Estructura de Datos y Algoritmos

ITBA 2024-Q2

TP 2A- Ejer 11-1

¿Tiene Java 8 manejo de Strings Matching?

¿Tiene Java estos algoritmos (Soundex, Metaphone, Levenshtein, Q-Grams)?

String Matching

¿Existen bibliotecas externas con estas implementaciones?

Imaginarse que están solos liderando un Proyecto y precisan usar esas funcionalidades. Buscar qué biblioteca/s de **Apache Commons** sirven para eso (analizar en https://commons.apache.org/) y tienen un manejo avanzado de Strings.

TP 2A- Ejer 11-2

Crear un nuevo proyecto mvn que utilice las dependencias para manejo de Strings de apache commons.

Usar dichas implementaciones para detectar similitud entre 2 strings por medio de: Soundex, Levenshtein, Qgrams

Donde se puede, agregar métodos para extraer la máxima info posible. Ej: de soundex el enconding, de qGrams printTokens, etc

Verificar que lo obtenido coincide con la implementación de Uds.

Es decir, se busca que el código permita obtener como mínimo:

- El soundex(“maven”), soundex(“meibem”) y la similitud de ambos, según soundex, que en este caso es 1.
- El soundex(“threshold”) y soundex(“hold”) y la similitud de ambos, según soundex, que en este caso es 0.
- El soundex(“hold”) y soundex(“joul”) y la similitud de ambos, según soundex, que en este caso es 0.5
- LevenshteinDistance(“exkusa”, “ex-amigo”) y la similitud de ambos que es 1-6/8, o sea 0.25

Pista:

Buscar en: **codec** y **text**

¿Qué no pudieron hacer?

Como habrán observado, no hay tratamiento de Q-Grams.

- Tuvieron que incluir para Soundex (encoding):

```
<!-- https://mvnrepository.com/artifact/commons-codec/commons-codec -->
<dependency>
   <groupId>commons-codec</groupId>
   <artifactId>commons-codec</artifactId>
   <version>1.15</version>
</dependency>
```

- Tuvieron que incluir para Levenshtein (similitud):

```
<!-- https://mvnrepository.com/artifact/org.apache.commons/commons-text -->
<dependency>
   <groupId>org.apache.commons</groupId>
   <artifactId>commons-text</artifactId>
   <version>1.9</version>
</dependency>
```

Es común que tengamos que utilizar más de una biblioteca en nuestras aplicaciones. Por eso, la ventaja de usar Maven…

Analizaron cómo obtener para Levenshtein:

- Cantidad de sustituciones necesarias? Cantidad de borrados? Cantidad de inserciones?

Analizaron cómo obtener para Levenshtein:

- Cantidad de sustituciones necesarias? Cantidad de borrados? Cantidad de inserciones?

Rta: lo pueden hacer utilizando la clase LevenshteinDetailedDistance

TP 2A- Ejer 12

Usar distancia de Levenshtein para calcular la similitud en Metaphone (que Uds. no lo implementaron from scratch, pero ahora lo tienen gracias a Apache commons!)

TP 2A- Ejer 13

Cómo implementó Apache commons Soundex, Metaphone y Levenshtein

Muy distinto a lo que Uds. Pensaron?

Ej: busquen en google
apache commons Soundex.java

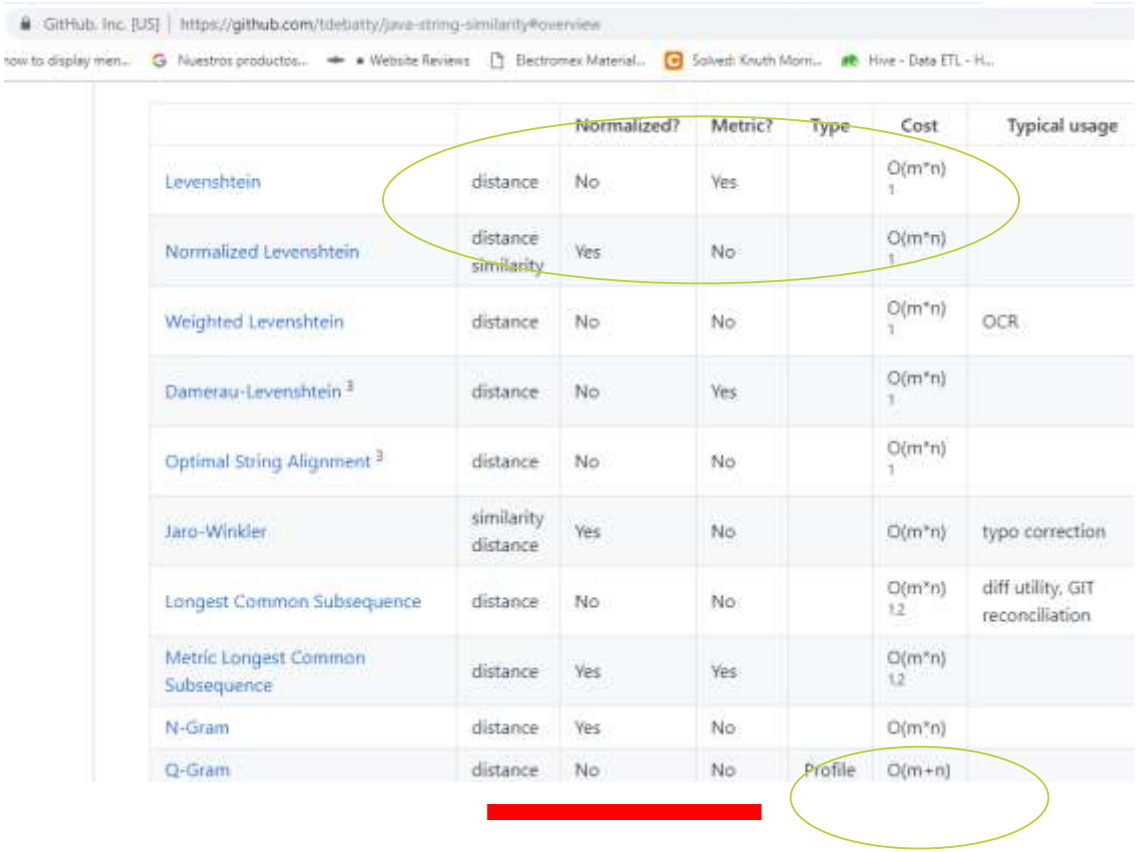
¿Existe alguna otra biblioteca que nos permira trabajar con QGrams?

Usando varias, llegaríamos al objetivo…

En integración de bibliotecas, Maven ayuda…

Agregar al proyecto maven la biblioteca **java-string-similarity**. Buscarla en Maven e incluir su dependencia.

Si se fijan en su documentación



GitHub, Inc. [US] | https://github.com/tdebatty/java-string-similarity#overview

how to display men... | Nuestros productos... | Website Reviews | Electromex Material... | Solved: Knuth Morri... | Hive - Data ETL - H...

		Normalized?	Metric?	Type	Cost	Typical usage
Levenshtein	distance	No	Yes		O(m*n) [1]	
Normalized Levenshtein	distance similarity	Yes	No		O(m*n) [1]	
Weighted Levenshtein	distance	No	No		O(m*n) [1]	OCR
Damerau-Levenshtein [3]	distance	No	Yes		O(m*n) [1]	
Optimal String Alignment [3]	distance	No	No		O(m*n) [1]	
Jaro-Winkler	similarity distance	Yes	No		O(m*n)	typo correction
Longest Common Subsequence	distance	No	No		O(m*n) [1,2]	diff utility, GIT reconciliation
Metric Longest Common Subsequence	distance	Yes	Yes		O(m*n) [1,2]	
N-Gram	distance	Yes	No		O(m*n)	
Q-Gram	distance	No	No	Profile	O(m+n)	

TP 2A- Ejer 14-1 y 14-2

Agregar al proyecto anterior la dependencia de **java-string-similarity** que permite implementar QGram paramétrico.

Usar la biblioteca para implementar:
similitud entre 2 strings y printTokens()

Verificar que lo obtenido coincide con la implementación de Uds.

```
<!-- Soundex -->
<dependency>
   <groupId>commons-codec</groupId>
   <artifactId>commons-codec</artifactId>
   <version>1.5</version>
</dependency>

<!-- Levenshtein -->
<dependency>
   <groupId>org.apache.commons</groupId>
   <artifactId>commons-text</artifactId>
   <version>1.5</version>
</dependency>

<!-- QGrams -->
<dependency>
   <groupId>info.debatty</groupId>
   <artifactId>java-string-similarity</artifactId>
   <version>2.0.0</version>
</dependency>
```

```
import org.apache.commons.codec.language.Metaphone;
import org.apache.commons.codec.language.Soundex;
import org.apache.commons.text.similarity.LevenshteinDetailedDistance;
import org.apache.commons.text.similarity.LevenshteinDistance;
import info.debatty.java.stringsimilarity.QGram;
```

```
Soundex s = new Soundex();
s.difference("HELLO", "ALO");
Metaphone m = new Metaphone();
m.encode("HELLO");
LevenshteinDistance l = new LevenshteinDistance();
 l.apply( "HELLO", "ALO" );

QGram qg = new QGram( 2 );
qg.distance( "Hello", "Alo" );
qg.getProfile( "Hello" );
```