



Universidad de Ingeniería y Tecnología

Base de Datos 2  
Informe

**LAB06 Inverted Index**

**Alumnos:**

Claudio Echarre

Christian Salazar

**Profesor:**

Heider Sanchez

Barranco, octubre de 2021

## 1. Introducción

El presente informe busca desarrollar y analizar las funciones de un índice invertido para la implementación en gestores de bases de datos. Esta implementación es utilizada para la recuperación de la información. Procederemos a explicar la implementación de nuestro índice invertido, mostrando con imágenes partes claves del código en Python. En este caso analizaremos 6 libros de “El Señor de los Anillos” y aplicaremos la función de recuperación booleana y los operadores AND, OR, NOT.

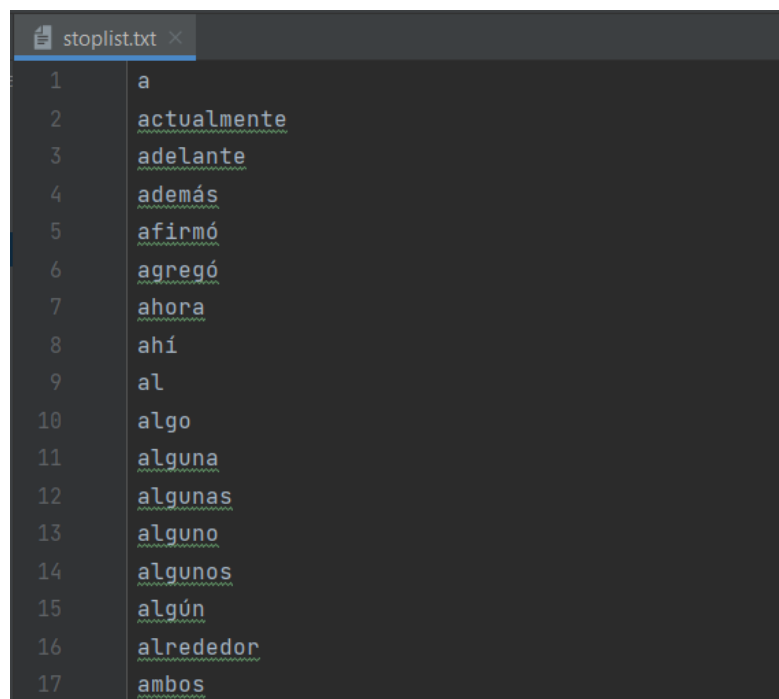
## 2. Herramientas

El código del programa estará escrito en lenguaje Python.

Utilizaremos la librería nltk que sirve para el procesamiento de lenguaje y reducción de palabras a su raíz.

```
1 import nltk
2 from nltk.stem.snowball import SnowballStemmer
3 nltk.download('punkt')
4
```

Usaremos un stoplist estándar para filtrar cada uno de los stopwords en cada uno de los libros.



```
stoplist.txt
1 a
2 actualmente
3 adelante
4 además
5 afirmó
6 agregó
7 ahora
8 ahí
9 al
10 algo
11 alguna
12 algunas
13 alguno
14 algunos
15 algún
16 alrededor
17 ambos
```

### 3. Implementación

Lo primero que haremos es definir las estructuras que vamos a utilizar. Definimos el número de roots o raíces de palabras y la cantidad de archivos txt. También definimos el Stemmer que nos proporciona la librería para poder dividir las palabras en sus raíces. Generamos manualmente la lista de libros con sus nombres cada uno respectivamente. Preparamos el almacenamiento de los tokens, stoplist y la lista que almacenará los tokens por archivo.

```
NUM_ROOTS = 500
CANT_ARCHIVOS = 7
derivador = SnowballStemmer('spanish')
libros = ['libro1.txt', 'libro2.txt', 'libro3.txt', 'libro4.txt', 'libro5.txt', 'libro6.txt']
tokens = {}
stoplist = []
tokensArchivo = []
```

Creamos la función init para leer libro por libro y extraer las palabras que buscamos. Además de leer cada uno de las líneas del archivo stoplist.txt, agregando adicionalmente algunos caracteres manualmente.

```
def init():
    for archivo in libros:
        with open('Libros/' + archivo, 'r') as file:
            tokensArchivo.append(nltk.word_tokenize(file.read().lower()))
    global stoplist
    with open('stoplist.txt', 'r') as file:
        fileRead = file.read()
        stoplist = fileRead.split('\n')
    stoplist += [',', '.', ';', ':', '(', ')', '«', '»']
```

#### 4. Funciones índice invertido

**L:** retorna la lista de publicaciones asociadas al término.

```
def L(token):  
    raiz = derivador.stem(token)  
    #print("ROOT", root)  
    lista = tokens[raiz]  
    return lista
```

Output:

```
print("A:", L('trop'))  
  
main x  
C:\Users\usuario\AppData\Local  
[nltk_data] Downloading package  
[nltk_data] C:\Users\usua  
[nltk_data] Package punkt is  
A: [4, 5, 6]
```

**AND:** retorna los documentos que contienen a ambos términos de manera conjunta.

```
def AND(token1, token2):  
    archivoA = L(token1)  
    archivoB = L(token2)  
    countA = countB = 0  
    consulta = []  
    while countA < len(archivoA) and countB < len(archivoB):  
        if archivoA[countA] == archivoB[countB]:  
            consulta.append(archivoA[countA])  
            countA += 1  
            countB += 1  
        elif archivoA[countA] > archivoB[countB]:  
            countB += 1  
        else:  
            countA += 1  
    return consulta
```

Output:

```
print(AND('trop', 'ungol'))

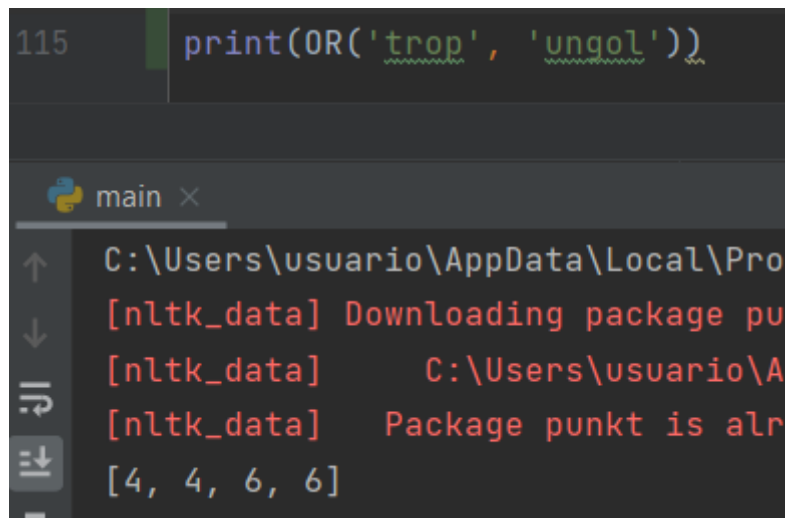
main x
C:\Users\usuario\AppData\Local\Pr
[nltk_data] Downloading package p
[nltk_data] C:\Users\usuario\
[nltk_data] Package punkt is al
[4, 6]
```

**OR:** retorna los documentos que contienen a al menos uno de los términos.

```
def OR(token1, token2):
    archivoA = L(token1)
    archivoB = L(token2)
    countA = countB = 0
    consulta = []
    while countA < len(archivoA) or countB < len(archivoB):
        if countA < len(archivoA) and countB < len(archivoB):
            if archivoA[countA] == archivoB[countB]:
                consulta.append(archivoA[countA])
                countA += 1
                countB += 1
            elif archivoA[countA] < archivoB[countB]:
                consulta.append(archivoA[countA])
                countA += 1
            elif archivoA[countA] > archivoB[countB]:
                consulta.append(archivoB[countB])
                countB += 1
        elif countA < len(archivoA) and countB >= len(archivoB):
            consulta.append(archivoA[countA])
            countA += 1
        elif countA >= len(archivoA) and countB < len(archivoB):
            consulta.append(archivoB[countB])
            countB += 1
    return consulta
```

Output:

```
115 print(OR('trop', 'ungol'))
```



The screenshot shows a Python IDE with a terminal window. The terminal output for the OR function is as follows:

```
C:\Users\usuario\AppData\Local\Pro  
[nltk_data] Downloading package pu  
[nltk_data] C:\Users\usuario\A  
[nltk_data] Package punkt is alr  
[4, 4, 6, 6]
```

**AND-NOT:** retorna los documentos que contienen al primer término pero no al segundo.

```
def ANDNOT(token1, token2):  
    archivoA = L(token1)  
    archivoB = L(token2)  
    countA = countB = 0  
    for token in archivoB:  
        archivoA.remove(token)  
    return archivoA
```

Output:

```
A: [4, 5, 6]  
B: [4, 6]  
[4, 6]  
[5]
```

Por último almacenaremos todos los resultados en un index.txt para poder visualizar y comparar los outputs.

```
def write_file(tokens):  
    f = open("index.txt", 'w')  
    text=""  
    for token in tokens:  
        text = str(token)+":"+str(tokens[token])+'\n'  
    f.write(text)  
    f.close()
```

## 5. Conclusiones

Pudimos crear las funciones que solicitaba el laboratorio, obteniendo las respuestas correctas a cada una de las consultas. La librería nltk es muy útil para este tipo de proyectos en los cuales se trabaja con archivos. Realizar este trabajo en Python es ligeramente más sencillo que hacerlo en otros lenguajes por la cantidad y facilidad de uso de sus librerías.