ARYAMAN MISHRA

19BCE1027

LAB 4

EXERCISE 1

Problem Statement

Collect any 10 documents (English text documents) from the web and create inverted index by doing necessary preprocessing steps using python.

Proposed Algorithm

- **Fetch the Document**
  Removing of Stop Words: Stop words are most occurring and useless words in document like “I”, “the”, “we”, “is”, “an”.
- **Stemming of Root Word**
  Whenever I want to search for “cat”, I want to see a document that has information about it. But the word present in the document is called “cats” or “catty” instead of “cat”. To relate the both words, I’ll chop some part of each and every word I read so that I could get the “root word”. There are standard tools for performing this like “Porter’s Stemmer”.
- **Record Document IDs**
  If word is already present add reference of document to index else create new entry. Add additional information like frequency of word, location of word etc.

Data Structure Proposed: Arrays

Implementation

```
file1 = open('file1.txt', encoding='utf8')
file2 = open('file2.txt', encoding='utf8')
file3 = open('file3.txt', encoding='utf8')
file4 = open('file4.txt', encoding='utf8')
file5 = open('file5.txt', encoding='utf8')
file6 = open('file6.txt', encoding='utf8')
file7 = open('file7.txt', encoding='utf8')
file8 = open('file8.txt', encoding='utf8')
file9 = open('file9.txt', encoding='utf8')
file10 = open('file10.txt', encoding='utf8')
```

FOR EACH FILE:

```
read = file.read()
file.seek(0)
read

line = 1
for word in read:
    if word == '\n':
        line += 1

array = []
for i in range(line):
    array.append(file.readline())

array
punc = '''!()-[]{};:'"\, <>./?@#$%^&*_~'''
for ele in read:
  if ele in punc:
    read = read.replace(ele, " ")

read


read=read.lower()
read
from nltk.tokenize import word_tokenize
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')

for i in range(1):
  text_tokens = word_tokenize(read)

tokens_without_sw = [
  word for word in text_tokens if not word in stopwords.words()]
```

CREATING INVERTED INDEX:

```
dict = {}

for i in range(line):
  check = array[i].lower()
  for item in tokens_without_sw:

    if item in check:
      if item not in dict:
```

```
        dict[item] = []

      if item in dict:
        dict[item].append(i+1)

dict
```

Results

```
'minutes.': {1},
'morning': {1},
'move,': {7},
'new': {3},
'not': {4, 9},
'nowhere.': {7},
'of': {0, 1, 2, 3, 4, 6, 7, 9},
'officers"': {6},
'officials': {3, 4},
'officials.': {3},
'on': {9},
'other': {3},
'out': {0, 5, 6},
'part': {3},
'personnel': {6},
'personnel.': {0},
'plan': {7},
'planned': {4},
'planning': {9},
'police': {6},
'proofs': {2},
'quotes': {3},
'recounts.': {5},
'relations': {7},
'releases': {2},
'resulting': {0},
'same': {8},
'say': {6},
'security': {6},
'senior': {3},
'sentence': {9},
'seven': {6},
'shooting': {2},
'shortly': {7},
'sneak': {6},
'strikes': {5},
'suicide': {0, 1},
'surprise': {7},
'team': {7},
'terror': {0, 1, 7, 9},
'terrorists': {6, 8},
'that': {1, 3, 6, 8, 9},
'the': {0, 1, 3, 4, 5, 6, 7, 8, 9},
'to': {3, 4, 5, 6, 7},
'top': {3},
'trainers': {1},
'training': {1},
'two': {3, 7},
'uniformed': {8},
'unique': {3},
'was': {0, 4, 9},
'went': {7},
'were': {1, 6, 8},
'which': {4, 6},
'who': {9},
'with': {3},
'within': {4},
'year': {8},
'"asset"': {9},
'"corrupt': {6},
'"humiliate': {5},
'"officer"': {9},
'"trapped"': {9}}
```

EXERCISE 2

Problem Statement

Collect any 10 documents (Indian Language text Documents in Unicode) from the web and create inverted index by doing necessary preprocessing steps using python.

Proposed Algorithm

- **Fetch the Document**
  Removing of Stop Words: Stop words are most occurring and useless words in document like “I”, “the”, “we”, “is”, “an”.
- **Stemming of Root Word**
  Whenever I want to search for “cat”, I want to see a document that has information about it. But the word present in the document is called “cats” or “catty” instead of “cat”. To relate the both words, I’ll chop some part of each and every word I read so that I could get the “root word”. There are standard tools for performing this like “Porter’s Stemmer”.
- **Record Document IDs**
  If word is already present add reference of document to index else create new entry. Add additional information like frequency of word, location of word etc.

Data Structure Proposed

Arrays

Implementation

```
file1 = open('file1.txt', encoding='utf8')
file2 = open('file2.txt', encoding='utf8')
file3 = open('file3.txt', encoding='utf8')
file4 = open('file4.txt', encoding='utf8')
file5 = open('file5.txt', encoding='utf8')
file6 = open('file6.txt', encoding='utf8')
file7 = open('file7.txt', encoding='utf8')
file8 = open('file8.txt', encoding='utf8')
file9 = open('file9.txt', encoding='utf8')
file10 = open('file10.txt', encoding='utf8')
```

FOR EACH FILE:

Importos

```
import re
import sys
import time

# list of  stop words
stopwords=['अत', 'अपना', 'अपनी', 'अपने', 'अभी', 'अंदर', 'आदि', 'आप', 'इत्यादि', 'इन
', 'इनका', 'इन्हीं', 'इन्हें', 'इन्हों', 'इस', \
           'इसका', 'इसकी', 'इसके', 'इसमें', 'इसी', 'इसे', 'उन', 'उनका', 'उनकी',
'उनके', 'उनको', 'उन्हीं', 'उन्हें', 'उन्हों', 'उस', \
           'उसके', 'उसी', 'उसे', 'एक', 'एवं', 'एस', 'ऐसे', 'और', 'कई', 'कर',
'करता', 'करते', 'करना', 'करने', 'करें', 'कहते', \
           'कहा', 'का', 'काफ़ी', 'कि', 'कितना', 'किन्हें', 'किन्हों', 'किया', 'किर',
'किस', 'किसी', 'किसे', 'की', 'कुछ', 'कुल', 'के', \
           'को', 'कोई', 'कौन', 'कौनसा', 'गया', 'घर', 'जब', 'जहाँ', 'जा', 'जितना',
'जिन', 'जिन्हें', 'जिन्हों', 'जिस', 'जिसे', 'जीधर', \
           'जैसा', 'जैसे', 'जो', 'तक', 'तब', 'तरह', 'तिन', 'तिन्हें', 'तिन्हों', 'तिस',
'तिसे', 'तो', 'था', 'थी', 'थे', 'दबारा', 'दिया', \
           'दुसरा', 'दूसरे', 'दो', 'द्वारा', 'न', 'नके', 'नहीं', 'ना', 'निहायत', 'नीचे', 'ने',
'पर', 'पहले', 'पूरा', 'पे', 'फिर', 'बनी', \
           'बही', 'बहुत', 'बाद', 'बाला', 'बिलकुल', 'भी', 'भीतर', 'मगर', 'मानो', 'मे',
'में', 'यदि', 'यह', 'यहाँ', 'यही', 'या', 'यिह', \
           'ये', 'रखें', 'रहा', 'रहे', 'ऱ्वासा', 'लिए', 'लिये', 'लेकिन', 'व', 'वग़ैरह', 'वर्ग',
'वह', 'वहाँ', 'वहीं', 'वाले', 'वुह', 'वे', \
           'सकता', 'सकते', 'सबसे', 'सभी', 'साथ', 'साबुत', 'साभ', 'सारा', 'से', 'सो',
'संग', 'ही', 'हुआ', 'हुई', 'हुए', 'है', 'हैं', \
           'हो', 'होता', 'होती', 'होते', 'होना', 'होने', '']


#---- Index Creation-------#

from collections import Counter
from pprint import pprint as pp
from glob import glob
try: reduce
except: from functools import reduce
try:    raw_input
except: raw_input = input


#---- Stemmer for Hindi---------------------#
```

```
suffixes = {
    1: ["ो", "े", "ू", "ु", "ी", "ि", "ा"],
    2: ["कर", "ाओ", "िए", "ाई", "ाए", "ने", "नी", "ना", "ते", \
        "ीं", "ती", "ता", "ाँ", "ां", "ों", "ें"],
    3: ["ाकर", "ाइए", "ाईं", "ाया", "ेगी", "ेगा", "ोगी", "ोगे", "ाने", \
        "ाना", "ाते", "ाती", "ाता", "तीं", "ाओं", "ाएं", "ुओं", "ुएं", "ुआं"],
    4: ["ाएगी", "ाएगा", "ाओगी", "ाओगे", "एंगी", "ेंगी", "एंगे", "ेंगे", "ूंगी",\
        "ूंगा", "ातीं", "नाओं", "नाएं", "ताओं", "ताएं", "ियाँ", "ियों", "ियां"],
    5: ["ाएंगी", "ाएंगे", "ाऊंगी", "ाऊंगा", "ाइयाँ", "ाइयों", "ाइयां"],
}

def hi_stemmer(word):
    for L in 5, 4, 3, 2, 1:
        if len(word) > L + 1:
            for suf in suffixes[L]:
                if word.endswith(suf):
                    return word[:-L]
    return word

def stem_terms(terms_list):
    for term in terms_list:
        term=hi_stemmer(term)
    return terms_list


content={}
def parsetexts(fileglob='C:\\Users\\laisha wadhwa\\Documents\\sem5\\IR\\project\\IR
project_IR_Course\\InputFiles\\*.txt'):
    texts = {}
    words=[]
    for txtfile in glob(fileglob):
        per_file_words=[]
        arr=[]
        f=open(txtfile, encoding='utf-8-sig')
        txt = f.read()
        arr=txt.split("|")
        for i in arr:
            i=i.replace(',','')
            i=i.replace('.','')
            i=i.replace('!','')
            i=i.replace(')','')
            i=i.replace('(','')
            i=i.replace('"','')
```

```
            i=i.replace('\'','')
            per_file_words=per_file_words+ i.strip().strip('"').split()
        per_file_words = list(set(per_file_words))
        per_file_words =list(set(per_file_words)-set(stopwords))
        #time to stem
        per_file_words=stem_terms(per_file_words)
        filename= txtfile.split('\\')[-1]
        texts[filename] = per_file_words
        #content[filename]=txt

        words=words+per_file_words

    return  texts, list(set(words))

def termsearch(terms):
    return reduce(set.intersection,
                  (invindex[term] for term in terms),
                  set(texts.keys()))


#print('\nWords')
#pp(sorted(words))

#print('\nInverted Index')
#pp({k:sorted(v) for k,v in invindex.items()})


def search_inv_idx(phrase):
    #phrase = '"चुपचाप कुएँ में मिट्टी डालते रहे"'
    global texts,words,invindex,intmd_res
    it_str=''
    result=[]
    start = time.clock()
    texts, words = parsetexts()
    invindex = {word:set(txt for txt, wrds in texts.items() if word in wrds) for
word in words}
    query_terms = phrase.strip().strip('"').split()
    query_terms=list(set(query_terms)-set(stopwords))
    query_terms=stem_terms(query_terms)
    intmd_res=list(termsearch(query_terms))
    for i in intmd_res:
        it_str=it_str+ i +'\n'
```

```
        print('\nTerm Search on full inverted index for: ' + repr(query_terms))
        pp(sorted(termsearch(query_terms)))
        result.append("============================================================")
        result.append("\n\n\n-------सामान्य खोज के परिणाम---------------\n\n")
        result.append("परिणाम प्राप्त करने के लिए "+  "  " + str(round(time.clock() -
    start,4))+"s " +"का समय लिया गया")
        result.append(it_str)
        return result
```

Results

आत्मविश्लेषण.txt

आधी रोटी का कर्ज.txt

कर्त्तव्य का पाठ.txt

कर्म की महानता.txt

क्षमा व सद्भावना.txt

गुरु जी की सीख.txt

चिड़िया की परेशानी.txt

नम्रता का पाठ.txt

पेन्सिल की कहानी.txt

बस नज़रिए का फ़र्क है.txt

मन की झील.txt

मैं और मेरा लैपटॉप अक्सर ये बातें करते हैं.txt

शक्ति जीवन है, दुर्बलता ही मृत्यु.txt

हाथी क्यों हारा.txt

हिम्मत मत हारो.txt

प्रेरणादायक

आत्मविश्लेषण.txt

आधी रोटी का कर्ज.txt

कर्त्तव्य का पाठ.txt

कर्म की महानता.txt

क्षमा व सद्भावना.txt

बस नज़रिए का फ़र्क है.txt

हिम्मत मत हारो.txt

नम्रता का पाठ.txt

नैतिक आधारित

गुरु जी की सीख.txt

चिड़िया की परेशानी.txt

मन की झील.txt

हाथी क्यों हारा.txt

शक्ति जीवन है

दुर्बलता ही मृत्यु.txt

लघु कथा

पेन्सिल की कहानी.txt

मैं और मेरा लैपटॉप अक्सर ये बातें करते हैं.txt

उदय प्रकाश 4:

आधी रोटी का कर्ज.txt

कर्म की महानता.txt

गुरु जी की सीख.txt

हिम्मत मत हारो.txt

जयशंकर प्रसाद 4 :

आत्मविश्लेषण.txt

कर्त्तव्य का पाठ.txt

शक्ति जीवन है; दुर्बलता ही मृत्यु.txt

पेन्सिल की कहानी.txt

तारा सिंह 3 :

क्षमा व सद्भावना.txt

मन की झील.txt

हाथी क्यों हारा.txt

धर्मवीर भारती 2:

नम्रता का पाठ.txt

मैं और मेरा लैपटॉप अक्सर ये बातें करते हैं.txt

श्रीलाल शुक्ल 2:

चिड़िया की परेशानी.txt

बस नज़रिए का फ़र्क है.txt

EXERCISE 3

Problem Statement

Collect any 10 documents (Documents in different formats such as PDF, DOC, ODF) from the web and create inverted index by doing necessary preprocessing steps using python.

Proposed Algorithm

- **Fetch the Document**
  Removing of Stop Words: Stop words are most occurring and useless words in document like “I”, “the”, “we”, “is”, “an”.
- **Stemming of Root Word**
  Whenever I want to search for “cat”, I want to see a document that has information about it. But the word present in the document is called “cats” or “catty” instead of “cat”. To relate the both words, I’ll chop some part of each and every word I read so that I could get the “root word”. There are standard tools for performing this like “Porter’s Stemmer”.
- **Record Document IDs**
  If word is already present add reference of document to index else create new entry. Add additional information like frequency of word, location of word etc.

Data Structure Proposed

Arrays

Implementation

```
file1 = open('file1.csv', encoding='utf8')
.read().decode('ascii', 'ignore')

file2 = open('file2.pdf', encoding='utf8')
.read().decode('ascii', 'ignore')



file3 = open('file3.txt', encoding='utf8')
.read().decode('ascii', 'ignore')


file4 = open('file4.sql', encoding='utf8')
.read().decode('ascii', 'ignore')


file5 = open('file5.odf', encoding='utf8')
.read().decode('ascii', 'ignore')
```

```
file6 = open('file6.docx', encoding='utf8')
.read().decode('ascii', 'ignore')


file7 = open('file7.txt', encoding='utf8')
.read().decode('ascii', 'ignore')


file8 = open('file8.psd', encoding='utf8')
.read().decode('ascii', 'ignore')


file9 = open('file9.dxf', encoding='utf8')
.read().decode('ascii', 'ignore')


file10 = open('file10.pdf', encoding='utf8')
.read().decode('ascii', 'ignore')
```

FOR EACH FILE:

```
read = file.read()
file.seek(0)
read

line = 1
for word in read:
    if word == '\n':
        line += 1

array = []
for i in range(line):
    array.append(file.readline())

array
punc = '''!()-[]{};:'"\, <>./?@#$%^&*_~'''
for ele in read:
  if ele in punc:
    read = read.replace(ele, " ")

read


read=read.lower()
```

```
read
from nltk.tokenize import word_tokenize
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')

for i in range(1):
  text_tokens = word_tokenize(read)

tokens_without_sw = [
  word for word in text_tokens if not word in stopwords.words()]
```

CREATING INVERTED INDEX:

```
dict = {}

for i in range(line):
  check = array[i].lower()
  for item in tokens_without_sw:

    if item in check:
      if item not in dict:
        dict[item] = []

      if item in dict:
        dict[item].append(i+1)

dict
```

Results

```
'minutes.': {1},
'morning': {1},
'move,': {7},
'new': {3},
'not': {4, 9},
'nowhere.': {7},
'of': {0, 1, 2, 3, 4, 6, 7, 9},
'officers"': {6},
'officials': {3, 4},
'officials.': {3},
'on': {9},
'other': {3},
'out': {0, 5, 6},
'part': {3},
'personnel': {6},
'personnel.': {0},
'plan': {7},
'planned': {4},
'planning': {9},
'police': {6},
'proofs': {2},
'quotes': {3},
'recounts.': {5},
'relations': {7},
'releases': {2},
'resulting': {0},
'same': {8},
'say': {6},
'security': {6},
'senior': {3},
'sentence': {9},
'seven': {6},
'shooting': {2},
'shortly': {7},
'sneak': {6},
'strikes': {5},
'suicide': {0, 1},
'surprise': {7},
'team': {7},
'terror': {0, 1, 7, 9},
'terrorists': {6, 8},
'that': {1, 3, 6, 8, 9},
'the': {0, 1, 3, 4, 5, 6, 7, 8, 9},
'to': {3, 4, 5, 6, 7},
'top': {3},
'trainers': {1},
'training': {1},
'two': {3, 7},
'uniformed': {8},
'unique': {3},
'was': {0, 4, 9},
'went': {7},
'were': {1, 6, 8},
'which': {4, 6},
'who': {9},
'with': {3},
'within': {4},
'year': {8},
'"asset"': {9},
'"corrupt': {6},
'"humiliate': {5},
'"officer"': {9},
'"trapped"': {9}}
```

Conclusion:

All 3 exercises have been succesfully executed and full output has been added with font size 1.