TECNOLOGICO DE SOFTWARE

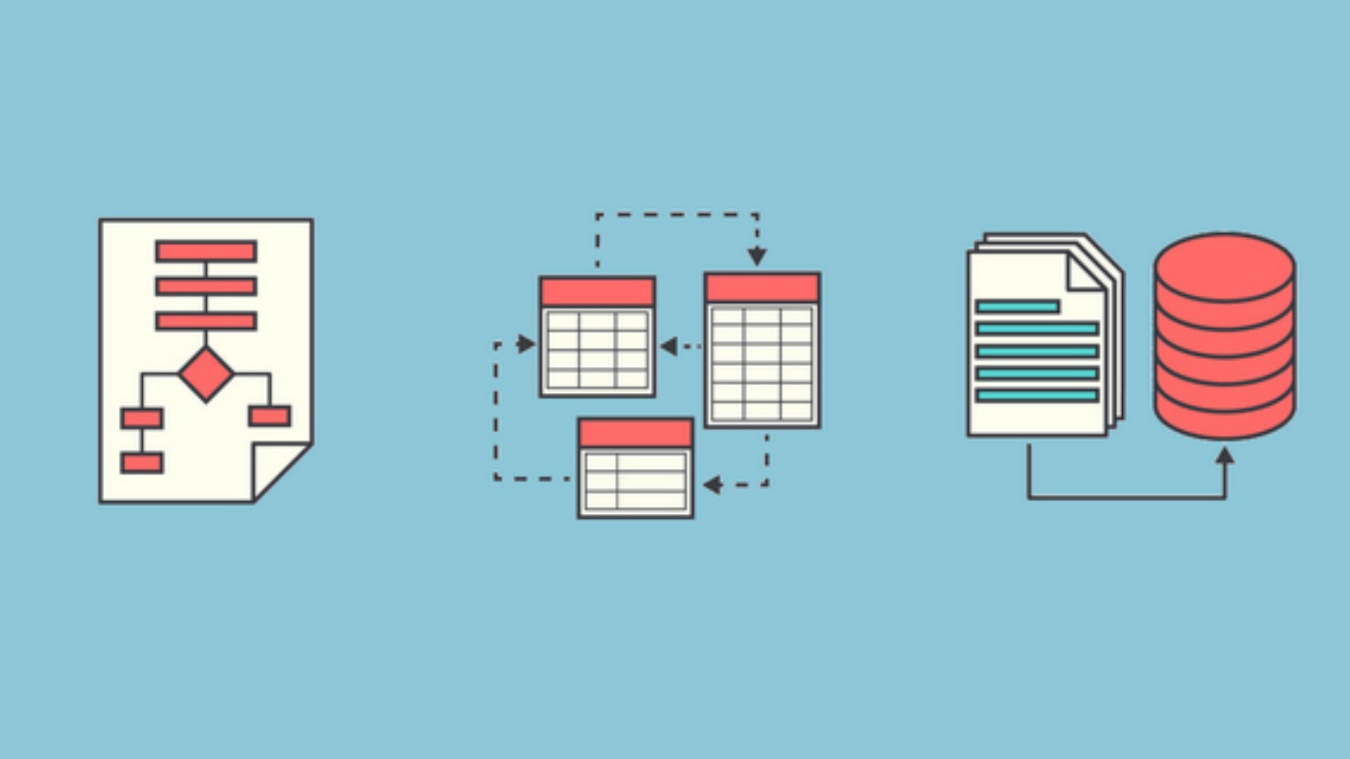
ESTEBAN PRIEGO

EVERTH KANTUN

JOSE CASTAÑEDA

ESTRUCTURA DE DATOS

2C



Este proyecto consiste en el desarrollo de un **Sistema de Detección de Similitud de Textos**, diseñado para comparar documentos en formato .txt con el objetivo de identificar posibles coincidencias o similitudes, lo cual resulta útil en contextos como la detección de plagio académico. El sistema está implementado en Python y utiliza diversas técnicas eficientes de procesamiento de texto y estructuras de datos avanzadas.

**Descripción General del Proyecto**

El sistema carga automáticamente todos los archivos de texto ubicados en una carpeta específica. Luego, realiza un preprocesamiento que incluye la limpieza del texto (eliminando signos de puntuación y convirtiendo todo a minúsculas) y la generación de n-gramas, es decir, secuencias de *n* palabras consecutivas. Estos n-gramas se transforman en valores hash usando SHA-256, y se almacenan en una tabla hash, con la opción de utilizar también un **filtro de Bloom** para mejorar la eficiencia en la búsqueda.

Para comparar documentos, se aplica la **Similitud de Jaccard**, que mide el grado de coincidencia entre dos conjuntos de n-gramas. Posteriormente, los resultados de similitud entre cada par de documentos se ordenan usando el algoritmo **Merge Sort**, elegido por su eficiencia y estabilidad.

El sistema incluye un **menú interactivo** por consola donde el usuario puede:

1. Cargar los documentos y visualizar los pares más similares en forma de tabla.
2. Ver un **grafo de similitud**, donde los nodos representan documentos y las conexiones indican el grado de similitud.

**Instrucciones de Instalación y Ejecución**

Para ejecutar el proyecto, primero se debe clonar el repositorio y luego instalar las dependencias necesarias (pybloom-live, networkx, matplotlib, y tabulate) usando pip. Los documentos a comparar deben colocarse en una carpeta (por defecto es: C:/Users/esteb/OneDrive/Desktop/txt\_docs). Finalmente, el sistema se ejecuta desde consola con el comando python main.py.

**Ejemplo de Uso**

Al ejecutar el sistema, el usuario elige una opción del menú. Si selecciona la opción de cargar documentos, el sistema analiza las similitudes y muestra los resultados en una tabla, indicando el porcentaje de coincidencia entre documentos. Si elige la visualización gráfica, se genera un grafo con los documentos y su nivel de relación visualizado mediante conexiones y etiquetas.

**Organización del Código**

El código está **modularizado** y **comentado**. Cada función cumple una tarea específica:

* cargar\_documentos(): carga todos los textos desde una carpeta.
* limpiar\_y\_tokenizar(): realiza el preprocesamiento.
* crear\_tabla\_hash(): genera la tabla hash y aplica opcionalmente un filtro de Bloom.
* similitud\_jaccard(): calcula la similitud entre documentos.
* merge\_sort(): ordena los resultados.
* mostrar\_tabla() y visualizar\_grafo(): presentan los resultados de manera visual.
* menu(): permite al usuario interactuar con el sistema fácilmente.

Esto garantiza que el sistema sea **reutilizable, escalable y mantenible**.

**Replicación de Resultados**

Para replicar los resultados del análisis, solo se requiere colocar los documentos .txt en la carpeta indicada y ejecutar el sistema. Es importante que los textos tengan contenido suficiente (al menos una cuartilla) para generar n-gramas relevantes. También se puede ajustar el parámetro n para trabajar con diferentes tamaños de n-gramas.

**Visualización de Resultados**

El sistema presenta los resultados de dos maneras:

* Una **tabla tabulada** con los documentos más similares, usando porcentajes claros.
* Un **grafo de similitud** que muestra las conexiones entre los documentos y su grado de relación.

Ambas visualizaciones permiten entender fácilmente qué documentos comparten más contenido y cómo se relacionan entre sí.

**Historial de Desarrollo**

Durante el desarrollo del sistema se realizaron múltiples avances, reflejados en los *commits*. Estos incluyen la carga inicial del proyecto, la implementación de funciones clave como el cálculo de similitud, el uso del algoritmo Merge Sort, la visualización de resultados y la creación del menú interactivo. Este historial demuestra una evolución continua del código, asegurando su funcionalidad y eficiencia.