Proyecto 3

202111219 – Ricardo Andreé Paniagua Rodenas

Resumen

El Proyecto 3 de Tecnologías Chapinas, S.A. tiene como objetivo desarrollar una herramienta que analice el contenido de redes sociales para determinar el sentimiento de los usuarios hacia empresas y servicios. Para lograrlo, se implementará una API utilizando el lenguaje Python y el framework Flask, que permitirá procesar mensajes estructurados en formato XML. La herramienta clasificará los mensajes como positivos, negativos o neutros, basándose en un diccionario de palabras que definen el sentimiento. Además, se emplearán expresiones regulares para extraer información clave, como nombres de usuarios, lugares y redes sociales. La respuesta se generará en formato XML, presentando estadísticas sobre la cantidad total de mensajes y su clasificación. El frontend se desarrollará con Django, permitiendo cargar archivos, realizar consultas y generar reportes. Este enfoque integral no solo facilitará el análisis de datos, sino que también proporcionará a la empresa una visión clara de la percepción pública, lo que permitirá tomar decisiones informadas y mejorar sus servicios.

Palabras clave

API (Interfaz de Programación de Aplicaciones): Conjunto de reglas y protocolos que permiten la comunicación entre diferentes sistemas de software, facilitando la integración y el intercambio de datos.

Sentimiento: Análisis que determina si un mensaje es positivo, negativo o neutro, basado en el contenido textual, utilizando un diccionario de palabras clave que representan cada categoría.

XML (Lenguaje de Marcado Extensible): Formato de archivo que permite la estructuración y el almacenamiento de datos en texto legible, utilizado en este proyecto para intercambiar información entre la API y el frontend.

Abstract

The objective of Project 3 of Tecnologías Chapinas, S.A. is to develop a tool that analyzes the content of social networks to determine the sentiment of users towards companies and services. To achieve this, an API will be implemented using the Python language and the Flask framework, which will allow processing structured messages in XML format. The tool will classify messages as positive, negative or neutral, based on a dictionary of words that define the sentiment. In addition, regular expressions will be used to extract key information such as user names, locations and social networks. The response will be generated in XML format, presenting statistics on the total number of messages and their classification. The frontend will be developed with Django, allowing file uploads, queries and reporting. This comprehensive approach will not only facilitate data analysis, but will also provide the company with a clear view of

public perception, allowing it to make informed decisions and improve its services.

Keywords
API (Application Programming Interface): *Set of rules and protocols that enable communication between different software systems, facilitating the systems, facilitating integration and data exchange.*

Sentiment: *Analysis that determines whether a message is positive, negative or neutral, based on textual content, using based on textual content, using a dictionary of keywords representing each category.*

keywords that represent each category.

XML (Extensible Markup Language):

A file format that allows the structuring and structuring and storage of data in readable text, used in this project to exchange information project to exchange information between the API and the frontend.

Introducción

La transformación digital ha llevado a las empresas a buscar soluciones innovadoras que les permitan adaptarse a un entorno en constante cambio. En este contexto, el Proyecto 3 de Tecnologías Chapinas, S.A. tiene como objetivo desarrollar una herramienta integral que permita analizar el contenido de las redes sociales y evaluar el sentimiento de los usuarios respecto a empresas y servicios. Utilizando el Protocolo HTTP y el paradigma de programación orientada a objetos (POO), este proyecto se enfoca en la implementación de una API que facilita la interacción con los datos. Mediante el uso de archivos XML como insumos, el sistema es capaz de procesar mensajes estructurados y clasificarlos en categorías de sentimiento, ofreciendo así a la empresa una visión clara de la percepción pública. Además, se busca almacenar información de manera persistente y utilizar expresiones regulares para extraer contenido relevante, garantizando la eficacia y eficiencia del análisis. Este ensayo examina los componentes técnicos del proyecto, la metodología de desarrollo empleada y los beneficios potenciales para Tecnologías Chapinas, S.A.

Desarrollo del tema
Abordando el uso de Flask para construir la API, el uso de Python como lenguaje de programación, y cómo la programación orientada a objetos permite diseñar una estructura modular que facilita la expansión de funcionalidades.

- **Backend en Flask**: Explica cómo se ha configurado la API en Flask, permitiendo la recepción de mensajes en formato XML. Los métodos diseñados para esta API se encargan de leer, analizar y almacenar los datos de los mensajes en un archivo XML. Aquí también se podrían explicar brevemente los métodos implementados, como el de clasificación de sentimientos, y cómo estos métodos interpretan cada mensaje.

- **Almacenamiento en XML**: Profundiza en cómo el sistema almacena datos en archivos XML para mantener una base de datos persistente de los mensajes recibidos. La estructura de los archivos XML está diseñada para registrar detalles de cada mensaje y la clasificación del sentimiento (positivo, negativo o neutro), el cual se determina comparando palabras del mensaje con un diccionario de sentimientos.

Procesamiento y Análisis de Mensajes
Detalla el flujo de trabajo del procesamiento de los mensajes:

1. **Recepción de Mensajes**: La API recibe mensajes en una estructura específica que

incluye el usuario, la red social y el contenido. Esta estructura facilita la identificación de palabras clave y el análisis de sentimientos.

2. **Clasificación del Sentimiento**: El análisis del contenido se realiza mediante un diccionario de palabras clasificadas como positivas o negativas, y el sistema asigna un sentimiento según el predominio de términos. Este proceso incluye la normalización de palabras para ignorar variaciones como mayúsculas, minúsculas y tildes.

3. **Extracción de Empresas y Servicios**: Describe cómo el sistema identifica menciones de empresas y servicios utilizando alias en el diccionario. Esto permite registrar la relación entre el sentimiento y la empresa o servicio mencionado, lo cual es clave para el análisis de reputación.

Generación de Reportes y Consultas en el Frontend
Discute las funciones del frontend en Django, cuya interfaz permite al usuario cargar archivos XML para procesarlos, ver resúmenes de clasificaciones, y generar reportes. Expón cómo el sistema ofrece distintos tipos de consultas, como la clasificación de mensajes por fecha o rango de fechas, el total de menciones por empresa o servicio y un resumen de los resultados en PDF. Este enfoque asegura que la herramienta no solo clasifique los mensajes, sino que también permita a los usuarios visualizar los resultados de manera clara y organizada.

Diseño y Arquitectura del Proyecto
El diseño de este proyecto se ha estructurado en una arquitectura **cliente-servidor** donde el frontend se desarrolla en Django y el backend en Flask. El uso de esta arquitectura permite una clara separación entre las responsabilidades de presentación de la información y el procesamiento de datos.

- **Django en el Frontend**: La elección de Django facilita la creación de una interfaz web que es eficiente, segura y de fácil escalabilidad. Django incluye un conjunto de herramientas predefinidas que agilizan la gestión de bases de datos, autenticación y la creación de formularios, elementos que se utilizan para diseñar un sistema de carga de archivos XML, consulta de datos y generación de reportes para los usuarios.

- **Flask en el Backend**: Para el backend, se utiliza Flask debido a su flexibilidad y ligereza, lo que es ideal para crear una API RESTful. Flask permite una integración directa con herramientas de manejo de archivos XML y procesamiento de texto, siendo también compatible con bibliotecas de análisis y visualización de datos, lo que facilita la ampliación del sistema para posibles funcionalidades futuras.

Componentes Clave del Backend
El desarrollo del backend se basa en módulos específicos que organizan el procesamiento de los mensajes, el análisis de sentimientos, y la generación de reportes.

1. **Módulo de Recepción y Almacenamiento de Mensajes**: Este módulo recibe los mensajes enviados desde el frontend en un formato estructurado (XML) y los almacena en archivos XML. Aquí se define un esquema XML, asegurando que todos los datos se guarden de manera estandarizada, facilitando la consistencia en el análisis posterior. Además, el sistema realiza validaciones básicas de formato y contenido en el archivo XML antes de almacenarlo, mostrando errores en caso de datos incompletos o estructuras mal formadas.

2. **Análisis de Sentimientos y Clasificación**: Este es el núcleo del procesamiento de datos. Utilizando un diccionario de palabras positivas y negativas, este módulo analiza cada mensaje, identificando las palabras relevantes y clasificando el sentimiento como positivo, negativo o neutro. Para esto, el sistema emplea expresiones regulares, que permiten buscar palabras y variantes (mayúsculas, tildes) en el texto. Este enfoque garantiza una mayor precisión en la clasificación de sentimientos, aumentando la confiabilidad de los resultados.

3. **Extracción de Entidades (Empresas y Servicios)**: Este módulo permite identificar menciones de empresas y servicios en los mensajes mediante el uso de alias en un diccionario. Este proceso permite que el sistema no solo clasifique los mensajes en función del sentimiento, sino también identifique a qué entidad están referidos. Así, el sistema puede crear reportes específicos para cada empresa o servicio, una funcionalidad importante para el análisis de reputación.

4. **Generación de Resúmenes y Reportes**: Este módulo es responsable de preparar resúmenes estadísticos y reportes detallados basados en las clasificaciones de los mensajes. Además de la clasificación general de sentimientos, se genera un reporte detallado por empresa o servicio, que puede ser visualizado en el frontend. Este módulo también permite la generación de reportes en PDF para un análisis más formal y documentado.

Manejo de Almacenamiento con XML
El uso de XML como formato de almacenamiento facilita la compatibilidad y la portabilidad de los datos. Al usar un esquema XML específico, el sistema garantiza que los datos almacenados se ajusten a una estructura bien definida, lo que facilita el intercambio y procesamiento de estos datos en el futuro. La elección de XML también permite al sistema incorporar archivos de manera modular, permitiendo procesar grandes volúmenes de datos sin cargar excesivamente la memoria del sistema.

- **Estructura de los Archivos XML**: La estructura incluye nodos principales que registran detalles del mensaje, como la fecha, el usuario y el contenido, seguidos de la clasificación de sentimiento y la empresa/servicio mencionado. Este enfoque jerárquico permite realizar búsquedas más rápidas y específicas dentro del archivo.

- **Validación de Archivos XML**: Para asegurar la precisión, el sistema valida cada archivo XML al ser cargado, verificando que cumpla con el esquema y alertando al usuario en caso de errores de estructura o datos faltantes.

Pruebas y Validación
La metodología de desarrollo incluye un enfoque iterativo de pruebas para garantizar la precisión de la API. A medida que se implementan nuevas funcionalidades, cada módulo pasa por pruebas unitarias, pruebas de integración y pruebas de aceptación de usuario.

- **Pruebas Unitarias**: Se realizan pruebas específicas para cada módulo, como la verificación de las expresiones regulares en el análisis de sentimientos o la validación de alias de empresas en el módulo de extracción de entidades. Estas pruebas permiten identificar errores en cada módulo sin afectar el funcionamiento general.

- **Pruebas de Integración**: Una vez validados individualmente, los módulos se integran y se someten a pruebas de integración. Esto

permite verificar que el sistema completo funcione de manera coherente, garantizando que los datos se transfieran correctamente entre el frontend y el backend y que el almacenamiento en XML se realice sin problemas.

- **Pruebas Funcionales con Usuarios**: Las pruebas finales incluyen el uso de casos de prueba por parte de usuarios seleccionados, quienes validan la funcionalidad completa de la aplicación desde la carga de archivos XML hasta la generación de reportes. Esto permite verificar que la interfaz y el procesamiento cumplan con las expectativas de usuario final.

Plan de Documentación y Soporte
La documentación es esencial en este proyecto para asegurar la facilidad de mantenimiento y futuras mejoras. Este plan incluye manuales para desarrolladores, administradores y usuarios finales.

- **Documentación Técnica**: Se elaboran manuales técnicos para desarrolladores que describen la arquitectura de la API, los esquemas XML, y la configuración del entorno Flask-Django. También se documenta cada módulo, explicando su funcionalidad, métodos y parámetros clave.

- **Manual de Usuario**: El manual de usuario está diseñado para guiar a los usuarios finales en el uso de la aplicación, desde la carga de archivos XML hasta la generación de reportes. Este documento incluye capturas de pantalla y ejemplos de los tipos de reportes y resultados que se pueden generar.

Conclusiones

La arquitectura modular del código, dividida en módulos independientes para análisis de sentimientos, extracción de entidades, y generación de reportes, demuestra un diseño eficiente y escalable. Esta estructura permite que cada componente del sistema sea mantenido y mejorado individualmente sin afectar el funcionamiento general. Además, el uso de XML como formato de almacenamiento brinda una solución adecuada para el manejo de datos estructurados, facilitando la portabilidad y compatibilidad con otros sistemas.

El proyecto ha sido diseñado para poder adaptarse a las necesidades cambiantes de la empresa. La implementación de pruebas unitarias e integradas, junto con una documentación detallada, asegura una alta calidad en el producto final y simplifica futuras expansiones. La integración de tecnologías como Flask para el backend y Django para el frontend proporciona una base sólida, flexible y segura, que no solo satisface los requisitos actuales, sino que también permite la integración de nuevas funcionalidades en el futuro, como almacenamiento en bases de datos más robustas o mejora en los algoritmos de procesamiento de texto.

Anexos

Diagrama de clases

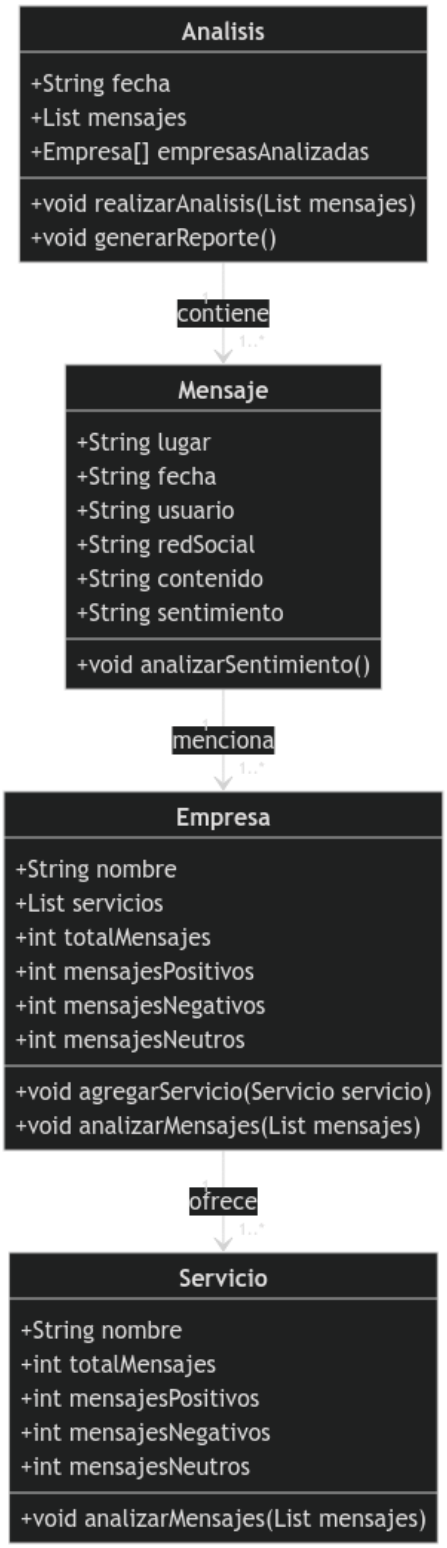

Diagrama de Actividades