```
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer, PorterStemmer
from nltk.tokenize import sent_tokenize , word_tokenize
import glob
import re
import os
import numpy as np
import sys
Stopwords = set(stopwords.words('english'))
def finding_all_unique_words_and_freq(words):
    words_unique = []
    word_freq = {}
    for word in words:
     if word not in words_unique:
      words_unique.append(word)
    for word in words_unique:
     word_freq[word] = words.count(word)
    return word_freq

    #freq = words.count(word)

def remove_special_characters(text):
    regex = re.compile('[^a-zA-Z0-9\s]')
    text_returned = re.sub(regex,'',text)
    return text_returned
class Node:
    def __init__(self ,docId, freq = None):
        self.freq = freq
        self.doc = docId
        self.nextval = None

class SlinkedList:
    def __init__(self ,head = None):
        self.head = head
all_words = []
dict_global = {}
file_folder = 'C:/nltk_data/corpora/shakespeare/*'
idx = 1
files_with_index = {}
for file in glob.glob(file_folder):
    print(file)
    fname = file
   # os.startfile(r'C:/nltk_data/corpora/shakespeare/')
    #file = open(file,"r")
    #text = file.read()
    with open(file, 'r') as f:
```

```
     text = f.read()
    text = remove_special_characters(text)
    text = re.sub(re.compile('\d'),'',text)
    sentences = sent_tokenize(text)
    words = word_tokenize(text)
    words = [word for word in words if len(words)>1]
    words = [word.lower() for word in words]
    words = [word for word in words if word not in Stopwords]
    dict_global.update(finding_all_unique_words_and_freq(words))
    files_with_index[idx] = os.path.basename(fname)
    idx = idx + 1
#print(words)
unique_words_all = set(dict_global.keys())
linked_list_data = {}
for word in unique_words_all:
    linked_list_data[word] = SlinkedList()
    linked_list_data[word].head = Node(1,Node)
    word_freq_in_doc = {}
    idx = 1
for file in glob.glob(file_folder):
    file = open(file, "r")
    text = file.read()
    text = remove_special_characters(text)
    text = re.sub(re.compile('\d'),'',text)
    sentences = sent_tokenize(text)
    words = word_tokenize(text)
    words = [word for word in words if len(words)>1]
    words = [word.lower() for word in words]
    words = [word for word in words if word not in Stopwords]
    word_freq_in_doc = finding_all_unique_words_and_freq(words)
    for word in word_freq_in_doc.keys():
        linked_list = linked_list_data[word].head
        while linked_list.nextval is not None:
            linked_list = linked_list.nextval
        linked_list.nextval = Node(idx ,word_freq_in_doc[word])
    idx = idx + 1
    #print(idx)
query = input('Enter your query:')
query = word_tokenize(query)
connecting_words = []
cnt = 1
different_words = []
for word in query:
    if word.lower() != "and" and word.lower() != "or" and word.lower() !=
"not":
        different_words.append(word.lower())
    else:
```

```
        connecting_words.append(word.lower())
        print(connecting_words)
total_files = len(files_with_index)
zeroes_and_ones = []
zeroes_and_ones_of_all_words = []
for word in (different_words):
    if word.lower() in unique_words_all:
        zeroes_and_ones = [0] * total_files
        linkedlist = linked_list_data[word].head
        print(word)
        while linkedlist.nextval is not None:
            zeroes_and_ones[linkedlist.nextval.doc - 1] = 1
            linkedlist = linkedlist.nextval
        zeroes_and_ones_of_all_words.append(zeroes_and_ones)
    else:
        print(word," not found")
        sys.exit()
        print(zeroes_and_ones_of_all_words)
for word in connecting_words:
    word_list1 = zeroes_and_ones_of_all_words[0]
    word_list2 = zeroes_and_ones_of_all_words[1]
    if word == "and":
        bitwise_op = [w1 & w2 for (w1,w2) in zip(word_list1,word_list2)]
        zeroes_and_ones_of_all_words.remove(word_list1)
        zeroes_and_ones_of_all_words.remove(word_list2)
        zeroes_and_ones_of_all_words.insert(0, bitwise_op);
    elif word == "or":
        bitwise_op = [w1 | w2 for (w1,w2) in zip(word_list1,word_list2)]
        zeroes_and_ones_of_all_words.remove(word_list1)
        zeroes_and_ones_of_all_words.remove(word_list2)
        zeroes_and_ones_of_all_words.insert(0, bitwise_op);
    elif word == "not":
        bitwise_op = [not w1 for w1 in word_list2]
        bitwise_op = [int(b == True) for b in bitwise_op]
        zeroes_and_ones_of_all_words.remove(word_list2)
        zeroes_and_ones_of_all_words.remove(word_list1)
        bitwise_op = [w1 & w2 for (w1,w2) in zip(word_list1,bitwise_op)]
        zeroes_and_ones_of_all_words.insert(0, bitwise_op);

files = []
print(zeroes_and_ones_of_all_words)
lis = zeroes_and_ones_of_all_words[0]
cnt = 1
for index in lis:
    if index == 1:
        files.append(files_with_index[cnt])
    cnt = cnt+1
```

```
print(files)
```

Output:

```
C:\ProgramData\Anaconda3\lib\site-packages\scipy\__init__.py:138:
UserWarning: A NumPy version 1.16.5 and 1.23.0 is required for
this version of Scipy (detected version 1.23.1)warnings.warn(f"A
NumPy version >{np_minversion} and <{np_maxversion} is required for
this version of "
C:/nltk_data/corpora/shakespeare\a_and_c.xml
C:/nltk_data/corpora/shakespeare\dream.xml
C:/nltk_data/corpora/shakespeare\hamlet.xml
C:/nltk_data/corpora/shakespeare\j_caesar.xml
C:/nltk_data/corpora/shakespeare\macbeth.xml
C:/nltk_data/corpora/shakespeare\merchant.xml
C:/nltk_data/corpora/shakespeare othello.xml
C:/nltk_data/corpora/shakespeare\play.dtd
C:/nltk_data/corpora/shakespeare\README
C:/nltk_data/corpora/shakespeare\r_and_j.xml
C:/nltk_data/corpora/shakespeare\shakes.css
Enter your query: othello and cleopatra
['and']
othello
cleopatra
[[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]]
README
```