Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2024

ENSAYO DE PROYECTO 3

202302220 – Enner Esaí Mendizabal Castro

Resumen

El ensayo presenta un proyecto para la empresa Tecnologías Chapinas, S.A. desarrolla un analizador de contenido de redes sociales para evaluar el sentimiento de los usuarios sobre empresas y sus servicios. Este análisis se basa en un análisis léxico que utiliza un diccionario de palabras positivas y negativas, enviado junto a un archivo de entrada con mensajes. El proyecto incluye la implementación de un backend en Python utilizando Flask y un frontend desarrollado con Django. El backend se encarga de procesar el archivo XML, separando los elementos relevantes de los mensajes y almacenando esta información en estructuras de datos organizadas. El frontend, construido con Django y Bootstrap, se encarga de presentar los datos de manera intuitiva y responsiva. La comunicación entre el frontend y el backend se realiza a través de peticiones HTTP, donde se intercambian archivos JSON que contienen la información necesaria para la visualización y análisis.

Palabras clave

HTTP, Django, Frontend, Backend, análisis léxico

Abstract

The essay presents a project for the company Tecnologías Chapinas, S.A. that develops a social media content analyzer to evaluate user sentiment about companies and their services. This analysis is based on a lexical analysis that uses a dictionary of positive and negative words, sent along with an input file with messages. The project includes the implementation of a backend in Python using Flask and a frontend developed with Django. The backend is responsible for processing the XML file, separating the relevant elements of the messages and storing this information in organized data structures. The frontend, built with Django and Bootstrap, is responsible for presenting the data in an intuitive and responsive way. Communication between the frontend and the backend is done through HTTP requests, where JSON files containing the necessary information for visualization and analysis are exchanged.

Keywords

HTTP, Django, Frontend, Backend, lexical analysis

Introducción

La empresa Tecnologías Chapinas, S.A. pretende desarrollar un analizador de contenido de redes sociales, el cual permita conocer el sentimiento de los usuarios respecto a cada una de las empresas y a sus servicios. Para esto se realizará un análisis léxico que funcionará a través de un diccionario que se enviará junto a un archivo de entrada que contendrá los mensajes.

En es este proyecto se realizará el software que será capaz de ejecutar analizar el contenido, determinar si el mensaje contiene un sentimiento positivo, negativo o neutro sobre algún servicio de alguna empresa y mostrará toda la información mediante un frontend creado a partir de Django, el cual se conectará al backend, que realizará todas las operaciones, con Flask. Este frontend fungirá como un simulador de la aplicación principal que contendrá las funcionalidades necesarias para testear el backend mediante gráficas y más opciones.

Desarrollo del tema

El Proyecto 3 de Introducción a la Programación y Computación 2 trató sobre la creación de un programa web capaz de analizar los mensajes, permitiendo saber la opinión de los usuarios sobre distintas empresas y sus distintos servicios.

Backend

El backend es la parte del programa que ejecutará todas las acciones de este y simplemente enviará la información requerida para que se muestre en el frontend.

Todo el programa fue realizado con Python, pero, adicionalmente, se usó el Framework de Flask para la conexión con el frontend.

Flask

Flask es un micro Framework que permite un desarrollo de aplicaciones bajo el patrón MCV, patrón que divide las aplicaciones en tres componentes principales: Modelo, Vista y Controlador. (Muñoz, 2017)

La palabra *micro* para referirse a Flask no significa que sea pequeño y únicamente permita la creación de proyecto pequeños, sino a que cuando se instala simplemente no incluye las herramientas necesarias y esenciales para crear una página web funcional, pero si se necesitan más funcionalidades, se pueden instalar extensiones.

Si bien, una de las principales ventajas de Flask es el trabajo fácil con vistas, también permite un manejo simple de rutas para utilizarlo en el backend.

REGEX

En este proyecto era necesaria la implementación de un analizador léxico para cada una de las palabras que se busca encontrar para poder definir los sentimientos de las personas hacia las empresas y sus servicios, pero también es podía usar una expresión regular.

Las expresiones regular o *REXEX* son, de cierta forma, un autómata finito que es capaz de aceptar el lenguaje especificado por la expresión. Estas son cadenas de caracteres basadas en reglas sintácticas que permiten describir secuencias de caracteres (Equipo editorial de IONOS, 2019).

El funcionamiento de la expresión regulares se basa simplemente en la coincidencia exacta, es decir, si se tiene el patrón *abc*, entonces se buscará la cadena que contenga con exactitud esos caracteres en exactamente ese mismo orden (Equipo editorial de IONOS, 2019)

Para poder definir esto se empleó una expresión regular la cual sería capaz de leer todo el mensaje y separar el lugar, la fecha, la hora, el usuario, la red social, la empresa y el texto de este mensaje. Una vez con todo separado, se crearon ciclos que analizarían individualmente los textos de los mensajes con cada una de las palabras positivas y negativas; esto para contarlas y saber cuántas positivas y negativas había. Por último, se trabajó de la misma manera para encontrar las empresas de las que se habla en cada mensaje y sus servicios.

Almacenamiento de la información de entrada

La principal función del backend en este proyecto es la lectura del archivo de entrada con extensión .XML.

Para el proceso de lectura del archivo de entrada, se comenzó creando un total de 5 listas globales, las cuales permitirían que se pudiera almacenar en estas todas las empresas dentro del diccionario, todas las palabras del diccionario que son positivas, todas las palabras del diccionario que son negativas, todos los mensajes con cada uno de sus elementos ya analizados, separados y organizados correctamente y un última con todas las fechas con la lista de todos los mensajes que hay en esa fecha.

Se comienza creando una copia del archivo de entrada dentro de la carpeta *data* para poder volverlo a leer de ser necesario, posteriormente se procede a la guardar la información del archivo.

Para guardar la información se crearon varios objetos de POO, se fue recorriendo todo el archivo .XML y conforme avanzaba se iba almacenando dentro de objetos, que en muchos casos contenían otros objetos, se fueron colocando cada uno de los elementos conforme se fueron analizando y se terminan colocando dentro de cada una de sus listas respectivas.

Una lista especial es la de fechas, dado a que analiza las otras listas ya creadas para crear esa nueva lista que ordena todo en las fechas que se tienen entre todos los mensajes analizados.

Creación del archivo de salida

Para la creación del archivo de salida se recurrió al uso de la librería ElementTree, la cual se ha estado utilizando en todos los proyectos, al igual que para la lectura del archivo de entrada.

Se comenzó con la creación del archivo de salida con el establecimiento de la ruta en la que se guardaría la salida y un ciclo que leería todos los mensajes que hay por fecha para poder colocar su total, sus positivos y sus negativos. Posteriormente, debido a que era necesario mostrar el análisis por cada empresa, anidó otro ciclo a partir de la lista de empresas y se contrarían todos los mensajes que mencionen a cada empresa, pero también se necesitaba que se mostrara por servicio, por lo que, como última anidación, se ingreso un ciclo dentro del ciclo de empresas que sería a partir de la lista de servicios que contendría la empresa, de esta forma se contrarían todos los mensajes que contengan tal servicio dentro de su texto. Finalmente, toda esta información, se fue colocando conforme se fue creando en la estructura del árbol y al finalizar, se indentó y se escribió en la posición indicada.

Mediante HTTP se envió la salida la frontend para que lo mostrara.

Frontend

Para la creación del frontend, se utilizó Django, este es otro Framework que se puede utilizar con Python que, a diferencia de Flask, es más completo y tiene una mayor cantidad de posibilidades, especialmente pensado para proyecto de mayor magnitud.

Un proyecto de este tipo se divide en distintas partes, en las cuales están las rutas, las plantillas y las vistas.

Templates

Para la creación de las plantillas se usó Bootstrap, el cual es un Framework de desarrollo web gratuito que permite que se puedan crear sitios web más fácilmente mediante *pre-sets* que este mismo proporciona, permitiendo que se pueda generar un sitio web más responsivo e intuitivo sin tanto esfuerzo.

Vistas y Rutas

En la sección de rutas con Django, se colocó cada una de las rutas que se usarían para mostrar y recibir información con el frontend, estas *rutas* van siempre conectas a las vistas, dado a que cada ruta está conectada directamente con una función que permite la visualización del contenido y la ejecución de los procesos que se desean realizar.

Conexión entre Templates y Vistas

Dicho burdamente, las vistas son las encargadas de conectar todo en el frontend con Django, esto debido aquí es donde se colocan todas las funciones.

Al igual que Flask, Django trabaja con Jinja2 para poder comunicarse con las Templates escritas en HTML. Para la comunicación se renderiza la plantilla HTML en la ruta indicada y se le envía un diccionario, el cual se puede leer posteriormente con Jinja2 dentro del archivo HTML con su respectiva sintaxis.

Conexión entre el Frontend y el Backend

Probablemente la parte más difícil del proyecto es la conexión entre el backend y el frontend, esto debido a que, de cierta forma, se simulan dos servidores que se deben ir comunicando entre sí para conseguir que se puedan ejecutar las operaciones de la aplicación web.

Para la conexión se utilizaron peticiones http. Específicamente en este proyecto, se enviaron archivos de tipo JSON, que contendrían su nombre y el archivo que se enviaría.

Conclusiones

La implementación de un analizador léxico basado en expresiones regulares es una herramienta poderosa para extraer y clasificar sentimientos en mensajes de redes sociales de manera sencilla.

La comunicación entre el frontend y el backend representa un desafío significativo en el desarrollo de aplicaciones web si no se tiene la experiencia necesaria. El uso de peticiones HTTP y el formato JSON para el intercambio de datos es crucial para facilitar la trasmisión de información.

El uso de Bootstrap para el diseño del frontend permite crear una interfaz de usuario atractiva y funcional rápidamente.

La combinación de Flask y Django para el desarrollo de aplicaciones web es muy buena. Se aprovechan las fortalezas de cada Framework, pues Flask permite una gestión eficiente y simple del backend y Django proporciona un frontend robusto y escalable.

Referencias bibliográficas

Equipo editorial de IONOS. (30 de diciembre de 2019). *Regex o expresiones regulares: la manera más sencilla de describir secuencias de caracteres*. Obtenido de IONOS: https://www.ionos.mx/digitalguide/paginas-web/creacion-de-paginas-web/regex/

Muñoz, J. D. (17 de noviembre de 2017). *Qué es Flask*. Obtenido de OpenWebinars: https://openwebinars.net/blog/que-es-flask/

Apéndice

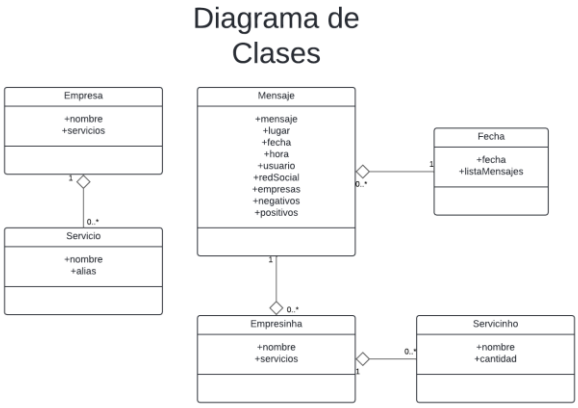

Figura 1. Diagrama de clases del proyecto
Fuente: Elaboración propia

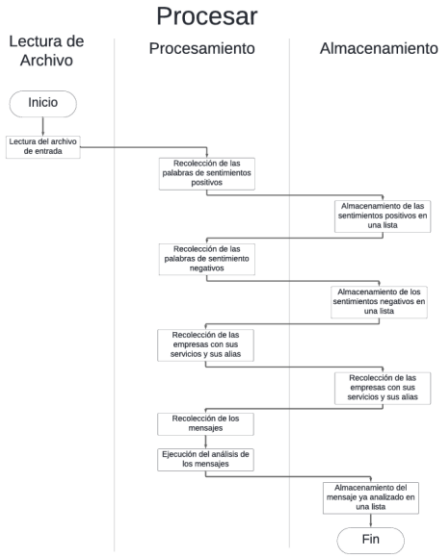

Figura 2. Diagrama de actividades de la función dedicada al almacenamiento y ordenamiento del archivo de entrada
Fuente: Elaboración propia

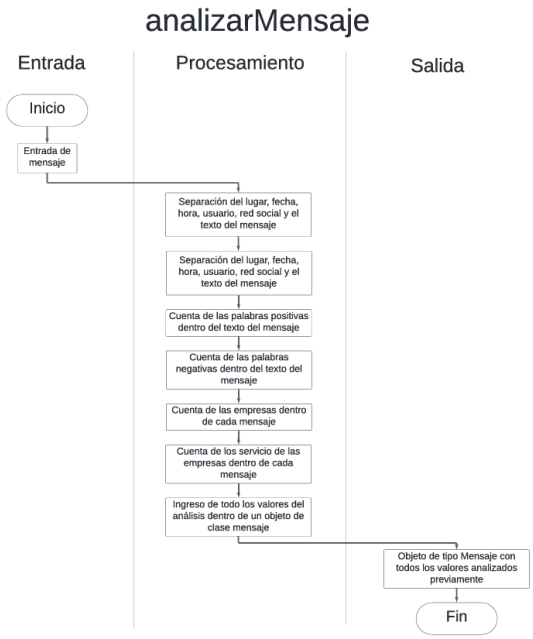

Figura 3. Diagrama de actividades de la función dedicada a analizar el mensaje individual e identificar las palabras positivas y negativas y las empresas y sus servicios.
Fuente: Elaboración propia

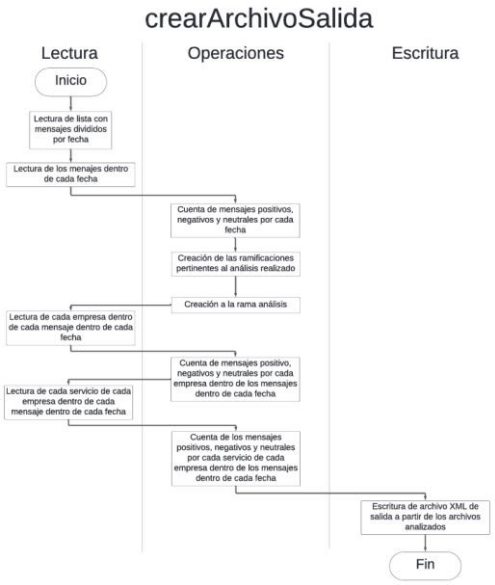

Figura 4. Diagrama de actividades de la función dedicada a crear el archivo de salida con extensión .XML dentro de la base de datos
Fuente: Elaboración propia

Figura 5. Diagrama de actividades de la función dedicada a clonar el archivo de entrada con extensión .XML dentro de la base de datos
Fuente: Elaboración propia