1 Preprocesamiento

Primero, se almacenaron las palabras de cada libro para posteriormente limpiar los datos de cada libro eliminando signos innecesarios y palabras de la stoplist.

Almacenamiento de las palabras

```
    palabras = []
    books = []
    for i in range (1,7):
    name = "libro"
    name += str(i) + ".txt"
    with open(name) as libro:
        temp = nltk.word_tokenize(libro.read().lower())
        books.append(temp)
    palabras += temp
```

Limpieza de datos

```
with open('stoplist.txt') as file:
    stoplist = [line.lower().strip() for line in file]
stoplist += ['.','?','-','¿',',',';','  ','  ','  ','  ','¡','!',':','(',')','\'\'',]

clean_books = books[:]

for book in clean_books:
    for token in book:
        if token in stoplist:
            book.remove(token)
```

Finalmente, se reemplazaron las palabras de los textos por los índices obtenidos gracias a la función SnowballStemmer

```
from nltk.stem.snowball import SnowballStemmer
stemmer = SnowballStemmer('spanish')

books_index = []
for book in clean_books:
    temp = []
    for w in book:
        temp.append(stemmer.stem(w))
    books_index.append(temp)
```

2 Construir el índice invertido

Para la construcción del índice invertido se usó un diccionario de datos en el que cada palabra almacena una lista de los libros en las que estas aparecen.

Primero, se hizo el conteo de la frecuencia en la que las palabras aparecen en los libros para trabajar con los 500 índices que más se repiten. Luego, se ordenó alfabéticamente la lista de índices.

```
import operator
frec = {}

allwords = []

for book in books_index:
    for w in book:
        allwords.append(w)

for w in allwords:
   frec[w] = allwords.count(w)

frec = sorted(frec.items(), key=operator.itemgetter(1), reverse=True)

most_frec = []
term = 0
for w in enumerate(frec):
    term += 1
    if(term == 501):
        break
    most_frec.append(w[1][0])

most_frec.sort()
```

Finalmente, para cada palabra del índice se le agregó el número del libro en el cual dicha palabra aparece y se guardó el índice invertido en un archivo "index.txt".

```
ind = {}

for w in most_frec:
    ind[w] = []

#print(ind)
n_book = 1
for book in books_index:
    for w in most_frec:
        if w in book:
            ind[w].append(n_book)
    n_book+=1

indexFile = open('index.txt','w')
for i in ind.items():
    indexFile.write(str(i))
    indexFile.write('\n')
indexFile.close()
```

3 Aplicar consultas booleanas

3.1 Función L

Esta función se encarga de retornar los libros en la que una palabra aparece.

```
def L(token):
    if ind.get(stemmer.stem(token).lower()) == None:
        return []
    else:
        return ind.get(stemmer.stem(token).lower())
```

3.2 Función AND

Esta función se encarga de retornar la intersección de los libros de dos palabras.

```
def AND(books_1, books_2):
    result = []
    for i in books_1:
        for j in books_2:
            if i == j:
                result.append(i)
    return result

#Ejemplo
print(AND(L("Frodo"), L("abismo")))
```

3.3 Función OR

Esta función se encarga de retornar la unión de los libros de dos palabras.

```
def OR(books_1, books_2):
    result = books_1 + books_2
    result = list(set(result))
    sorted(result)
    return result
```

3.4 Función AND_NOT

Esta función se encarga de eliminar los libros en los una palabra aparece de otra lista de libros.

```
def AND_NOT(books_1, books_2):
    for i in books_2:
        if i in books_1:
```

```
            books_1.remove(i)
    sorted(books_1)
    return books_1
```

3.5 Ejecución de consulta

También se pueden juntar las funciones para poder realizar consultas

```
result = AND_NOT(AND(L("Frodo"), L("Abismo")), L("acompa ar"))

print(result)

[3]
```

Repositorio Github

Enlace al repositorio del programa: https://github.com/RenzoT/lab6.2-bd2