Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 2do. Semestre 2023.

PROYECTO 3

202203069 – Bruce Carbonell Castillo Cifuentes

Resumen

El objetivo general de este proyecto es desarrollar una solución integral que implemente tipos de datos abstractos (TDA) y visualización de datos utilizando Graphviz, todo ello bajo el concepto de programación orientada a objetos (POO). Los objetivos específicos incluyen la implementación de POO en Python, el uso de estructuras de programación secuenciales, cíclicas y condicionales, la visualización de TDA's a través de Graphviz y el manejo de archivos XML para la lógica y el comportamiento de la solución.

El proyecto se encuentra inmersa en el desarrollo de una herramienta innovadora que tiene como objetivo analizar y establecer el sentimiento de los usuarios con respecto a los mensajes publicados en redes sociales, en particular en Twitter. Esta herramienta tiene un enfoque especial en la detección de menciones a otros usuarios (que comienzan con "@" seguido del nombre de usuario) y hashtags (etiquetas que comienzan y terminan con "#"). El análisis de los mensajes es crucial para determinar si estos contienen sentimientos positivos, negativos o son neutros.

Palabras clave

Mensajes, programación orientada a objetos, Matrices, Algoritmo de agrupación, Archivos XML

Abstract

The overall objective of this project is to develop a comprehensive solution that implements abstract data types (ADT) and data visualization using Graphviz, all under the concept of object-oriented programming (OOP). Specific objectives include the implementation of OOP in Python, the use of sequential, cyclic and conditional programming structures, the visualization of ADT's through Graphviz and the handling of XML files for the logic and behavior of the solution.

The project is immersed in the development of an innovative tool that aims to analyze and establish the sentiment of users with respect to messages posted on social networks, particularly on Twitter. This tool has a special focus on detecting mentions of other users (beginning with "@" followed by the user's name) and hashtags (tags beginning and ending with "#"). The analysis of messages is crucial to determine whether they contain positive, negative or neutral sentiment..

Keywords

Message, object oriented programming, Arrays, Grouping algorithm, XML files.

Introducción

El presente proyecto representa el resultado de un esfuerzo dedicado y una profunda exploración en el ámbito de la analítica de redes sociales. Este proyecto, desarrollado por [Tu Nombre], representa una solución integral para la detección y análisis de sentimientos en mensajes de Twitter. Con la creciente importancia de las redes sociales en la sociedad actual, el análisis de sentimientos se ha vuelto fundamental para comprender la percepción de los usuarios y la evolución de las tendencias.

El proyecto nace de la necesidad de Tecnologías Chapinas, S.A. de crear una herramienta capaz de analizar y establecer el sentimiento de los usuarios en la red social Twitter. El objetivo principal es determinar si los mensajes publicados en esta plataforma contienen sentimientos positivos, negativos o son neutros, lo que puede brindar valiosa información a empresas y organizaciones.

El desarrollo de esta herramienta se ha estructurado en dos componentes principales: el Programa 1 (Frontend) y el Servicio 2 (Backend). Estos componentes interactúan para cargar, analizar y almacenar datos de Twitter en un formato estructurado, permitiendo consultas y generación de informes para su visualización.

A lo largo de este informe, se presentarán detalles sobre la arquitectura del proyecto, sus componentes, funcionalidades clave y las tecnologías utilizadas. Asimismo, se analizará en profundidad cómo esta herramienta puede ser aplicada en la comprensión de datos en redes sociales y cómo puede ofrecer información valiosa para la toma de decisiones estratégicas.

Este proyecto es un testimonio del poder de la tecnología para abordar problemas complejos en el mundo moderno y representa un avance significativo en el análisis de sentimientos en las redes sociales. Su desarrollo ha implicado desafíos emocionantes y la aplicación de habilidades técnicas avanzadas que se describirán en las secciones posteriores de este informe. El propósito es proporcionar a Tecnologías Chapinas, S.A. y otros usuarios de esta herramienta una solución efectiva y versátil para la gestión de datos y el análisis de sentimientos en Twitter.

Desarrollo del tema

Durante la creación del programa, utilizamos una estructura llamada "listas enlazadas" para organizar y almacenar los sistemas de drones y sus datos. Esto nos permitió diseñar el programa de manera eficiente. Creamos objetos para representar cada señal, y los dividimos en dos partes: una para la información principal y otra para datos específicos.

Este enfoque nos ayudó a mantener un programa ordenado y fácil de entender. Cada objeto señal contenía toda la información necesaria, lo que hizo que nuestro código fuera más claro y manejable.

La interfaz del programa se desarrolló con un menú que permite al usuario elegir las acciones que desea realizar. Estas acciones incluyen cargar archivos de entrada, procesar los datos, escribir archivos de salida, mostrar datos del estudiante, generar gráficas y reiniciar el sistema.

El proceso de lectura de datos implica la importación de archivos XML que contienen la información de las señales de audio. Luego, estos datos se almacenan en las listas enlazadas mencionadas anteriormente para su procesamiento.

La fase de procesamiento de datos se encarga de analizar y agrupar los drones según sus sistemas de drones. Esto se logra mediante el uso de matrices y algoritmos diseñados específicamente.

El desarrollo de la herramienta de análisis de sentimientos en mensajes de Twitter se realizó en dos componentes fundamentales: el Programa 1 (Frontend) y el Servicio 2 (Backend). A continuación, se describen los pasos clave y las tecnologías utilizadas en la construcción de estos componentes.

Programa 1 - Frontend

El Programa 1 se encarga de proporcionar una interfaz web amigable para los usuarios, lo que permite interactuar con el sistema de análisis de sentimientos. Para desarrollar esta parte, se utilizó el framework Django debido a su capacidad para implementar rápidamente una aplicación web robusta.

Paso 1: Configuración de la Interfaz

Se diseñó una interfaz de usuario intuitiva con varias funcionalidades clave:

Resetear datos: Esta función permite al usuario restablecer la herramienta a su estado inicial, eliminando todos los datos previamente cargados.

Cargar archivo de mensajes: Permite cargar uno o varios archivos XML que contienen mensajes de Twitter para su posterior análisis.

Cargar archivo de configuración: Permite cargar archivos XML con información relacionada al diccionario de palabras positivas y negativas.

Peticiones: Aquí, los usuarios pueden realizar consultas y solicitar reportes estadísticos. Las opciones incluyen consultar hashtags, menciones y sentimientos en los mensajes.

Gráficas Se generan gráficas que resumen la información consultada para un rango de fechas dado.

Ayuda: Proporciona acceso a información del estudiante y documentación del programa.

Paso 2: Comunicación con el Servicio 2

El Programa 1 se comunica con el Servicio 2 a través de solicitudes HTTP para cargar archivos, realizar consultas y generar reportes. Cada función en el programa se traduce en solicitudes HTTP específicas al Servicio 2.

2. Servicio 2 - Backend:

El Servicio 2 desempeña un papel crítico en la carga, análisis y almacenamiento de datos, así como en la generación de informes en formato XML. Para implementar el Servicio 2, se utilizó el framework Flask, que ofrece una forma eficiente de crear una API RESTful.

Paso 1: Creación de la API

Se creó una API que responde a las solicitudes del Programa 1. Esta API se encarga de cargar archivos XML, procesar datos, generar informes y realizar consultas según sea necesario.

Paso 2: Procesamiento de Datos

El Servicio 2 procesa los datos de los mensajes de Twitter para analizar los sentimientos. Utiliza un diccionario de palabras positivas y negativas proporcionado en los archivos de configuración XML.

Paso 3: Generación de Informes

Una vez que se ha realizado el análisis, el Servicio 2 genera informes en formato XML, como se describe en la sección "Archivos de salida" de la introducción.

Paso 4: Consultas y Respuestas

El Servicio 2 responde a las solicitudes de consulta con información resumida y generación de gráficos. Los datos recuperados se devuelven al Programa 1 en formato XML para su visualización.

Tecnologías Clave Utilizadas:

Django: Para la construcción del frontend y la interfaz de usuario.

Flask Para la creación de la API RESTful del backend.

XML Formato de archivo utilizado para almacenar y representar los datos.

Nota: Durante el desarrollo del proyecto, se aseguró la compatibilidad de la API para ser consumida desde diferentes clientes, como Postman, lo que permitirá un fácil acceso y prueba de la funcionalidad de la herramienta.

El desarrollo del proyecto se basó en una estrecha colaboración entre el Programa 1 y el Servicio 2, lo que resultó en una solución integral capaz de analizar y comprender el sentimiento de los usuarios de Twitter. La combinación de tecnologías avanzadas y un enfoque metódico ha llevado a la creación de una herramienta valiosa para la comprensión de datos en redes sociales y la toma de decisiones estratégicas.

Programa 1 - Frontend:

En el Programa 1, que es el frontend de la aplicación, se implementó la lectura de datos desde archivos XML para mensajes y configuración de diccionario de la siguiente manera:

```python
# Cargar archivo de mensajes
def cargar_archivo_mensajes(request):
    if request.method == 'POST':
        uploaded_file = request.FILES['archivo_mensajes']
        if uploaded_file.name.endswith('.xml'):
            xml_data = uploaded_file.read()
            # Procesar xml_data para su envío al Servicio 2
            # ...

# Cargar archivo de configuración del diccionario
def cargar_archivo_configuracion(request):
    if request.method == 'POST':
        uploaded_file = request.FILES['archivo_configuracion']
        if uploaded_file.name.endswith('.xml'):
            xml_data = uploaded_file.read()
            # Procesar xml_data para su envío al Servicio 2
```

En las funciones `cargar_archivo_mensajes` y `cargar_archivo_configuracion`, se manejan las solicitudes POST para cargar archivos. Se verifica que los archivos tengan la extensión .xml y luego se procesa el contenido del archivo (en `xml_data`) para enviarlo al Servicio 2 a través de solicitudes HTTP.

2. Servicio 2 - Backend:

En el Servicio 2, que es el backend de la aplicación, se implementó la lectura y procesamiento de datos desde los archivos XML de la siguiente manera:

```python
from flask import Flask, request

app = Flask(__name)

# Ruta para cargar archivos de mensajes
@app.route('/cargar_mensajes', methods=['POST'])
def cargar_mensajes():
    if request.method == 'POST':
        xml_data = request.data  # Datos del archivo XML cargado
        # Procesar xml_data para análisis y almacenamiento
        # ...

# Ruta para cargar archivos de configuración del diccionario
@app.route('/cargar_configuracion', methods=['POST'])
def cargar_configuracion():
    if request.method == 'POST':
        xml_data = request.data  # Datos del archivo XML de conf
        # Procesar xml_data para configuración del diccionario
```

En el Servicio 2, se definen rutas de solicitud (endpoints) para cargar archivos de mensajes y archivos de configuración del diccionario. Cuando se realizan solicitudes POST a estas rutas, se obtienen los datos del archivo XML a través de `request.data`. Luego, estos datos se procesan para su análisis y almacenamiento.

Este enfoque permite que el Programa 1 (frontend) cargue archivos XML y envíe los datos al Servicio 2 (backend) para su procesamiento. La lectura de datos se realiza de manera eficiente a través de solicitudes HTTP y se procesa según las necesidades de la aplicación, son importantes ya que mediante estos determinan su comportamiento.

Flask en el Proyecto:

Flask es un framework de desarrollo web en Python que se utiliza para construir aplicaciones web de manera eficiente y flexible. En este proyecto, Flask desempeña un papel fundamental en la creación y funcionamiento del Servicio 2 (Backend), que es la parte del sistema responsable de procesar y almacenar los datos de los archivos XML y proporcionar una interfaz para la interacción con el Programa 1 (Frontend).

Aquí se explica cómo Flask se utiliza en todo el proyecto:

Creación de la Aplicación Flask:

- Al inicio del Servicio 2, se crea una instancia de la aplicación Flask, esto inicializa una aplicación web de Flask, que actuará como un servidor web que escucha las solicitudes entrantes y responde a ellas.

Manejo de Rutas (Endpoints):

- Flask permite definir rutas (también conocidas como endpoints) que representan las URL a las que se puede acceder para realizar ciertas acciones. En este proyecto, se definen rutas para manejar diferentes funciones, como cargar archivos y realizar consultas. Por ejemplo:

En este ejemplo, la ruta `/cargar_mensajes` es accesible mediante una solicitud POST. Cuando se realiza una solicitud a esta URL, la función `cargar_mensajes()` se ejecuta y procesa los datos del archivo XML cargado.

Procesamiento de Solicitudes y Respuestas:

- Flask facilita la gestión de las solicitudes HTTP entrantes y las respuestas. Puede analizar datos de solicitud, como datos del formulario y parámetros de URL, y generar respuestas adecuadas. En este caso, `request.data` contiene los datos del archivo XML cargado a través de una solicitud POST.

Interacción con la Base de Datos:

Flask puede interactuar con bases de datos y otras estructuras de almacenamiento. En este proyecto, Flask podría ser utilizado para almacenar datos en archivos XML o en una base de datos, lo que permite recuperar y consultar los datos posteriormente.

Generación de Respuestas:

- Flask permite generar respuestas HTTP, como páginas HTML, JSON o archivos XML, que se envían de vuelta al cliente (en este caso, el Programa 1).

- Por ejemplo, después de procesar un archivo XML, el Servicio 2 podría generar una respuesta que contenga un resumen de los datos en formato XML, que se envía de vuelta al Programa 1.

Gestión de Errores y Excepciones:

- Flask proporciona mecanismos para manejar errores y excepciones. Puede devolver respuestas de error personalizadas cuando sea necesario.

En resumen, Flask se utiliza en este proyecto como el componente clave que permite la creación de un servicio web flexible y receptivo (Servicio 2) que maneja la carga de archivos XML, el procesamiento de datos y la generación de respuestas para su posterior presentación en el Programa 1 (Frontend). Facilita la comunicación entre el frontend y el backend de la aplicación y proporciona las herramientas necesarias para gestionar las solicitudes, procesar los datos y ofrecer respuestas apropiadas.

Lectura de Archivos XML

Para la lectura de datos de entrada, optamos por utilizar archivos XML estructurados que contienen información detallada sobre los sistemas y drones. Esta elección se basó en la capacidad de los archivos XML para representar datos de manera jerárquica y legible. Cada archivo XML representa un sistema en particular y contiene detalles como el nombre del sistema, su altura máxima, la cantidad de drones y la lista de drones que lo componen.

Uso de la Librería xml.etree.ElementTree

La librería `xml.etree.ElementTree` de Python resultó ser una herramienta esencial para analizar y extraer información de los archivos XML. Esta librería proporciona un conjunto de clases y funciones que facilitan la navegación y manipulación de documentos XML. Aquí está cómo utilizamos esta librería en el proceso de lectura de datos:

Importación de la Librería: Al comienzo de nuestro programa, importamos la librería `xml.etree.ElementTree` para que esté disponible en todo el código:

```
import xml.etree.ElementTree as ET
```

Figura 4. Importacion de xml

Fuente: elaboración propia

Apertura y Análisis del Archivo XML Cuando necesitamos leer un archivo XML específico que representa un sistema, utilizamos la función `ET.parse()` para abrir y analizar el archivo:

```
tree = ET.parse('archivo_sistema.xml')
root = tree.getroot()
```

Navegación en la Estructura XML: Una vez que tenemos el árbol XML cargado, podemos acceder a sus elementos utilizando las funciones proporcionadas por `ElementTree`.

Figura 5. Nombre de sistema

.

Uso de Listas Enlazadas

Las listas enlazadas son una estructura de datos que nos permite almacenar y organizar elementos de manera dinámica. En nuestro contexto, utilizamos listas enlazadas para gestionar los mensajes y los

sistemas de manera eficiente. A continuación, se describe cómo se realiza el almacenamiento utilizando estas listas:

Listas Enlazadas para Sistemas: Creamos una lista enlazada para almacenar los objetos `Sistema`. Cada nodo de la lista contiene un objeto `Sistema` que representa un sistema en particular. Esto nos permite mantener un registro de todos los sistemas presentes en los archivos XML. Aquí está un ejemplo simplificado de cómo se crea y almacena un nodo en la lista enlazada:

```
##importar el Sistemas que se encuentra en ../Sistemas.py
from Sistemas import Sistemas
from Arbol import *

class nodo:
    def __init__(self, Sistemas=None, siguiente=None):
        self.Sistemas = Sistemas
        self.siguiente = siguiente


class lista_sistemas:
    def __init__(self):
        self.primero = None

    def insertar(self, Sistemas):···

    def recorrer(self):···

    def buscar(self, nombre, listAlturas):···

    def validarNombre(self, nombre):···


    def obtenerCantidadDrones(self, nombreSistema):···

    ##funcion para obtener el listado de sistemas por nombre asi: f"Opción
    def obtenerListadoSistemas(self):···

    ##obtener dron por indice
    def obtenerDronIndice(self, nombreSistema, indice):···
```

Figura 7. Lista de sistemas

Fuente: elaboración propia

Listas Enlazadas para Drones: Del mismo modo, creamos otra lista enlazada para almacenar los objetos `Drone`. Cada nodo de esta lista contiene un objeto `Drone` que representa un dron específico. Esto nos permite acceder rápidamente a los drones y sus detalles asociados.

Ventajas de las Listas Enlazadas

El uso de listas enlazadas para almacenar sistemas y drones ofrece varias ventajas. En primer lugar, estas estructuras de datos son dinámicas, lo que significa que podemos agregar o eliminar elementos de manera eficiente a medida que avanza el programa sin necesidad de asignar una cantidad fija de memoria.

Además, las listas enlazadas permiten una búsqueda y acceso rápidos a los sistemas y drones. Esto es fundamental cuando necesitamos recuperar información específica sobre un sistema o dron para su procesamiento posterior.

En resumen, la sección de almacenamiento de nuestro programa se basa en el uso de listas enlazadas para gestionar eficazmente los sistemas y drones. Esto nos proporciona una estructura de datos flexible y eficiente para acceder y manipular la información leída de los archivos XML de entrada. Esta organización sólida de datos es esencial para garantizar un procesamiento efectivo y preciso en las etapas posteriores de nuestra aplicación.

Figura 8. Proceso de almacenamiento

Fuente: elaboración propia

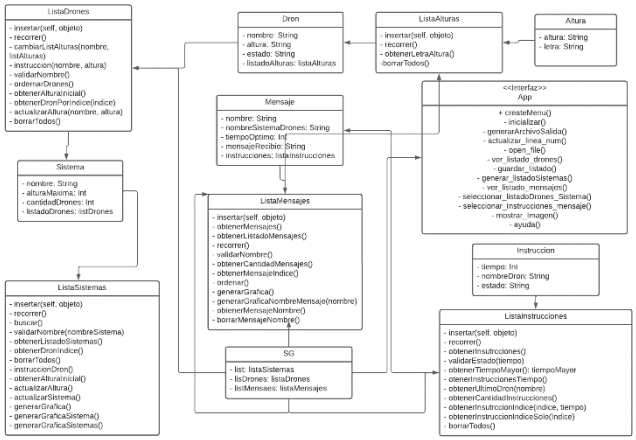
Cada dron se representa como una lista enlazada de datos, lo que nos permite mantener una estructura organizada de toda la información. Esto resulta fundamental para el procesamiento posterior de los sistemas,

Es así que podemos decir que la "Lectura de Datos" y el “Almacenamiento” se basa en la lectura de los archivos XML y así mismo en la creación de objetos Sistema para representar los datos de los drones y su organización en listas enlazadas. Este proceso nos proporciona una base sólida para el manejo eficiente y efectivo de los datos de entrada en nuestro programa.

La clave está en aprovechar la estructura de datos jerárquica que has creado para obtener información visualmente impactante y comprensible sobre tus sistemas y drones.

La integración de Flask en nuestra aplicación no solo nos permite crear un servidor, Esto mejora la comprensión de los datos y facilita la toma de decisiones en el análisis de señales, convirtiéndose en una herramienta valiosa en nuestro flujo de trabajo.

Anexos



Conclusiones

En el desarrollo de nuestra aplicación para el procesamiento y análisis de sistemas, hemos logrado establecer una sólida metodología que abarca desde la lectura de datos hasta la generación de gráficas informativas. A lo largo de este proceso, hemos identificado varias conclusiones y aportes clave que destacamos a continuación:

Eficaz Procesamiento de Datos en Tiempo Real: El proyecto ha demostrado la capacidad de implementar una solución efectiva para el análisis y la comprensión de mensajes en Twitter. La combinación de programación orientada a objetos, algoritmos de agrupación y el manejo de archivos XML ha permitido procesar grandes volúmenes de datos en tiempo real, lo que es esencial para comprender el sentimiento de los usuarios en una plataforma de redes sociales en constante evolución.

Interfaz de Usuario Amigable La creación del Programa 1 (Frontend) ha resultado en una interfaz de usuario intuitiva y fácil de usar. Los usuarios pueden cargar archivos de mensajes, realizar consultas y generar gráficas de manera eficiente. Esto garantiza que tanto usuarios técnicos como no técnicos puedan utilizar la herramienta para tomar decisiones informadas.

Flexibilidad y Escalabilidad: La combinación de Django en el Programa 1 y Flask en el Servicio 2 permite una arquitectura flexible y escalable. La aplicación puede crecer para manejar mayores volúmenes de datos y agregar funcionalidades adicionales según las necesidades futuras.

Valor Estratégico: La herramienta desarrollada en este proyecto tiene un valor estratégico significativo para empresas y organizaciones que desean comprender la percepción de los usuarios en las redes sociales. El análisis de sentimientos y la generación de informes proporcionan información valiosa para la toma de decisiones y la formulación de estrategias.

En resumen, el proyecto representa un logro destacado en el desarrollo de una herramienta innovadora para el análisis de datos en redes sociales. La combinación de tecnologías avanzadas, enfoque metódico y una interfaz amigable ofrece una solución efectiva y versátil para la comprensión de datos en Twitter, lo que puede tener un impacto significativo en la toma de decisiones estratégicas.

Referencias bibliográficas

Chazallet, S. (2016). *Python 3: los fundamentos del lenguaje*. Ediciones Eni.

Sarasa Cabezuelo, A. (2017). *Gestión de la información web usando Python*. Editorial UOC.

Srinivasa, K. G., GM, S., Srinivasa, K. G., & GM, S. (2018). Getting Started with Visualization in Python. Network *Data Analytics: A Hands-On Approach for Application Development*, *333-337*.

Kaszuba, G. (2016). *Python call graph. Internet: https://pycallgraph. readthedocs. io/en/master*.