Presentación del Proyecto de Programación I:

MOOGLE!

24 de julio de 2023

Eveliz Espinaco Milián
Primer año de Lic. en Ciencia de la Computación
Facultad de Matemática y Computación, Universidad de La Habana

Curso 2023-2024

1. Introducción

2. Clase Moogle

3. Clase StaticMatrix

4. Clase QueryVectors

5. Clase OperationsVectors

Introducción

MOOGLE! está dividida en dos componentes fundamentales:

- **MoogleServer**, un servidor web que renderiza la interfaz gráfica y sirve los resultados.
- **MoogleEngine**, una *biblioteca de clases* donde está implementada la lógica del algoritmo de búsqueda.

Biblioteca de clases:

- Moogle
- SearchItem
- SearchResult
- StaticMatrix
- QueryVectors
- OperationsVectors

Clase Moogle

Métodos:

- Query
- OpenDataBase
- Snippet

Clase StaticMatrix

StaticMatrix

Esta clase es la encargada de crear la matriz donde estará toda la información disponible para dar respuesta a la búsqueda.

Métodos que la conforman:

- ToDoMatrix.
- Cleaner.
- RefillVocabulary.
- RefillMatrix.
- KeyVocabulary.

Campos de clase:

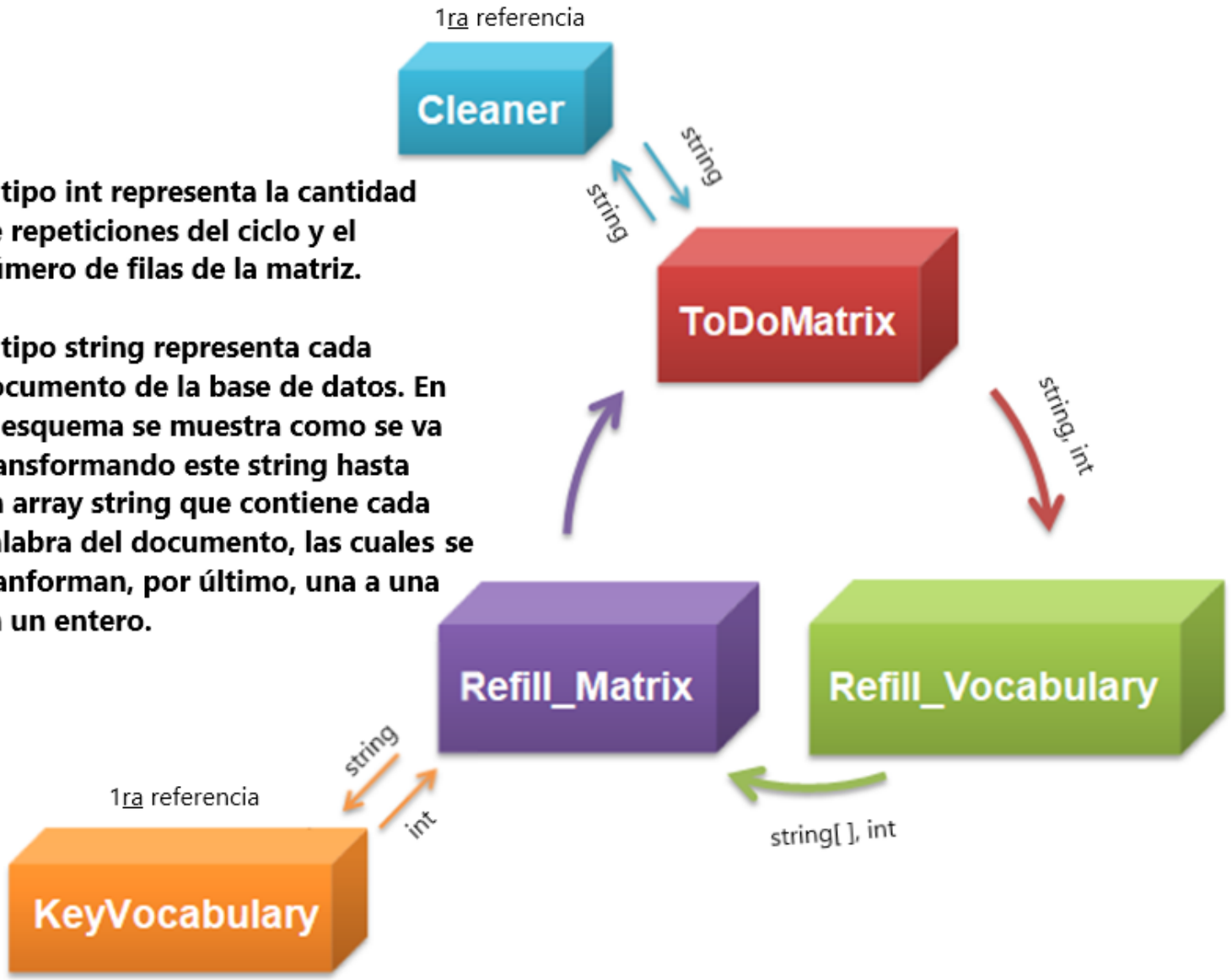
- Key.
- Vocabulary:

StaticMatrix (*¿CÓMO FUNCIONA?*)

Una vez abierto cada documento de la base de datos que se dispone en el array content, este es enviado al método ToDoMatrix de la clase StaticMatrix, este va recorriendo el array e interactuando en cada recorrido con los demás métodos de la clase como se muestra en el esquema:

El tipo int representa la cantidad de repeticiones del ciclo y el número de filas de la matriz.

El tipo string representa cada documento de la base de datos. En el esquema se muestra como se va transformando este string hasta un array string que contiene cada palabra del documento, las cuales se tranforman, por último, una a una en un entero.



Clase QueryVectors

QueryVectors

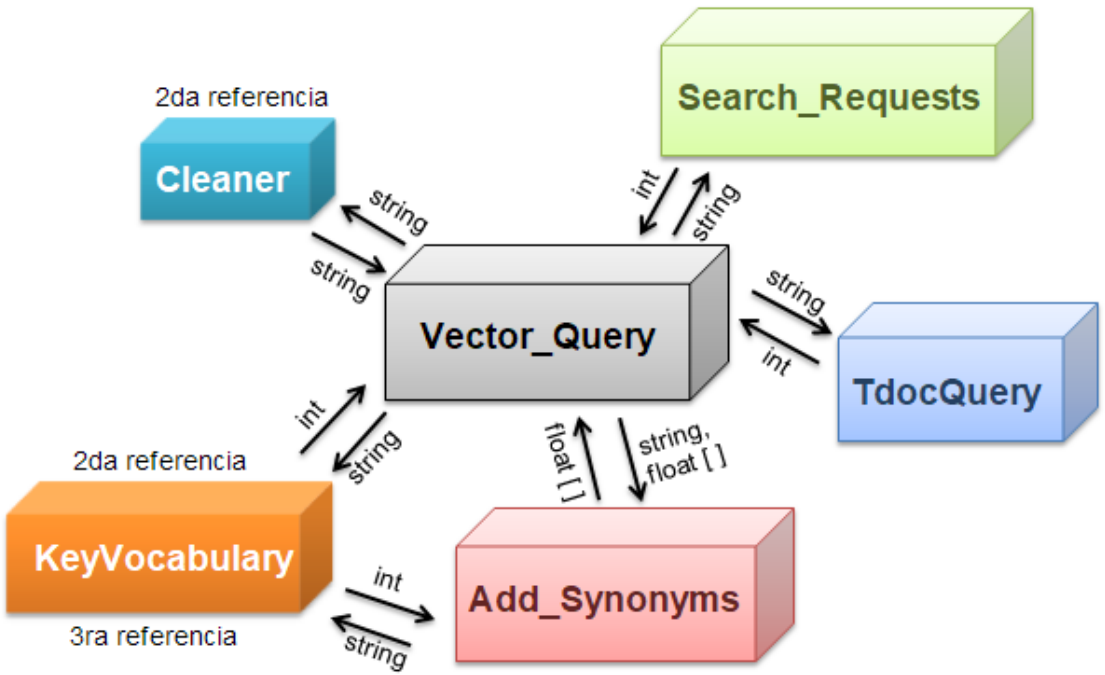
Su funcionalidad es la creación de dos vectores: el primero, un fiel seguidor de cada palabra del usuario y el segundo adiciona sinónimos disponibles de los vocablos introducidos.

Métodos que la conforman:

- QueryVector:
  $(SearchRequests) * log(total\_de\_documentos/(TdocQuery)$
- SearchRequests: analiza si el usuario implementó alguna petición, con la utilización de los caracteres: ! $\wedge$ $*$
- TdocQuery.
- AddSynonyms.

QueryVectors (*¿CÓMO FUNCIONA?)*

Una vez que el usuario introduce una frase de búsqueda esta se envía al método VectorQuery que interactúa con otros métodos como se muestra en el esquema:



Clase OperationsVectors

Métodos que la conforman:

- Multiplication.

La matriz $\mathbf{M_{mxn}}$ contiene la frecuencia de cada palabra en cada documento, esto se obtiene a través de la fórmula **a**/**b**, donde **a** es la cantidad de repeticiones de la palabra en el documento y **b** la cantidad total de palabras que tiene el documento. **(TF)**

La matriz $\mathbf{M_{nx1}}$ está formada en su mayoría por ceros, solo en la posición que le corresponde a la palabra del query valdrá $(SearchRequests) * log(c/d)$, donde **c** es el total de documentos de la base de datos y **d** la cantidad de documentos que contienen la palabra. **(IDF)**

Gracias

Eveliz Espinaco Milián
Primer año de Lic. en Ciencia de la Computación
Facultad de Matemática y Computación, Universidad de La Habana
Curso 2023-2024