IR ASSIGNMENT 1.(REPORT)

Group No 80:

Group members:

1.Madiha Tariq (MT21125)
2.Shivnath Singh Gaur (MT21085)

General Assumptions

Here we are taking some assumption that mention in assignment as well like we are traversing query from left to right .we are processing for variable length query and we are also handling the case in which user enter some word that are not present in document and the operator that are not valid.So these are the general assumptions that we are assuming here.

Q1.

Preprocessing

for 1st question we after unzipping data set we are storing it in google drive simple folder .then 1stly we we are converting each file into lower case then remove punctuation using regex from NLTK and then tokenized that file into tokens and then remove stopwords using NLTK stopwords also remove digits from data set.
These are the common library that we are using for preprocessing

```
import os
import glob
import pandas as pd
import numpy as np
import nltk
import re
import ast
nltk.download('stopwords')
import nltk
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

import nltk
nltk.download('stopwords')


nltk.download('punkt')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from gensim.parsing.preprocessing import remove_stopwords
from nltk.stem import WordNetLemmatizer
import nltk
```

Methodology

Here we are using two dictionary 1st dictionary is used to keep track of term to documents id(means which word present in which document) and the 2nd dictionary named as posting_dict keep track of term in which document these are present in data set.

So here we are reading all file one-one and if the word already present in the dictionary or unigram inverted index then check the document id of that word if the doc_id already present in unigram index nothing to do else add that document id into dictionary corresponding to that word in this way we implementing unigram inverted index that contain list of documents in which a particular word present.

Result

```
Enter no of Query  2
Enter  query to process  telephone roads
Enter operator  Or
Expected query
['telephone', 'OR', 'road']
['capital.txt', 'egg-bred.txt', 'hedgehog.txt', 'mitch.txt',
'films_gl.txt', 'bnbeg2.4.txt', 'stuf10.txt', 'recipe.009',
'tpquotes.txt', 'aeonint.txt', 'classicm.hum', 'japantv.txt', 'anime.lif',
'homermmm.txt', 'facedeth.txt', 'clancy.txt', 'mov_rail.txt', 'widows',
'tpquote2.txt', 'adt_miam.txt', 'epi_tton.txt', 'lost.txt',
'gd_hhead.txt', 'amazing.epi', 'allfam.epi', 'bnbguide.txt',
'epi_bnb.txt', 'gd_tznew.txt', 'ateam.epi', 'a-team', 'epi_rns.txt',
'epiquest.txt', 'avengers.lis', 'moore.txt', 'tribble.hum', 'teevee.hum',
'popmusi.hum', 'odearakk.hum', 'empeval.txt', 'finalexm.hum',
'forsooth.hum', 'italoink.txt', 'jrrt.riddle', 'mydaywss.hum',
'abbott.txt', 'boston.geog', 'cartoon_.txt', 'chainltr.txt',
'anorexia.txt', 'btscke02.des', 'food', 'orgfrost.bev', 'epikarat.txt',
'twinpeak.txt', 'gd_alf.txt', 'outlimit.txt', 'basehead.txt', 'exidy.txt',
'hackingcracking.txt', 'hackmorality.txt', 'happyhack.txt', 'jimhood.txt',
'lipkovits.txt', 'mog-history', 'necropls.txt', 'radexposed.txt',
'reeves.txt', 'sawyer.txt', 'whatbbs', 'yuban.txt', 'truths.hum',
'vaguemag.90s', 'valujet.txt', 'vegkill.txt', 'washroom.txt',
'wimptest.txt', 'wrdnws3.txt', 'wrdnws5.txt', 'wrdnws6.txt',
'wrdnws7.txt', 'wrdnws8.txt', 'y.txt', 'yogisays.txt', 'solviets.hum',
'strsdiet.txt', 'st_silic.txt', 'suicide2.txt', 'taping.hum',
'televisi.hum', 'tfpoems.hum', 'thecube.hum', 'thermite.ana',
'top10st1.txt', 'shuttleb.hum', 'skincat', 'smackjok.hum', 'smokers.txt',
```

'smurfs.cc', 'quest.hum', 'rabbit.txt', 'racist.net', 'rapmastr.hum',
'readme.bat', 'reagan.hum', 'reddwarf.sng', 'renorthr.txt', 'robot.tes',
'poopie.txt', 'prac1.jok', 'prac4.jok', 'prac3.jok', 'practica.txt',
'pracjoke.txt', 'prawblim.hum', 'problem.txt', 'proof.met', 'nigel.2',
'nigel.5', 'nigel.6', 'nigel.7', 'nigel.10', 'nuke.hum', 'nukwaste',
'one.par', 'ookpik.hum', 'o-ttalk.hum', 'paddingurpapers.txt',
'disaster.hum', 'disclym.txt', 'd-ned.hum', 'dubltalk.jok',
'dthought.txt', 'enlightenment.txt', 'excuse30.txt', 'failure.txt',
'fegg!int.txt', 'female.jok', 'free-cof.fee', 'fuck!.txt', 'headlnrs',
'herb!.hum', 'hoosier.txt', 'horoscop.jok', 'htswfren.txt',
'humpty.dumpty', 'insults1.txt', 'insuranc.sty', 'jc-elvis.inf', 'johann',
'jokes.txt', 'just2', 'kilsmur.hum', 'lawyers.txt', 'liceprof.sty',
'looser.hum', 'losers86.hum', 'lozers', 'lucky.cha', 'marines.hum',
'misery.hum', 'modstup', 'moslem.txt', 'nasaglenn.txt', 'acronym.txt',
'adcopy.hum', 'alcatax.txt', 'anim_lif.txt', 'argotdic.txt', 'bad.jok',
'bagelope.txt', 'bored.txt', 'browneco.hum', 'cartoon.law',
'cartoon.laws', 'cartwb.son', 'catballs.hum', 'college.hum',
'computer.txt', 'cookie.1', 'contract.moo', 'cowexplo.hum', 'dead3.txt',
'dead5.txt', 'dead-r', 'bingbong.hum', 'ghostfun.hum', 'poets.hum',
'alcohol.hum', 'beginn.ers', 'bread.txt']
No of Document match 190
No comaparision 189

Enter query to process candy bar
Enter operator andNOt
Expected query
['candy', 'ANDNOT', 'bar']
['chili.txt', 'focaccia.brd', 'hamburge.nam', 'qttofu.vgn',
'strattma.txt', 'sfmovie.txt', 'epi_tton.txt', 'ateam.epi', 'a-team',
'epi_rns.txt', 'missdish', 'y.txt', 'solviets.hum', 'subrdead.hum',
'shuttleb.hum', 'one.par', 'o-ttalk.hum', 'eskimo.nel', 'fartting.txt',
'misc.1', 'acronym.lis', 'bmdn01.txt', 'cars.txt', 'chinesec.hum',
'cookie.1', 'drugshum.hum', 'packard.txt']
No of Document match 27
No comaparision 181

```
Enter no of Query2
Enter  query to process  telephone roads
Enter operator  Or
Expected query
['telephone', 'OR', 'road']
['capital.txt', 'egg-bred.txt', 'hedgehog.txt', 'mitch.txt', 'films_gl.txt', 'bnbeg2.4.txt', 'stuf10.txt',
No of Document match 190
No comaparision 189
Enter  query to process  candy bar
Enter operator  andNOt
Expected query
['candy', 'ANDNOT', 'bar']
['chili.txt', 'focaccia.brd', 'hamburge.nam', 'qttofu.vgn', 'strattma.txt', 'sfmovie.txt', 'epi_tton.txt',
No of Document match 27
No comaparision 181
```

Q2

Preprocessing

Preprocessing for 1st and 2nd question somewhat same as we wrote above here extra thing is that we also used strip() to remove extra space between words and all we used in 1st also used here

Methodology

In the first question we are using only a unigram inverted index that maintains a list of document id for each word.but here in the 2nd question we with that inverted index we also need to maintain relative position of word in that particular document so it is known as positional index.
Here we are using nested dictionary in 1st dictionary word as a key and then values as a another dictionary that contain doc_id as a key and their position in that document as a list

Here we are searching for for every document one-one if there a word that is not present present dictionary then add it to dictionary and if word if present then add document id and relative position in which that is present currently and this way complete all the document;

Results

+ Code + Text

user interface

```
query = input("Enter phares query : ")
query=list(query_preprocessing(query))
print("Expected query ",query);
Output(query)

#'aerospace', 'engineer', 'wear'
```

```
Enter phares query : put small amounts in blender;
Expected query  ['put', 'small', 'amount', 'blender']
No of document match  2
No of comparision  2266
['mrsfield', 'msfields.txt']
```

```
query = input("Enter phares query : ")
query=list(query_preprocessing(query))
print("Expected query ",query);
Output(query)

#'aerospace', 'engineer', 'wear'
```

```
Enter phares query : aerospace engineer wear
Expected query  ['aerospace', 'engineer', 'wear']
No of document match  1
No of comparision  188
['calif.hum']
```