Laboratorio 6.2

Rodriguez, Mauricio
Universidad de Tecnología e Ingeniería
Ciencia de la Computación
Lima, Perú
mauricio.rodriguez@utec.edu.pe

***Abstract*—Este proyecto está desarrollado para poner a prueba lo aprendido acerca de *information retrieval* mediante el uso de indices invertidos. Así como de las operaciones que se pueden realizar con estos**

I. Introducción

Con el objetivo de poner a prueba el indice invertido se nos fue dado el *dataset*, el cual consta de 6 archivos de texto. Que, para efectos prácticos, se considera que equivalen a un libro. Estos archivos se utilizaron para generar el indice invertido que pueda recuperar información para el usuario de acuerdo a Querys

II. Explicación

Para la implementación del índice invertido se realizaron 2 pasos, En primer lugar el preprocesamiento. En este obtenemos las siguientes funciones.

- Preprocesamiento
  - *load_books()*: Se encarga de recolectar los datos del *dataset* para almacenarlos en listas.
  - *get_stop_list()*: Se encarga recolectar los datos de la *stoplist* del *dataset* para tener una lista final de palabras o signos de puntuación sin relevancia para el resultado.
  - *tokenize()*: Se encarga de generar una lista única de tokens donde cada token tiene la forma <token, docID>

Dentro de la clase index tambien se realizaron varias funciones

- Index
  - *sort(sorted_tokens)*: Dados los tokens obtenidos en el preprocesamiento, la funcion se encarga de hacer un primer ordenamiento alfabetico por el *token* y un segundo ordenamiento numerico por el *docID*
  - *build_index()*: Genera el indice invertido mediante el uso de un diccionario de forma *token ->docId's*
  - *L(term)*: Con el índice ya construido, se encarga de devolver los documentos relacionados a un termino
  - *AND(array1 , array2)*: Se encarga de devolver la intersección entre 2 *arrays* de documentos
  - *OR(array1 , array2)*: Se encarga de devolver la unión entre 2 *arrays* de documentos
  - *AND_NOT(array1 , array2)*: Se encarga de devolver la diferencia entre 2 *arrays* de documentos

III. Experimentación

Para comprobar el funcionamiento del índice se probaron 3 consultas con términos distintos para probar.

- retrieve(L("comienzo"))
- retrieve(AND(L("comarca"),L("Frodo"))
- retrieve(AND_NOT(L("Gandalf"),L("Tirith"))

De forma que de acuerdo a nuestra implementación se obtuvo el siguiente código y los siguientes resultados.

- *Consultas*

```
if __name__ == "__main__":
    tokens = preprocesamiento.tokenize("libro")
    inverted_index = index()
    inverted_index.build_index(inverted_index.sort(tokens))
    #prueba 1
    print(inverted_index.L("comienzo"))
    #prueba 2
    print(inverted_index.AND(inverted_index.L("Comarca"),inverted_index.L("Frodo")))
    #prueba 3
    print(inverted_index.AND_NOT(inverted_index.L("Gandalf"),inverted_index.L("Tirith")))
```

Fig. 1. consultas

- *Resultados de las consultas*

```
-> BD-II-6.2 git:(main) X python3 main.py
[1, 4, 5, 6]
[1, 6]
[1]
```

Fig. 2. Resultados de las consultas

Conclusiones

- Se aplicaron los conocimientos de modelos de *information retrieval* aprendidos en clase de forma satisfactoria.
- En los experimentos pudimos observar un comportamiento adecuado del indice invertido como de sus operaciones