Experiment 4

**Inverted index challenging exercise**

Web scrapping documents and create Inverted Index after nessesary preprocessing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name:** | **:** | **Abhishek N N** | **Register No** | **:** | **20BCE1025** |
| Faculty | : | Dr. Alok Chauhan | Slot | : | L51-L52 AB1-605B |
| Course | : | Web Mining Lab | Code | : | CSE3024 |
| Programme | : | B.Tech CSE Core | Semester | : | Win – 22 - 23 |



**Problem- 1:**

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

# preprocessing

# import the necessary libraries

import string

import nltk

import re

def text\_lowercase(text):

return text.lower()

# Remove numbers

def remove\_numbers(text):

result = re.sub(r'\d+', '', text)

return result

# remove punctuation

def remove\_punctuation(text):

translator = str.maketrans('', '', string.punctuation)

return text.translate(translator)

# remove whitespace from text

def remove\_whitespace(text):

return " ".join(text.split())

# remove stopwords

def remove\_stopwords(text):

stopword\_list = nltk.corpus.stopwords.words('english')

tokens = nltk.word\_tokenize(text)

tokens = [token.strip() for token in tokens]

return ' '.join([token for token in tokens if token not in stopword\_list])

# stemming

def stem\_text(text):

ps = nltk.PorterStemmer()

tokens = nltk.word\_tokenize(text)

tokens = [token.strip() for token in tokens]

return ' '.join([ps.stem(token) for token in tokens])

# lemmatization

def lemmatize\_text(text):

wnl = nltk.WordNetLemmatizer()

tokens = nltk.word\_tokenize(text)

tokens = [token.strip() for token in tokens]

return ' '.join([wnl.lemmatize(token) for token in tokens])

# remove special characters

def remove\_special\_characters(text):

pattern=r'[^a-zA-z0-9\s]'

text=re.sub(pattern,'',text)

return text

# remove extra newlines

def remove\_extra\_newlines(text):

pattern=r'[\r|\n|\r]+'

text=re.sub(pattern,' ',text)

return text

# apply all the functions to the text

def preprocess(corpus):

normalized\_corpus = []

# normalize each document in the corpus

for doc in corpus:

doc = text\_lowercase(doc)

doc = remove\_numbers(doc)

doc = remove\_punctuation(doc)

doc = remove\_whitespace(doc)

doc = remove\_special\_characters(doc)

doc = remove\_extra\_newlines(doc)

doc = lemmatize\_text(doc)

doc = stem\_text(doc)

doc = remove\_stopwords(doc)

normalized\_corpus.append(doc)

return normalized\_corpus

# inverted index

def generateInvertedIndexDict(dataFromDoc: list[str]) :

d=dict()

termsListFromDoc = [s.split() for s in dataFromDoc]

for docId, termList in enumerate(termsListFromDoc):

for term in termList:

if term not in d:

d[term]={docId}

else:

d[term].add(docId)

return d

# file handling

from os import listdir

from os.path import isfile, join

def getDataFromDocs(dir):

"""

gets strings from docs

parameters:

dir (str) : directroy which contains all files

return:

list of str read from docs in the directory given by user

"""

return [open(join(dir, f)).read() for f in sorted(listdir(dir)) if isfile(join(dir, f))]

def getDocIDToDocNameMap(dir):

"""

gets the map of docID to docName

parameters:

dir (str) : directroy which contains all files

return:

dict of docID to docName of docs in the directory given by user

"""

return {i:x for i, x in enumerate([f for f in sorted(listdir(dir)) if isfile(join(dir, f))])}

from urllib.request import urlopen

urls=[

"https://shakespeare.folger.edu/downloads/txt/the-winters-tale\_TXT\_FolgerShakespeare.txt",

"https://shakespeare.folger.edu/downloads/txt/venus-and-adonis\_TXT\_FolgerShakespeare.txt",

"https://shakespeare.folger.edu/downloads/txt/the-two-noble-kinsmen\_TXT\_FolgerShakespeare.txt",

"https://shakespeare.folger.edu/downloads/txt/the-two-gentlemen-of-verona\_TXT\_FolgerShakespeare.txt",

"https://shakespeare.folger.edu/downloads/txt/twelfth-night\_TXT\_FolgerShakespeare.txt",

"https://shakespeare.folger.edu/downloads/txt/troilus-and-cressida\_TXT\_FolgerShakespeare.txt",

"https://shakespeare.folger.edu/downloads/txt/titus-andronicus\_TXT\_FolgerShakespeare.txt",

"https://shakespeare.folger.edu/downloads/txt/timon-of-athens\_TXT\_FolgerShakespeare.txt",

"https://shakespeare.folger.edu/downloads/txt/the-tempest\_TXT\_FolgerShakespeare.txt",

"https://shakespeare.folger.edu/downloads/txt/romeo-and-juliet\_TXT\_FolgerShakespeare.txt"

]

l=[]

for url in urls:

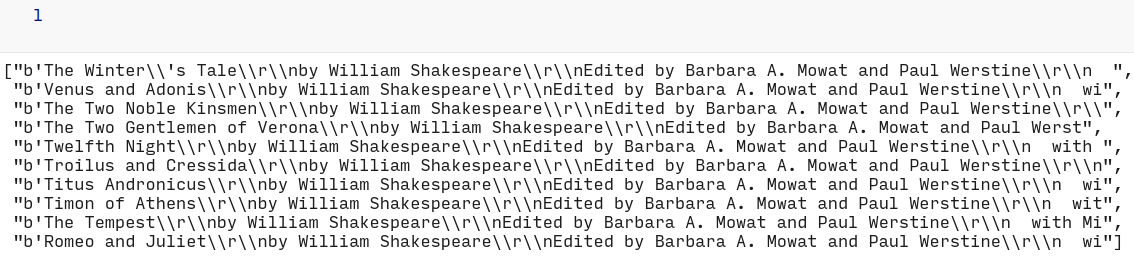
textPage = urlopen(url)

l.append(textPage.read())

# inverted index becomes to long so took only first 100 characters

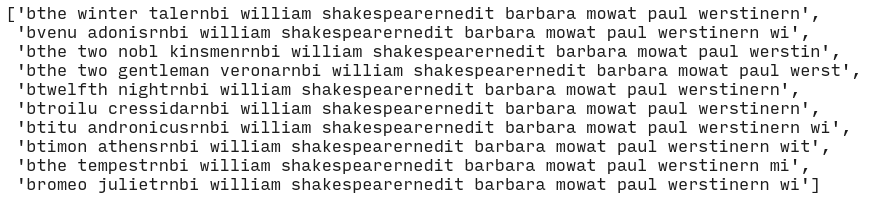
for i in range(len(l)):

l[i]=str(l[i])[:100]

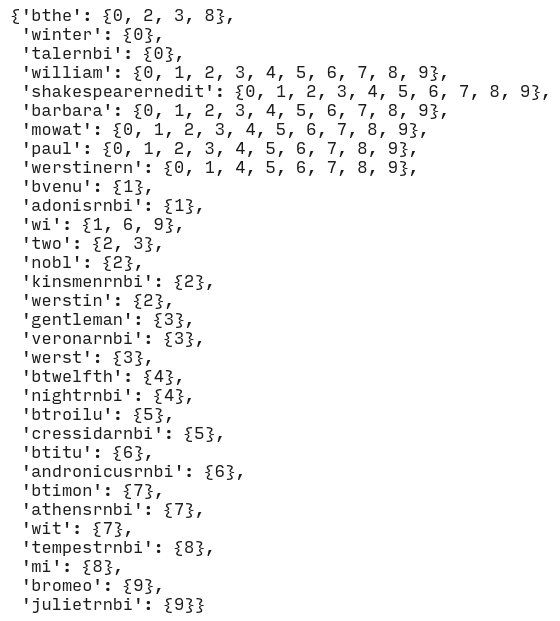


preprocessed\_text=preprocess(l)

preprocessed\_text

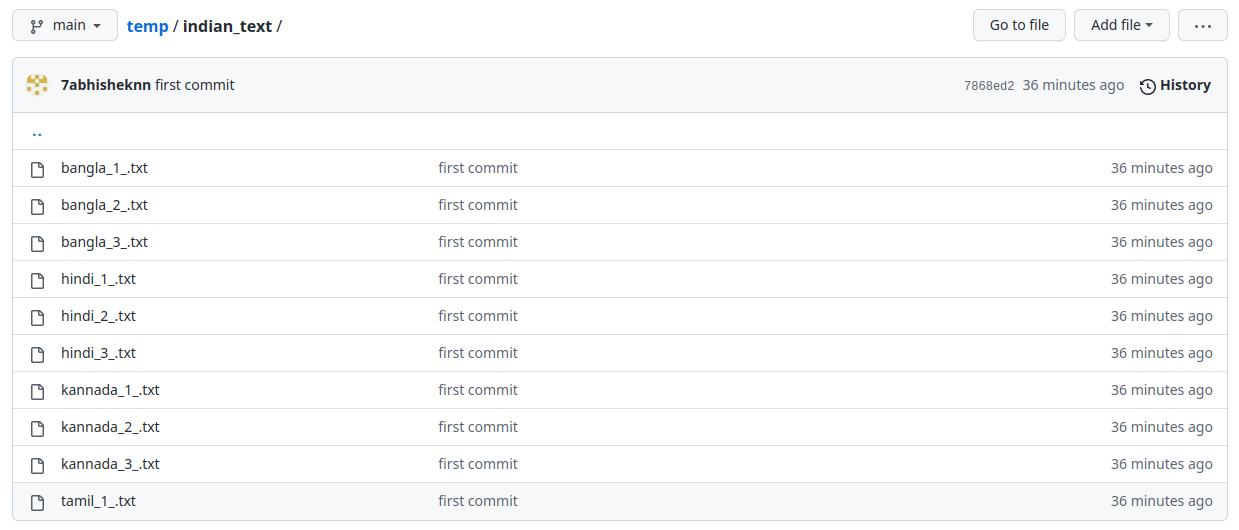


generateInvertedIndexDict(preprocessed\_text)



**Challenging Exercise-1:**

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



# direct web link to indian text file was not there so i made github folder containing files

urls=[

"https://raw.githubusercontent.com/7abhisheknn/temp/main/indian\_text/bangla\_1\_.txt",

"https://raw.githubusercontent.com/7abhisheknn/temp/main/indian\_text/bangla\_2\_.txt",

"https://raw.githubusercontent.com/7abhisheknn/temp/main/indian\_text/bangla\_3\_.txt",

"https://raw.githubusercontent.com/7abhisheknn/temp/main/indian\_text/hindi\_1\_.txt",

"https://raw.githubusercontent.com/7abhisheknn/temp/main/indian\_text/hindi\_2\_.txt",

"https://raw.githubusercontent.com/7abhisheknn/temp/main/indian\_text/hindi\_3\_.txt",

"https://raw.githubusercontent.com/7abhisheknn/temp/main/indian\_text/kannada\_1\_.txt",

"https://raw.githubusercontent.com/7abhisheknn/temp/main/indian\_text/kannada\_2\_.txt",

"https://raw.githubusercontent.com/7abhisheknn/temp/main/indian\_text/kannada\_3\_.txt",

"https://raw.githubusercontent.com/7abhisheknn/temp/main/indian\_text/tamil\_1\_.txt",

]

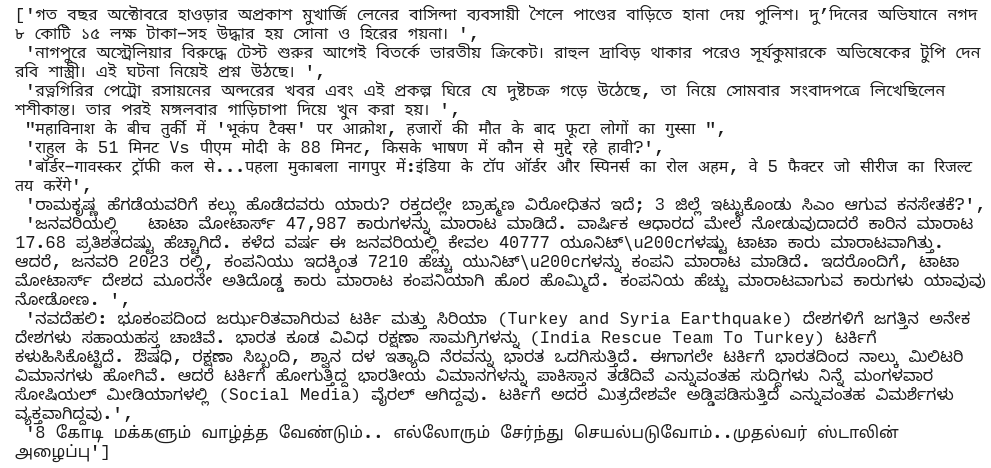
l=[]

for url in urls:

textPage = urlopen(url)

l.append(textPage.read().decode('utf-8'))

l



# since the data is indian we cannot apply all preprocessing steps

def preprocess\_indian\_text(corpus):

normalized\_corpus = []

# normalize each document in the corpus

for doc in corpus:

doc = remove\_numbers(doc)

doc = remove\_punctuation(doc)

doc = remove\_whitespace(doc)

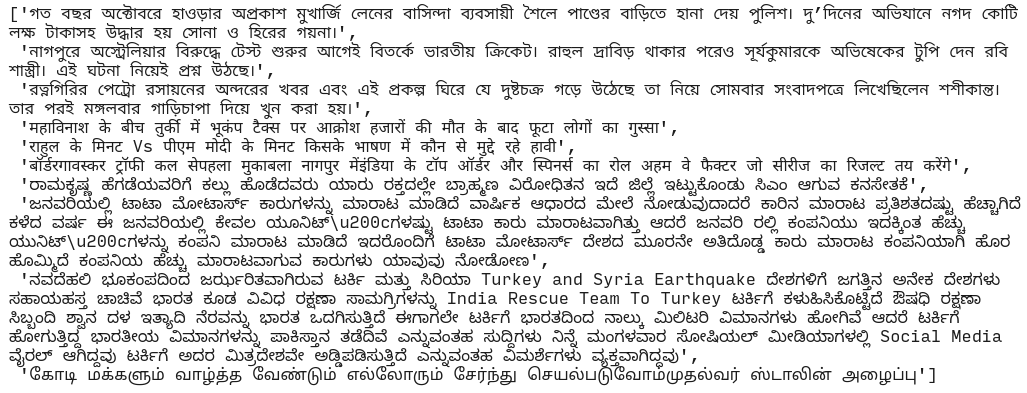
doc = remove\_extra\_newlines(doc)

normalized\_corpus.append(doc)

return normalized\_corpus

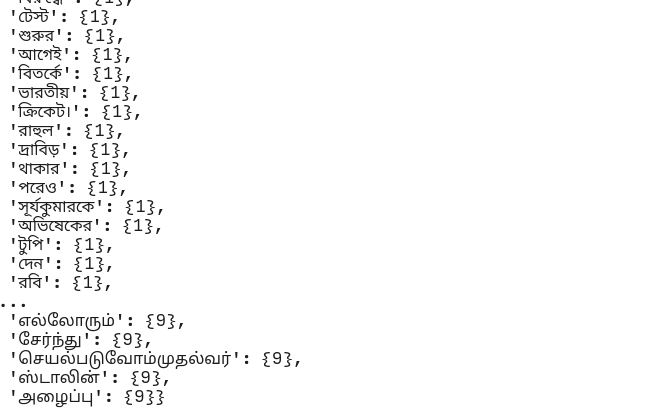
preprocessed\_text=preprocess\_indian\_text(l)

preprocessed\_text



generateInvertedIndexDict(preprocessed\_text)





**Challenging Exercise-2:**

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.

import requests

import textract

urls=[

"https://github.com/7abhisheknn/temp/raw/main/different\_format\_documents/1.odt",

"https://github.com/7abhisheknn/temp/raw/main/different\_format\_documents/2.odt",

"https://github.com/7abhisheknn/temp/raw/main/different\_format\_documents/3.odt",

"https://github.com/7abhisheknn/temp/raw/main/different\_format\_documents/4.odt",

"https://github.com/7abhisheknn/temp/raw/main/different\_format\_documents/1.docx",

"https://github.com/7abhisheknn/temp/raw/main/different\_format\_documents/2.docx",

"https://github.com/7abhisheknn/temp/raw/main/different\_format\_documents/3.docx",

"https://github.com/7abhisheknn/temp/raw/main/different\_format\_documents/1.pdf",

"https://github.com/7abhisheknn/temp/raw/main/different\_format\_documents/2.pdf",

"https://github.com/7abhisheknn/temp/raw/main/different\_format\_documents/3.pdf",

]

l=[]

for url in urls:

response = requests.get(url)

saveFile=""

if (url[-3:]=="odt"):

saveFile="temp.odt"

elif (url[-3:]=="ocx"):

saveFile="temp.docx"

else:

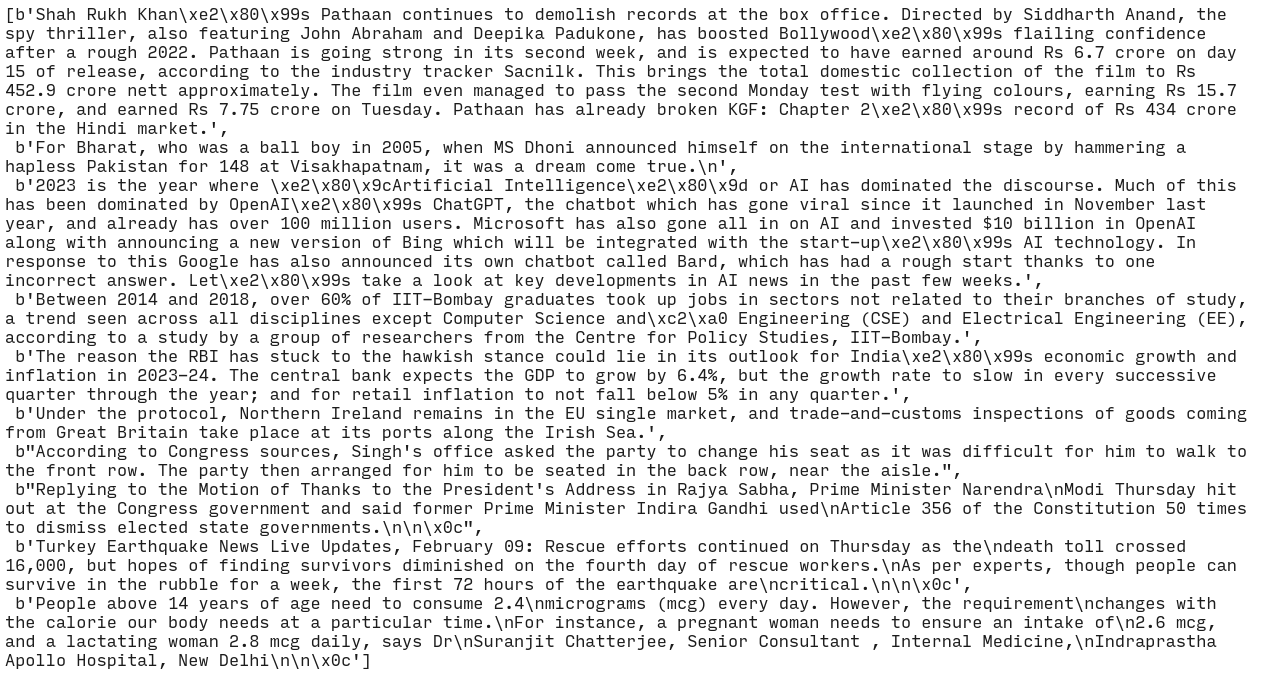
saveFile="temp.pdf"

open(saveFile, "wb").write(response.content)

text = textract.process(saveFile)

l.append(text)

l

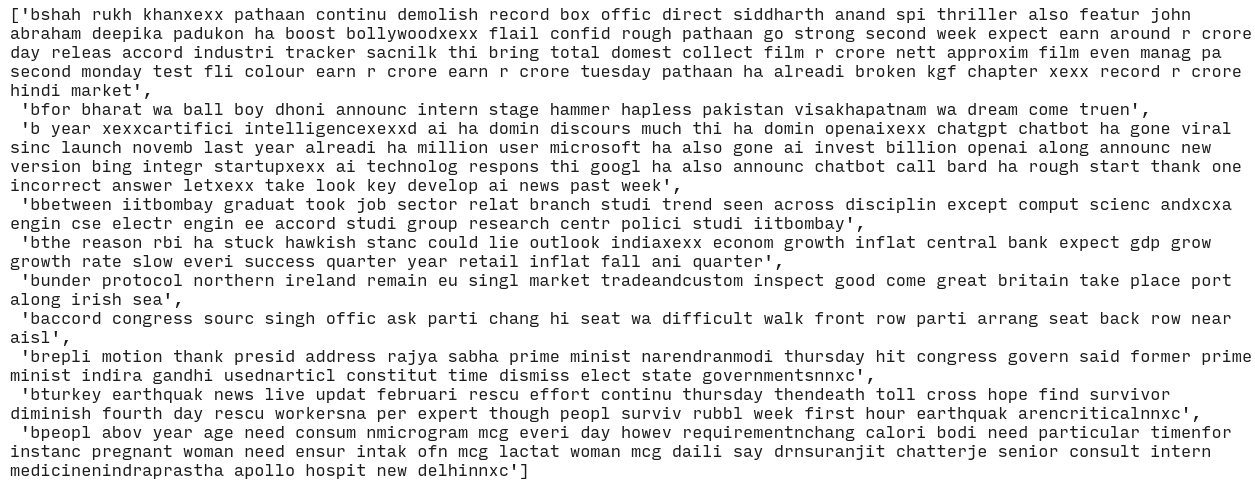


for i in range(len(l)):

l[i]=str(l[i])

preprocessed\_text=preprocess(l)

preprocessed\_text



generateInvertedIndexDict(preprocessed\_text)

