Prototipo: Creación de un Servicio de Análisis de Sentimientos para Textos Largos

Natalia Chitiva Huertas - 506221070
Juan Jose Caldera Tamayo - 506212033
Andres Felipe Lopez Anaya - 506221079

Fecha: 30-05-2024

Fundación Universitaria Konrad Lorenz

1. Resumen

Este proyecto es un servicio de evaluación de sentimientos para textos largos. Este permite el analisis en tiempo real de comentarios. Se utilizo FastAPI en el backend y una interfaz gráfica desarrollada con HTML, CSS y JavaScript, los usuarios pueden ingresar textos largos para ser analizados por un modelo de procesamiento de lenguaje natural basado en transformers de Hugging Face. Para usar el servicio Para usar el servicio, simplemente se ingresa el texto en la interfaz y se obtiene la evaluación de sentimiento, proporcionando una herramienta efectiva para gestionar y moderar comentarios.

2. Introducción

En este proyecto, hemos desarrollado un servicio de evaluación de sentimientos que permite analizar en tiempo real textos extensos y comentarios de redes sociales mediante un modelo impulsado por inteligencia artificial. La interfaz gráfica, esta diseñada para ser atractiva, permite a los usuarios ingresar textos para su análisis de manera sencilla y rápida.

3. Resultados

En el paso 1 se define el área principal del chatbox incluyendo un contenedor de los mensajes y un formulario para la entrada de texto.

```
<div class="chat-container">
    <div class="chat-box">
        <div class="chat-messages" id="chat-
    messages">
            <!-- Aqu  se mostrar n los mensajes
    -->
        </div>
        <div class="input-container">
            <form id="message-form">
            <input type="text" id="user-input"
    name="input_text" placeholder="Escribe tu
    mensaje...">
            <button type="submit">Enviar</button>
        </form>
        </div>
    </div>
</div>
```

Code 1. *Paso 1.*

En el paso 2 se envía una solicitud POST al servidor con el texto ingresado por el usuario y procesa la respuesta del servidor y llama a la función sendMessage con la clasificación final del sentimiento.

```
fetch('http://127.0.0.1:8000/analizarSentimiento',
     {
    method: 'POST',
    headers: {
        'Content-Type': 'application/x-www-form-
    urlencoded'
    },
    body: 'input_text=' + encodeURIComponent(
    userInput)
})
.then(response => response.json())
.then(data => {
    sendMessage(data.final_star_rating);
})
.catch(error => console.error('Error:', error));
```

Code 2. *Paso 2.*

En este paso 3 se comprueba si el campo de texto no está vacío, se drea y añade un nuevo mensaje del "doctorçon el resultado del análisis incluyendo la hora

```
function sendMessage(finalData) {
    var userInput = document.getElementById("user-
    input").value;

    if (userInput.trim() === "") {
        alert("Por favor, ingresa un mensaje antes
     de enviar.");
        return;
    }

    var chatMessages = document.getElementById('
    chat-messages');

    var userMessage = document.createElement('div'
    );
    userMessage.className = 'message user-message'
    ;

    var userAvatar = document.createElement('img')
    ;
    userAvatar.className = 'avatar';
    userAvatar.src = './logo.png';
```

Code 3. *Paso 3.*

En este paso 4 se importan las librerias necesarias, se inicializa la aplicación FastAPI y configura CORS para permitir solicitudes desde cualquier origen.

```
from fastapi import FastAPI, Form
from transformers import pipeline
import nltk
from nltk.tokenize import sent_tokenize
from fastapi.middleware.cors import CORSMiddleware

```

```
nltk.download('punkt')

app = FastAPI()

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["POST"],
    allow_headers=["*"],
    expose_headers=["Content-Disposition"]
)
```

Code 4. *Paso 4.*

En este paso 5 se divide el texto en fragmentos de tamaño adecuado para el model y se realiza la tokenizacion.

```
def fragmentar_texto(texto, max_tokens=512):
    oraciones = sent_tokenize(texto)
    fragmentos = []
    fragmento_actual = []

    for oracion in oraciones:
        fragmento_actual.append(oracion)
        if len(clasificador.tokenizer.encode(" ".join(fragmento_actual))) >= max_tokens:
            fragmentos.append(" ".join(fragmento_actual[:-1]))
            fragmento_actual = [oracion]

    if fragmento_actual:
        fragmentos.append(" ".join(fragmento_actual))

    return fragmentos
```

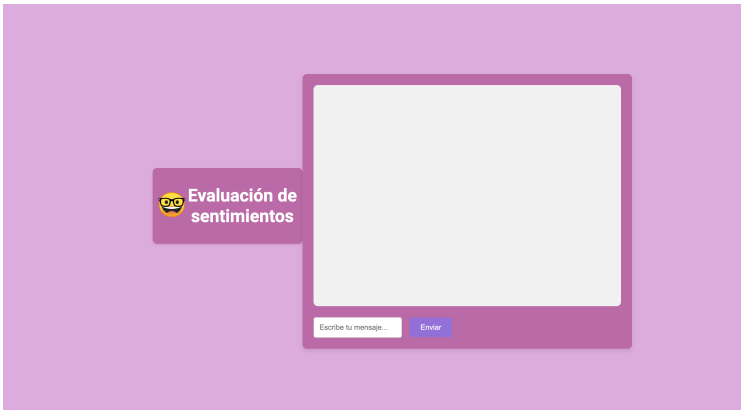
Code 5. *Paso 5.*

En este paso 6 se utiliza la función fragmentar.texto para dividir el texto en fragmentos manejables, calcula la puntuación promedio y determina la clasificación final del sentimiento y devuelve la clasificación final como respuesta JSON al usuario.

```
@app.post("/analizarSentimiento")
def analyze_sentiment(input_text: str = Form(...)):
    fragmentos = fragmentar_texto(input_text)

    puntos = {
        '1 star': -1,
        '2 stars': -0.5,
        '3 stars': 0,
        '4 stars': 0.5,
        '5 stars': 1
    }

    puntos_acumulados = 0

    for fragmento in fragmentos:
        resultado = clasificador(fragmento
```

Code 6. *Paso 6.*

4. Imagenes del resultado



5. Conclusiones

En conclusión, este proyecto nos ha permitido el desarrollo de una herramienta de análisis de sentimientos de textos largos, utilizando tecnologías avanzadas como FastAPI para el backend, NLTK para la tokenización de textos, y un modelo de análisis de sentimientos preentrenado de Hugging Face. La implementación de un frontend interactivo con HTML, CSS y JavaScript ha permitido a los usuarios enviar textos y recibir evaluaciones de sentimientos.