Sistemas de Recuperación de Información: Proyecto Final

Nadia González Fernández
José Alejandro Labourdette-Lartigue Soto
C-512

5to Año - Ciencias de la Computación - Curso 2022
Facultad de Matemática y Computación
Universidad de La Habana, La Habana, Cuba

Abstract. *El presente informe describe un sistema de recuperación de información basado en el modelo clásico Vectorial. Se explica el diseño del sistema, las ventajas y desventajas del modelo escogido y las herramientas utilizadas.*

Keywords: Model Vectorial, spaCy, Python, Consultas, Motor de Búsqueda

1 Introducción

La recuperación de información es una disciplina que, con el incremento de la informatización y los volúmenes de datos en la red, cada vez se hace más necesaria la utilización de sistemas de recuperación de información.

Un sistema de información es un conjunto de componentes interrelacionados que permiten capturar, procesar almacenar y distribuir la información para apoyar la toma de decisiones y el control en una organización. Por otra parte, la recuperación de información es la localización de materiales de naturaleza no estructurada para satisfacer una necesidad de información de una larga colección.

En el presente proyecto se realiza un sistema de recuperación de información con la utilización del modelo vectorial. El sistema se especializa en la recuperación de documentos de texto dada su relevancia respecto a una consulta.

El proyecto se encuentra en github: `https://github.com/nala7/information_retrieval_models`

2 Diseño del Sistema

Etapas de la Recuperación de Información:

2.1 Procesamiento de la consulta hecha por un usuario

Las consultas son recibidas como string. Luego son tokenizadas y lematizadas con la biblioteca de Python spaCy.

2.2 Representación de los documentos y la consulta

Los documentos y las consultas son expresados en las clases Document y Query respectivamente. Un documento, como estructura de datos, contiene el nombre del mismo y las palabras que fueron devueltas por el algoritmo de procesamiento de texto. Una query contiene los términos que fueron devueltos por el algoritmo de procesamiento de texto para eliminar stopwords y demás elementos del lenguaje sin carga semántica.

La clase DocumentCollection es la que representa la colección entera de documentos, en ella tiene un diccionario que permiten saber la frecuencia de ocurrencia de los términos en cada documento.

Como parte de la estrategia para ahorrar memoria se asignan a los nombres de los documentos y a los términos valores numéricos únicos. 4 diccionarios se usan para mapear esta representación numérica.

2.3 Funcionamiento del motor de búsqueda

El motor de búsqueda es una clase de Python que al ser instanciada se le específica la colección de documentos sobre la cual se desea trabajar. En el propio constructor se da la instrucción de computar los pesos de los documentos. El proceso de computar los pesos sigue los pasos establecidos por el motor de búsqueda vectorial para calcular pesos. Dichos valores son almacenados en la propia instancia de DocumentCollection.

Para realizar búsquedas existe la función find(), definida en el propio framework que recibe la query sobre la que se desea buscar. La propia función computa los pesos de la query y calcula la similitud con cada documento.

2.4 Obtención de los resultados

Para la obtención de resultados se define un límite de similitud mínima necesaria para considerar un documento relevante. Los nombres de los documentos con similitud mayor que ese mínimo son devueltos, ordenados de mayor a menor similitud

3 Herramientas Utilizadas

Presentación de las herramientas empleadas para la programación y aspectos más importantes del código.

Para el procesamiento de los documentos y las consultas se utilizó spaCy. Esta es una biblioteca de software para el procesamiento de lenguajes naturales desarrollado por Matt Honnibal y programado en lenguaje Python. Es software libre con Licencia MIT su repositorio se encuentra disponible en Github.

Esta es utilizada en la clase **read_content.py**

spaCy usado para procesar texto

```
nlp = spacy.load('en_core_web_sm')

nlp.max_length = 5030000  # or higher
doc = nlp(text)

# Tokenization and lemmatization
lemma_list = []
for token in doc:
        lemma_list.append(token.lemma_)

# Filter the stopword
filtered_sentence: list[Any] = []
for word in lemma_list:
        lexeme = nlp.vocab[word]
        if not lexeme.is_stop:
                filtered_sentence.append(word)

# Remove punctuation
punctuations = "?:!.,;"
for word in filtered_sentence:
        if word in punctuations:
                filtered_sentence.remove(word)
        if word == '\n':
                filtered_sentence.remove(word)
return filtered_sentence
```

4 Ventajas y Desventajas

4.1 Modelo vectorial

En el modelo Vectorial el esquema de ponderación $tf - idf$ de los documentos resulta en un buen rendimiento de la recuperación. La estrategia de coincidencia parcial permite la recuperación de documentos que se aproximen a los requerimientos de la consulta. Además la fórmula del coseno ordena los documentos de acuerdo al grado de similitud con la consulta. Por otra parte, el modelo vectorial tiene como desventaja que asume que los términos indexados son mutuamente independientes, sin embargo, esto hace que su rendimiento sea mejor.

Con respecto al MRI Booleano, el Vectorial tiene como ventaja que permite hacer un ranking de los documentos y da una correspondencia parcial entre documentos y consultas. En comparación con el modelo Probabilístico se ha demostrado que el MRI Vectorial tiene un mejor desempeño.

El modelo Vectorial es simple, rápido y, en algunos casos, brinda mejores resultados en la recuperación de información que el resto de los MRI clásicos.

References

1. Prof. Carlos Fleitasa Aparicio, Profe. Marcel E. Sánchez Aguilar, Departamento de Programación, Facultad MATCOM, Universidad de La Habana (2021)
2. Text normalization with spacy and nltk, https://towardsdatascience.com/text-normalization-with-spacy-and-nltk-1302ff430119
3. Documentación oficial de Spacy https://spacy.io/