación robusta tanto en el backend como en el frontend permite una experiencia de usuario fluida y una clasificación precisa de los comentarios emocionales. La función ”emotionsText-detect”, basada en un modelo BERT pre-entrenado, ha sido clave para ofrecer un análisis detallado del sentimiento de cada oración, mejorando la precisión general. La integración de JavaScript, JQuery y Ajax en el frontend facilita la interacción dinámica con la API, permitiendo a los usuarios ingresar texto, buscar comentarios aleatorios y filtrar comentarios por tipo de emoción. La capacidad de filtrar y seleccionar comentarios según su tono emocional es una herramienta valiosa para diversas aplicaciones, como la mejora de la atención al cliente y el análisis de opiniones en redes sociales. Este proyecto proporciona una plataforma accesible y eficiente para el análisis de emociones, ayudando a las organizaciones a tomar decisiones informadas basadas en el sentimiento de los usuarios.