Proyecto Final EDA – Trending Topics Analyzer

Este proyecto implementa un sistema completo para analizar *trending topics* sobre un flujo continuo de documentos.

El backend está desarrollado en C++ usando bloques temporales con el algortimo Space-Saving, y el frontend en Flask (Python).

1. Requisitos del sistema

1.1 Dependencias generales

Asegurese de tener esto previamente instalado:
- Python 3.9+
- pip
- g++ o clang++ con soporte C++17
- Make
- macOS o Linux

Si usa Windows, ejecutar en WSL2 (Ubuntu recomendado).

1.2 Crear entorno virtual listo

En la terminal, navegar hasta el directorio principal del proyecto y ejecutar:

```
chmod +x setup.sh
./setup.sh
```

Este script creará el entorno virtual en Python necesario para correr Flask e instalará todas sus dependencias necesarias.
Debe recibir el mensaje: `Entorno instalado correctamente`, eso indica que puede continuar.

2. Ejecutar el proyecto

En la terminal, navegar hasta el directorio principal del proyecto.
Ejecutar:

```
chmod +x exeApp.command
./exeApp.command
```

Este script primero verifica que el entorno virtual de Python exista y esté activado, luego inicia automáticamente el backend en C++ y el servidor Flask en puertos separados. Si todo funciona correctamente, deberá tener dos terminales (frontend y backend) con los mensajes clave:

1. `[Backend] Esperando conexiones en puerto 8081...` indicando que el backend está activo.
2. `Running on http://127.0.0.1:8080` indicando que la aplicación web Flask está levantada.

Si alguno de estos mensajes no aparece, significa que hubo un error en la ejecución.

Después de eso, solo accede a http://127.0.0.1:8080 y deberá visualizar la interfaz.

Si encontrara algún error en la ruta, modifique manualmente los comando para apuntar al directorio donde tenga almacenado el proyecto.

2. Estructura del proyecto

```
ProyectoFinal/
├── setup.sh
├── exeApp.command
│
├── backend/
│   ├── main.cpp
│   ├── Makefile
│   ├── gestorBloques/
│   ├── estructuras/
│
├── frontend/
│   ├── env/
│   ├── static/
│   ├── templates/
│   ├── preprocessign.py
│   ├── webapp.py
│   ├── wordCloud.py
```

```
│
├──── data/
│   ├── ins1/
│   ├── ins2/
│   ├── ins3/
│   └── ...
│
└── README.md
```

> Importante: las rutas deben mantenerse exactamente así.

3. Uso de la aplicación

3.1 Carpetas de datos

El sistema utiliza automáticamente las carpetas:

```
/data/ins1/
/data/ins2/
/data/ins3/
...
```

Cada carpeta contiene archivos `.txt` con noticias o documentos que serán preprocesados e insertados como un único bloque en el backend.

La aplicación avanzará secuencialmente: primero `ins1`, luego `ins2`, luego `ins3`, etc. Si una carpeta no existe, se detendrá el proceso de inserción.

3.2 Inserción de datos — Funcionamiento detallado

Al presionar el botón **Insertar**, ocurre lo siguiente:

1. **Selección automática del directorio**

   La aplicación detecta cuál es el siguiente directorio `insX` disponible.

   Si ya se procesó `ins1`, la siguiente inserción tomará `ins2`, y así sucesivamente.

2. **Lectura de archivos**

   Se procesan todos los `.txt` dentro de esa carpeta.

   Se intenta leer cada archivo primero en UTF-8, luego en ISO-8859-1.

Si ambos fallan, el archivo se omite sin detener la inserción.

3. **Preprocesamiento avanzado** (en Python)
   - limpieza de caracteres no deseados
   - normalización
   - tokenización
   - eliminación de stopwords
   - stemming / lematización
   - reconstrucción segura de JSON
   - límites automáticos para evitar strings rotos

   El resultado es un listado de tokens completamente limpios y listos para el backend.

4. **Envío al backend**

   Los tokens se envían por sockets al backend C++ con manejo de mensajes fragmentados.

   El backend recibe los tokens, crea un **nuevo bloque temporal** y lo inserta al gestor.

5. **Actualización en la interfaz**

   Se mostrará:
   - cantidad de tokens enviados
   - tiempo de preprocesamiento
   - confirmación del backend
   - ID del bloque creado

3.3 Consultas Top-K — Uso y modos especiales

La interfaz cuenta con dos *sliders* (Inicio y Fin) que detallan la Ventana Temporal (bloques), simulando una línea de tiempo de cada inserción: selecciona cuántos bloques recientes incluir en la consulta.

Adicionalmente se tiene que ingresar K: tamaño del Top-K solicitado

Y existen dos modos de uso:

Modo Manual

El usuario mueve los sliders y presiona el botón **Consultar**.

La consulta se envía al backend y se muestra:

- tiempo de consulta
- listado de tópicos trending
- word cloud generada automáticamente

Modo Automático (Sliders fijados)

Si el usuario presiona el botón **Fijar**, los sliders quedan bloqueados.

Esto activa un comportamiento especial:

- cada vez que se inserte un nuevo bloque, se ejecutará **automáticamente una consulta Top-K**
- la interfaz actualiza inmediatamente la tabla y la word cloud

Esta característica es útil para:

- demostraciones en tiempo real
- monitoreo continuo
- análisis de streams simulados

3.4 Word Cloud

- se genera automáticamente después de cada consulta
- utiliza el conteo real proveniente de la ventana seleccionada
- se guarda temporalmente y se muestra en la interfaz sin necesidad de recargar la página

> Para ver más detalles sobre la lógica y el funcionamiento interno del producto, revisar el informe detallado.

4. Detalles técnicos importantes

4.1 Comunicación Backend/Frontend

- Backend → puerto **8081**
- Frontend → puerto **8080**

4.2 Manejo de codificación

El lector intenta:
1. UTF-8
2. ISO-8859-1
3. Si falla el archivo es omitido

4.3 Robustez del sistema

- Reconstrucción de JSON fragmentado
- Saltar archivos corruptos
- Diccionario con HashMap dinámico
- Validación de carpetas

5. Solución de problemas comunes

Backend no recibe datos
Asegurar que el backend corre en 8081.

Connection reset by peer
El backend se cerró durante el envío.

Tokens cortados
Corregido con el acumulador de JSON.

Valores “undefined”
Frontend no recibió JSON válido.

6. Licencia

Proyecto académico para evaluación en la Universidad Católica San Pablo.