Fuzzy Matching

Integrantes:
- Mayerli Santander Sejas
- Gabriela Trujillo Lozada
- Isaias Rojas Condarco

2. Review which matching algorithm is implemented.

El principal algoritmo de coincidencia es una combinación de coincidencia basada en tokens y coincidencia de umbral.

El algoritmo de coincidencia basado en tokens funciona de la siguiente manera:
1. Los elementos son pre procesados utilizando funciones de preprocesamiento predefinidas o personalizadas según su tipo (por ejemplo, nombre, dirección, correo electrónico, teléfono, etc.).
2. Los elementos preprocesados se tokenizan utilizando funciones de tokenización predefinidas o personalizadas, que dividen los elementos en unidades más pequeñas llamadas tokens.
   - Por ejemplo, wordTokenizer() divide los elementos en palabras individuales, mientras que triGramTokenizer() crea trigramas (subcadenas de 3 caracteres) a partir de los elementos.
   - wordSoundexEncodeTokenizer() aplica la codificación Soundex a los tokens, lo que ayuda a hacer coincidir tokens basándose en su similitud fonética.
3. Luego, los tokens se comparan utilizando coincidencia por igualdad o coincidencia de vecinos más cercanos, dependiendo del tipo de elemento y la configuración.
   - La coincidencia por igualdad verifica si hay coincidencias exactas entre los tokens.
   - La coincidencia de vecinos más cercanos encuentra tokens dentro de un rango de vecindad especificado (por ejemplo, 0.9 para una similitud del 90%).

El algoritmo de coincidencia de umbral se aplica además de la coincidencia basada en tokens:
1. Después de que los tokens se comparan, se calcula una puntuación de coincidencia para cada par de elementos basándose en el número de tokens coincidentes y el número total de tokens.
2. La puntuación de coincidencia se compara con un valor de umbral predefinido para cada tipo de elemento.
   - Si la puntuación supera el umbral, los elementos se consideran una coincidencia.
   - El umbral se puede ajustar para controlar la sensibilidad del proceso de coincidencia.

El enfoque basado en tokens permite la coincidencia basada en palabras individuales, n-gramas o similitud fonética, mientras que la coincidencia de umbral agrega una capa adicional de control para ajustar los resultados de la coincidencia.

El proyecto Fuzzy-Matcher también proporciona varias funciones de preprocesamiento y tokenización listas para usar, como:
- namePreprocessing(): Preprocesa los elementos de nombre eliminando caracteres especiales y convirtiendo a minúsculas.
- addressPreprocessing(): Preprocesa los elementos de dirección eliminando caracteres especiales y convirtiendo a minúsculas.
- wordTokenizer(): Tokeniza los elementos en palabras individuales.
- triGramTokenizer(): Tokeniza los elementos en trigramas.
- wordSoundexEncodeTokenizer(): Aplica la codificación Soundex a los tokens de palabras.

En general, el algoritmo de coincidencia principal en el proyecto Fuzzy-Matcher es un enfoque híbrido que combina la coincidencia basada en tokens con la coincidencia de umbral.
Esto permite un fuzzy matching flexible y configurable de varios tipos de elementos, teniendo en cuenta diferentes medidas de similitud y umbrales.

a) Are there any other matchings that can be implemented?
Sí, hay otros algoritmos de coincidencia que se pueden implementar para mejorar las capacidades de fuzzy matching del proyecto.
Algunas posibilidades incluyen:

- Distancia de Levenshtein: mide el número mínimo de ediciones de un solo carácter necesarias para transformar una cadena en otra.

b) If so, what will be the idea, and what will be its use?
La idea sería seleccionar el algoritmo de coincidencia más adecuado según el tipo de datos y los requisitos de precisión del sistema. Por ejemplo:

Para datos textuales con errores tipográficos, podría ser útil implementar un algoritmo basado en distancias como la de Levenshtein, también podría ser útil para hacer coincidir documentos de texto más largos.

c) If we need to add another matching algorithm, review and explain how that can be done.
1. Creando una nueva clase o método que implemente el algoritmo de coincidencia deseado, tomando dos elementos o cadenas como entrada y devolviendo una puntuación de similitud.
2. Modifica las clases relevantes (por ejemplo, ElementMatch, Element) para incluir el nuevo algoritmo de coincidencia como una opción, utilizando parámetros de configuración o indicadores.

3. Actualiza el proceso de coincidencia para incorporar el nuevo algoritmo basándose en la opción seleccionada, utilizando bloques condicionales o declaraciones switch.
4. Modifica las funciones de puntuación para calcular la puntuación de los elementos basándose en el algoritmo de coincidencia seleccionado y cualquier otro factor relevante.
5. Proporciona opciones de configuración adecuadas o métodos de la API para permitir a los usuarios elegir el algoritmo de coincidencia deseado según sus requisitos.