Inverted Index
Profesor Heider Sanchez

Se le pide desarrollar un programa [preferiblemente en Python] para permitir la recuperación de documentos usando el Índice Invertido.

Probar el correcto funcionamiento del índice usando una colección de resúmenes de 6 libros de "El Señor de los Anillos". Considerar el siguiente proceso:

1. Preprocesamiento
    a. Filtrar los stopwords de cada uno de los textos. Para ello debe usar un stoplist estándar (countwordsfree)

```
import nltk
from nltk.corpus import stopwords
with open('Dataset\stoplist.txt') as file:
    stoplist = [line.lower().strip() for line in file]
stoplist += ['.','¿','?','-','anillo',',',':',';','(',')','«', '»']
```

Se crea una lista con las palabras que usaremos como stopwords

    b. Retirar signos innecesarios.

```
libro_palabras_limpias = libro_palabras[:]
for token in libro_palabras:
    if token in stoplist:
        libro_palabras_limpias.remove(token)
```

Con nuestra stoplist filtramos aquellas palabras que no sean relevantes para la busqueda

    c. Reemplazar cada palabra por su raíz (Stemming).

```
from nltk.stem.snowball import SnowballStemmer
stemmer = SnowballStemmer('spanish')
libro_palabras_raiz = []
for w in libro_palabras_limpias:
    libro_palabras_raiz.append(stemmer.stem(w))
```

Mediante SnowballStemmer transformamos cada palabra en su raíz con la finalidad de normalizar las palabras.

Puede ayudarse usando las librerías de nltk de Python solo para esta etapa.

2. Construcción del índice invertido:
    a. Construir el índice con los 500 términos más frecuentes de toda la colección.

```
libro1_palabras_raiz = palabras_raiz('Dataset\libro1.txt')
libro2_palabras_raiz = palabras_raiz('Dataset\libro2.txt')
libro3_palabras_raiz = palabras_raiz('Dataset\libro3.txt')
libro4_palabras_raiz = palabras_raiz('Dataset\libro4.txt')
libro5_palabras_raiz = palabras_raiz('Dataset\libro5.txt')
libro6_palabras_raiz = palabras_raiz('Dataset\libro6.txt')

from collections import Counter
libro_count_palabras = libro1_palabras_raiz + libro2_palabras_raiz + libro3_palabras_raiz + libro4_palabras_raiz + libro5_palabras_raiz + libro6_palabras_raiz
counter=Counter(libro_count_palabras)
most_common_palabras = sorted([i[0] for i in counter.most_common(500)])

index = {i: [] for i in most_common_palabras}

for i in index.keys():
    if i in libro1_palabras_raiz:
        index[i].append(1)
    if i in libro2_palabras_raiz:
        index[i].append(2)
    if i in libro3_palabras_raiz:
        index[i].append(3)
    if i in libro4_palabras_raiz:
        index[i].append(4)
    if i in libro5_palabras_raiz:
        index[i].append(5)
    if i in libro6_palabras_raiz:
        index[i].append(6)
```

Teniendo una lista con las palabras normalizadas de cada archivos encontramos las 500 más frecuentes con ayuda de la librería collections y hacemos un sorted para ordenar alfabéticamente. Luego a cada palabra enlazamos con los archivos en las cuales estas aparecen.

    b. Guardar el índice en un archivo de texto. Por fines comparativos, ordenar el índice alfabéticamente. Formato:

```
W1:1,3
W2:2,5,6
W3:1,2
W4:2
```

```
with open('invertIndex.txt', 'w') as f:
    for i in index.keys():
        f.write(str(i) + ' : ')
        for j in index[i]:
            f.write(str(j) + ' ')
        f.write('\n')
```

3. Aplicar Consultas Booleanas:
    a. Implemente la función de recuperación booleana y los operadores AND, OR y NOT. Ejemplo:

    Query:

    "(Comunidad AND Frodo) AND-NOT Gondor"

    Ejecución:

    result = retrieve(ANDNOT(AND(L("*Calpurnia*"), L("*Brutus*")), L("Cesar"))

    En donde:

    - L() retorna la lista de publicaciones asountitled:Untitled-1ciadas 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.

```
1.  from nltk.stem.snowball import SnowballStemmer
2.  class InvertIndex:
3.      def __init__(self):
4.              self.index = index
5.
```

```
6.      def L(self, key):
7.          stemmer = SnowballStemmer('spanish')
8.          key = stemmer.stem(key.lower())
9.          if key not in self.index:
10.             return []
11.         else:
12.             return self.index[key]
13.
14.     def AND(self, L1, L2):
15.         return self.intersect(L1, L2)
16.
17.     def OR(self, L1, L2):
18.         return self.union(L1, L2)
19.
20.     def ANDNOT(self, L1, L2):
21.         return self.difference(L1, L2)
22.
23.     def intersect(self, L1, L2):
24.         result = []
25.         p1 = 0
26.         p2 = 0
27.         while p1<len(L1) and p2<len(L2):
28.             if L1[p1] == L2[p2]:
29.                 result.append(L1[p1])
30.                 p1+=1
31.                 p2+=1
32.             elif L1[p1] < L2[p2]:
33.                 p1+=1
34.             else:
35.                 p2+=1
36.         return result
37.
38.     def union(self, L1, L2):
39.         result = []
40.         for i in L1:
41.             if i not in result:
42.                 result.append(i)
43.         for i in L2:
44.             if i not in result:
45.                 result.append(i)
46.         return result
47.
48.     def difference(self, L1, L2):
49.         result = []
50.         for i in L1:
51.             if i not in result:
52.                 result.append(i)
53.         for i in L2:
54.             if i in result:
55.                 result.remove(i)
56.         return result
```

b. Probar el programa con al menos 3 consultas y al menos 3 términos.

```
retrieve = InvertIndex()
consulta1 = retrieve.OR(retrieve.AND(retrieve.L("Frodo"), retrieve.L("Gandalf")), retrieve.L("Gondor"))
consulta2 = retrieve.ANDNOT(retrieve.OR(retrieve.L("Frodo"), retrieve.L("Gandalf")), retrieve.L("Gondor"))
consulta3 = retrieve.AND(retrieve.OR(retrieve.L("Frodo"), retrieve.L("Gandalf")), retrieve.L("Gondor"))
print("((Frodor AND Gandalf) OR Gondor) aparecen en los libros: ", consulta1)
print("((Frodor OR Gandalf) ANDNOT Gondor) aparecen en los libros: ", consulta2)
print("((Frodor OR Gandalf) AND Gondor) aparecen en los libros: ", consulta3)
```

✓ 0.4s

```
((Frodor AND Gandalf) OR Gondor) aparecen en los libros:  [1, 2, 3, 5, 6]
((Frodor OR Gandalf) ANDNOT Gondor) aparecen en los libros:  [1, 4]
((Frodor OR Gandalf) AND Gondor) aparecen en los libros:  [2, 3, 5, 6]
```

c. (opcional) Elabore un parser que transforme la consulta textual booleana en instrucciones de ejecución.

Entregable: elabore un informe detallando la implementación del índice invertido y los resultados obtenidos.