Proyecto de Sistemas de Recuperación de Información

Modelo Booleano

Kevin Talavera Diaz C-311

Facultad de Matemática y Computación

Universidad de La Habana

Curso 2021-2022

Link: https://github.com/KevinTD15/SRI-models.git

Resumen. Pre-entrega del proyecto final de Sistemas de Recuperación de Información. Modelo Booleano implementado en Python. Este está compuesto por 3 módulos principales: Procesamiento de Texto, Modelado del SRI y el módulo principal donde se realizan las consultas en consola.

Keywords: path: dirección

Requerimientos de Software:

- python 3.10.7
- numpy 1.23.4
- sympy 1.11.1
- nltk 3.7

- **Introducción**

El modelo booleano constituye el primer modelo teórico empleado para establecer el subconjunto de documentos relevantes, en relación a una consulta específica realizada a una colección de documentos sean estos páginas disponibles en la web o en una biblioteca digital. Está basado en el álgebra de Boole, por lo que se considera como un modelo simple y fácil de implementar, por lo que fue el preferido en los Sistemas de Recuperación tempranos

- **Módulo Principal**

En este módulo es donde se muestran las opciones disponibles para el uso del modelo. Este cuenta con 4 etapas:

--La ejecución se inicia al es escribir: **python main.py** en la consola

1- Ingresar el path: Se tiene que poner una dirección válida de donde quiera realizar consultas en la PC.
2- Modos de Consulta: Se implementaron 2, el **casual** y el **experto**: El casual es para usuarios no versados en el álgebra booleana y que puedan escribir consultas en lenguaje natural.
El experto es aquel que tiene al menos conocimientos basicos de lógica y puede escribir consultas con el formato de expresiones lógicas.
3- Tipo de coincidencia: Se implementaron 2 tipos, la coincidencia parcial y la total. Esto solo afecta a las consultas de usuarios casuales, ya que el algoritmo al recibir una consulta en lenguaje natural es incapaz de detectar signos de agrupación. Ej:
A y B o C puede ser interpretado como A y (B o C) ó
(A y B) o C lo cual sería muy ambiguo.
La propuesta para hacer una consulta mas amigable es que el usuario decida si quiere encontrar coincidencias exactas de los términos (coincidencia total) y esto en programa sería poner operadores **AND** entre los pares de términos, o si desea coincidencias parciales, es decir, cualquier término de la consulta que aparezca en un documento, es parte de los documentos recuperados, esto se lleva a cabo poniendo operadores **OR** entre todos los pares de términos de la consulta.
4- Teclear una consulta deseada en correspondencia con las opciones anteriores elegidas.
5- Se mostrarán los documentos recuperados.

- **Procesamiento de texto**

Para poder implementar mecanismos de recuperación sobre una colección de documentos de textos es necesario obtener una representación de los mismos. Con el objetivo de lograr dicha representación se utilizó una bibliotecla de **python** llamada **nltk** ya que esta nos modifica los textos de forma tal que solo queden palabras que aporten significado, por ejemplo: sustantivos, adjetivos, entre otras.

La forma de trabajo con esta biblioteca, que se ve reflejado en el módulo **tokenizer.py** del proyecto fue la siguiente:

1- Se recibe el conjunto de documentos.
2- Se itera por cada uno de estos.
3- Sea d_i el documento correspondiente a la iteración i-ésima de estos:
    3.1- Todos los términos de d_i son llevados a minúscula.
    3.2- Haciendo uso de la función **stopwords** de **nltk** son eliminados los artículos, preposiciones y otros terminos no deseados.
4- Al finalizar con todos los documentos, estas transformaciones son devueltas al módulo **model.py.**

- **Modelación**

De manera general, en este punto se recibe tanto el conjunto de documentos como la consulta.

1- El conjunto de documentos es procesado mediante el módulo **tokenizer.py** previamente explicado.
2- Se crea una matriz en la cual las filas son los documentos y las columnas los términos.
3- La consulta puede procesarse de 2 formas, primeramente si es escrita en lenguaje natural se hace de igual forma que los documentos, la otra forma es que si ya es escrita en forma de álgebra booleana nos saltamos ese paso de procesamiento ya que no es necesario.
4- Haciendo uso de la biblioteca **sympy** cada término de la consulta se vuelve un símbolo, el cual esta biblioteca reconoce como una variable con la que se pueden hacer operaciones booleanas.
5- Mediante la función **to_dnf** del propio sympy la consulta ya reconocida como epresión booleana es llevada a **FND** (Forma Normal Disyuntiva).
6- Cada una de las componentes conjuntivas de esta es buscada en la matriz de ocurrencia previamente mencionada y los que se correspondan son documentos a ser devueltos.

Referencias:

1. Expresiones booleanas con sympy:

   https://omz-software.com/pythonista/sympy/modules/logic.html

2. Listar archivos de un directorio:

   https://j2logo.com/python/listar-directorio-en-python/#:~:text=Para%20listar%20o%20recorrer%20un,archivos%20y%20carpetas%20que%20contiene.

3. Uso de numpy:

   https://numpy.org/doc/stable/user/index.html#user

4. Procesamiento de texto con nltk:
   https://www.nltk.org/