# Text Document Retrieval
# Indice Invertido
# CS2702

June 7, 2021


| | |
|---|---|
| Date Performed: | Junio 6, 2021 |
| Integrantes: | Fabrizio Vásquez |
| | Jean Miraval |
| Profesor: | Heider Sanchez |


## 1 Objetivos

- Entender el funcionamiento de un índice invertido.
- Implementar índice invertido en python.
- Realizar pruebas mediante consultas acerca de la colección de textos entregados
- Evaluar complejidad del índice invertido en las consultas elaboradas.

### 1.1 Definitions

**Indice Invertido** Un índice invertido es una forma de estructurar la información que va a ser recuperada posteriormente.

**Stop Words** Es el nombre que reciben las palabras sin significado como artículos, pronombres, preposiciones, etc. que son filtradas antes o después del procesamiento de datos en lenguaje natural.

**Stemming** Stemming es un método para reducir una palabra a su raíz.

**L()** Retorna la lista de publicaciones asociadas al termino

**AND()** Retorna los documentos que contienen a ambos términos de manera conjunta.

**OR()** Retorna los documentos que contienen a al menos uno de los términos.

**AND-NOT()** Retorna los documentos que contienen al primer termino pero no al segundo.

## 2 Experimentos

Conjunto de variables que se usaron durante el proceso de experimentación.

word_one = "Frodo"
word_two = "Bilbo"
word_three = "Gandalf"
word_four = "acabado"
word_five = "acompañar"

- El resultado de la operación L(word_one) [1, 2, 3, 4, 5, 6]
- El resultado de la operación L(word_two) [1, 6]
- El resultado de la operación L(word_three) [1, 2, 3, 5, 6]
- El resultado de la operación L(word_four) [1, 4, 5, 6]
- El resultado de la operación L(word_five) [1, 2, 5, 6]
- El resultado de la operación AND(word_one,word_three) [1, 2, 3, 5, 6]
- El resultado de la operación OR(word_four,word_five) [1, 2, 4, 5, 6]
- El resultado de la operacion AND_NOT(word_one,word_two) [2, 3, 4, 5]
- El resultado de la operacion RECOVERY(AND(AND(L(word_one),L(word_two)),L(word_three)) [1, 6]

## 3 Procedimiento

Para la elaboración del índice invertido se empleo librerías de python como ntlk para el procesamiento y tokenización de palabras de los archivos y se uso como recurso adicional un archivo que incluía stop word.
Esta implementación de índice invertido está basado en un diccionario que guarda los términos (key) y los asocia a una lista sin repetición de la ubicación de los términos en base a archivos previamente procesados (values).
Como primer paso durante la implementación se cargaron a memoria RAM todos los stop words. Luego se itero sobre cada archivo (documento o libro) para

tokenizar las palabaras en su interior. Posterior a ello, se realizo una limpieza de puntos, comas, corchetes o algún otro símbolo extraño que realmente no tenga un propósito dentro del mapeo de palabras. Seguidamente, se realizo el proceso de stemming, para que por último se construya un diccionario con la lista asociadas a cada termino en las páginas que se encuentre. Finalmente, debemos recordar que deseamos obtener las 500 palabras más frecuentes asociadas a su lista de ocurrencia en archivos, por lo que se ordena de forma inversa el diccionario (descendente) y en un proceso de eliminación se descarta el resto que no se encuentre en dichos 500 primeros más frecuentes. Obteniendo al final las 500 palabras más frecuentes asociadas a su lista de ocurrencia sin repetición y ordenadas de forma ascedente de los archivos procesados.
Para cálculos que se hicieron para la etapa de experimentación únicamente se uso la función L() que retorna la lista con los números de páginas asociadas al término. Debido a que, buscar las páginas asociadas a cada término en todos los documentos de forma visual, hubiera costado un tiempo razonablemente elevado, es por ello que se opto por esta manera.

Para esta etapa se usaron conceptos básicos de teoría de conjuntos como:

$$A \cup B$$

$$A \cap B$$

$$A \notin B$$

# 4 Resultados

En base a los resultados obtenidos, las funciones evaluadas en la parte de experimentación usando teoría de conjuntos de manera empírica son los mismos que los resultados evaluados por nuestro índice invertido luego de operar las diferentes operaciones que se indican en la imagen inferior.

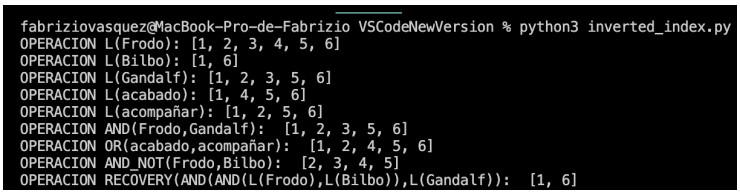


Figure 1: resultado de funciones

**Operacion L**
El costo de la operacion L en base a funcion de la clase InvertedIndex es de O(1) siendo, ya que simplemente búscamos en un diccionario que en otras palabras funcionaría como hash y el acceso en un hash es O(1).

**Operacion AND**
El costo de la operacion AND en base a funcion de la clase InvertedIndex es de O(min(n,m)) siendo n y m las longitudes de las listas con las ubicaciones de los términos.
**Operacion OR**
El costo de la operacion OR ordenado en base a funcion de la clase InvertedIndex es de O(n+m) siendo n y m las longitudes de las listas con las ubicaciones de los términos, ya que es el suma de la longitud de las dos listas porque los punteros recorren ambas listas en el peor caso para la union.
**Operacion AND_NOT**
El costo de la operacion AND_NOT en base a funcion de la clase InvertedIndex es de O(n+m).

# 5 Conclusiones

a. Se concluye que los experimentos realizados de manera empírica están validados durante el proceso de ejecución del programa con los test de las funciones presentadas.

b. Se podría realizar una mejor implementación en el sentido de poder mergear(mezclar) por archivo el conjunto de términos en uno solo. Sin embargo, habría que tomar en cuenta el tema de colisiones al momento de mezclar los diferentes indices invertidos por cada documento, ya que si nos colocamos en el caso de archivos muy pesados entonces la RAM simplemente no nos podría brindar el soporte para la construcción de un gran índice invertido como un todo.