WEB MINING
LAB

Faculty: Dr.Sridhar.R

LAB5
DATE: 2ND SEP 2021

VIT CHENNAI

Aim: To make an inverted indexer

Code:

```
inverted_index={}

def file_to_list(*files):
    files_list=[]
    for file in files:
        files_list.append(open(file,"r").read());
    return files_list

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

def word_dictionary(files_list):
    for i,file in enumerate(files_list):
        for word in file.split():
            clean_word = remove_punctuation(word)
            if clean_word in inverted_index:
                inverted_index[clean_word].append(i+1)
            else:
                inverted_index[clean_word]=[i+1]

def add_freq_to_dict():
    for word,posting_list in inverted_index.items():
        freq = len(posting_list)
        posting_list = list(dict.fromkeys(posting_list))
        inverted_index[word] = (freq,posting_list)

def print_dict_as_table():
    print("WORD ","    FREQ ","     POSTING_LIST ")
    for key,value in inverted_index.items():
        freq,posting_list = value
        print ("{:<10} {:<10}".format(key, freq),end=" ")
        print(posting_list)
```

```
def find_word_in_doc(word):
    if word in inverted_index:
        res = inverted_index[word][1]
        print(word," is found in docs with ids:",res)
    else:
        print("No such word is found")



files_list=file_to_list("file1.txt","file2.txt")
word_dictionary(files_list)
add_freq_to_dict()
find_word_in_doc("Caeser")
print_dict_as_table()
```

OutPut:

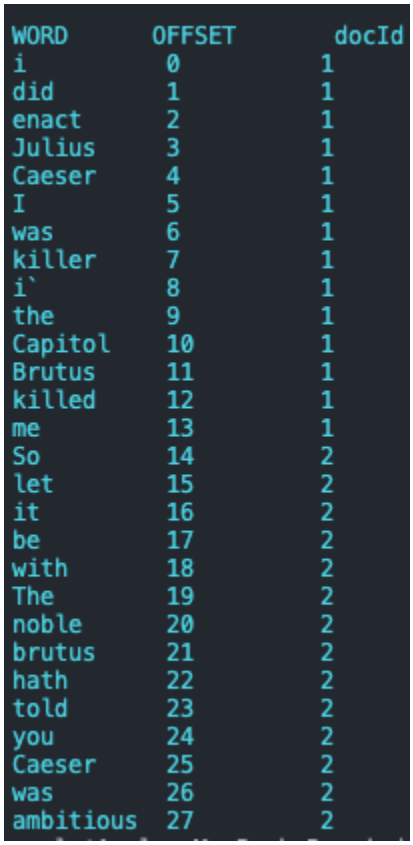
dictionary

```
WORD       FREQ       POSTING_LIST
i          1          [1]
did        1          [1]
enact      1          [1]
Julius     1          [1]
Caeser     2          [1, 2]
I          1          [1]
was        2          [1, 2]
killer     1          [1]
i`         1          [1]
the        1          [1]
Capitol    1          [1]
Brutus     1          [1]
killed     1          [1]
me         1          [1]
So         1          [2]
let        1          [2]
it         1          [2]
be         1          [2]
with       1          [2]
The        1          [2]
noble      1          [2]
brutus     1          [2]
hath       1          [2]
told       1          [2]
you        1          [2]
ambitious  1          [2]
```

Searching for a word:

```
  Caeser  is found in docs with ids: [1, 2]
```

Posting List:



WORD	OFFSET	docId
i	0	1
did	1	1
enact	2	1
Julius	3	1
Caeser	4	1
I	5	1
was	6	1
killer	7	1
i`	8	1
the	9	1
Capitol	10	1
Brutus	11	1
killed	12	1
me	13	1
So	14	2
let	15	2
it	16	2
be	17	2
with	18	2
The	19	2
noble	20	2
brutus	21	2
hath	22	2
told	23	2
you	24	2
Caeser	25	2
was	26	2
ambitious	27	2