INFORMATION RETRIEVAL - CSE 508
ASSIGNMENT 1

SUBMITTED BY:Abhinav garg- MT22002
Riya garg - MT22058
Saksham Nautiyal- MT22061

ASSUMPTIONS:

1. **All files have text based content that we need to preprocess.**

Q1. <u>DATA PREPROCESSING:</u>

LIBRARIES USED:

- **Nltk**
- **Beautiful Soup**
- **punkt**

(i)First we print 5 files before extraction of data between tags.
To extract contents between contents between the <TITLE>...</TITLE> and <TEXT>...</TEXT> tags we have used the **<u>Beautiful Soup</u>** library which provides simple methods for navigating, searching, and modifying a parse tree in HTML, XML files.
Then we concatenate the 2 strings using blank space and rewrite them into the files.

```
from bs4 import BeautifulSoup
for f1 in list1:
    f=open(f'{path}/{f1}',mode='r')
    data = f.read()
    f.close()
    title1=""
    text=""
    bsText = BeautifulSoup(data, 'html.parser')
        #title1 = beautifulSoupText.findAll('TITLE').text
        #text = beautifulSoupText.findAll('TEXT').text
       # print(beautifulSoupText)
    for tag in bsText.findAll('title'):
        title1 = bsText.find(tag.name).text
           #print(title1)
    for tag in bsText.findAll('text'):
        text = bsText.find(tag.name).text
    f=open(f'{path}/{f1}',mode='w')
    x = title1 +" "+ text
    f.write(x);
    f.close();
```

(ii)Preprocessing:

1. **Lowercase the text**: For this we have used the String lower() method. It converts all uppercase characters in a string into lowercase characters and returns it.
2. **Perform tokenization:** NLTK contains a module called tokenize with a word_tokenize() method that will help us split a text into tokens.
3. **Remove stopwords :** for this stopwords library from nltk.corpus
4. **Remove punctuations:** For this we use regular expressions and string function.
5. **Remove blank space tokens** using string function.

Output after preprocessing:

```
File  1


experimental investigation aerodynamics wing slipstream experimental study wing propeller slipstream made order determine spanwise distribution lift increase due slipstream different angles attack wing different free stream slipstream velocity ratios results intended part evaluation basis different theoretical treatments problem comparative span loading curves together supporting evidence showed substantial part lift increment produced slipstream due destalling boundarylayercontrol effect integrated remaining lift increment subtracting destalling lift found agree well potential flow theory empirical evaluation destalling effects made specific configuration experiment


File  2


simple shear flow past flat plate incompressible fluid small viscosity study highspeed viscous flow past twodimensional body usually necessary consider curved shock wave emitting nose leading edge body consequently exists inviscid rotational flow region shock wave boundary layer situation arises instance study hypersonic viscous flow past flat plate situation somewhat different prandtl s classical boundarylayer problem prandtls original problem inviscid free stream outside boundary layer irrotational hypersonic boundarylayer problem inviscid free stream must considered rotational possible effects vorticity recently discussed ferri libby present paper simple shear flow past flat plate fluid small viscosity investigated shown problem treated boundarylayer approximation novel feature free stream constant vorticity discussion restricted twodimensional incompressible steady flow
```

Ques 2: Boolean Queries:

Assumption: Stop words are irrelevant for creating a positional or inverted index.

UNIGRAM INVERTED INDEX:

To store the document ids for a given term, we created a dictionary we call postings ={}.On each distinct word, we have created a for loop. If a word is already in our dictionary postings, we append the document id to the list if it is. We shall create a new value pair if the term has just appeared.
Key in postings is current word and value is list which contains the document ids in which the word is present.

After creating all the posting lists for all the terms, we store them using the "unigram postings" Python pickle.

We have created four functions, OR_merge(), AND_ merge(), AND_NOT_merge(), and OR_NOT(), which when given two parameters—the document ids lists of two terms—perform the necessary logic.
After accepting the input in the required format, we first load the pickle we earlier saved in order to retrieve/load the posting lists of the terms.
Next we call the function process query(), which checks the operators list and calls the appropriate function based on its findings.
For the first two terms of the query, we first execute the process query() method, after which we save the result and the number of comparisons.
Then, after doing the process with the term posts and results for the remaining terms, we have the final result merged list and the minimal number of comparisons.
In order to retrieve the names of the documents, we also maintain a doc map dictionary using Python pickle, which maps document names to document ids.

```
Query  1 :
['boundary', 'layer']
['OR NOT']
Query  1 : boundary OR NOT layer
Number of documents retrieved for query  1 :  1400
Names of documents retrieved for query  1 :  cranfield0001, cranfield0002, cranfield0003, cranfield0004, cranfield0005, cranfield0006, cranfield0007,
cranfield0008, cranfield0009, cranfield0010, cranfield0011, cranfield0012, cranfield0013, cranfield0014, cranfield0015, cranfield0016, cranfield0017,
cranfield0018, cranfield0019, cranfield0020, cranfield0021, cranfield0022, cranfield0023, cranfield0024, cranfield0025, cranfield0026, cranfield0027,
cranfield0028, cranfield0029, cranfield0030, cranfield0031, cranfield0032, cranfield0033, cranfield0034, cranfield0035, cranfield0036, cranfield0037,
cranfield0038, cranfield0039, cranfield0040, cranfield0041, cranfield0042, cranfield0043, cranfield0044, cranfield0045, cranfield0046, cranfield0047,
cranfield0048, cranfield0049, cranfield0050, cranfield0051, cranfield0052, cranfield0053, cranfield0054, cranfield0055, cranfield0056, cranfield0057,
cranfield0058, cranfield0059, cranfield0060, cranfield0061, cranfield0062, cranfield0063, cranfield0064, cranfield0065, cranfield0066, cranfield0067,
cranfield0068, cranfield0069, cranfield0070, cranfield0071, cranfield0072, cranfield0073, cranfield0074, cranfield0075, cranfield0076, cranfield0077,
cranfield0078, cranfield0079, cranfield0080, cranfield0081, cranfield0082, cranfield0083, cranfield0084, cranfield0085, cranfield0086, cranfield0087,
cranfield0088, cranfield0089, cranfield0090, cranfield0091, cranfield0092, cranfield0093, cranfield0094, cranfield0095, cranfield0096, cranfield0097,
cranfield0098, cranfield0099, cranfield0100, cranfield0101, cranfield0102, cranfield0103, cranfield0104, cranfield0105, cranfield0106, cranfield0107,
cranfield0108, cranfield0109, cranfield0110, cranfield0111, cranfield0112, cranfield0113, cranfield0114, cranfield0115, cranfield0116, cranfield0117,
cranfield0118, cranfield0119, cranfield0120, cranfield0121, cranfield0122, cranfield0123, cranfield0124, cranfield0125, cranfield0126, cranfield0127,
cranfield0128, cranfield0129, cranfield0130, cranfield0131, cranfield0132, cranfield0133, cranfield0134, cranfield0135, cranfield0136, cranfield0137,
cranfield0138, cranfield0139, cranfield0140, cranfield0141, cranfield0142, cranfield0143, cranfield0144, cranfield0145, cranfield0146, cranfield0147,
cranfield0148, cranfield0149, cranfield0150, cranfield0151, cranfield0152, cranfield0153, cranfield0154, cranfield0155, cranfield0156, cranfield0157,
cranfield0158, cranfield0159, cranfield0160, cranfield0161, cranfield0162, cranfield0163, cranfield0164, cranfield0165, cranfield0166, cranfield0167,
cranfield0168, cranfield0169, cranfield0170, cranfield0171, cranfield0172, cranfield0173, cranfield0174, cranfield0175, cranfield0176, cranfield0177,
cranfield0178, cranfield0179, cranfield0180, cranfield0181, cranfield0182, cranfield0183, cranfield0184, cranfield0185, cranfield0186, cranfield0187,
cranfield0188, cranfield0189, cranfield0190, cranfield0191, cranfield0192, cranfield0193, cranfield0194, cranfield0195, cranfield0196, cranfield0197,
cranfield0198, cranfield0199, cranfield0200, cranfield0201, cranfield0202, cranfield0203, cranfield0204, cranfield0205, cranfield0206, cranfield0207,
cranfield0198, cranfield0199, cranfield0200, cranfield0201, cranfield0202, cranfield0203, cranfield0204, cranfield0205, cranfield0206, cranfield0207,
cranfield0208, cranfield0209, cranfield0210, cranfield0211, cranfield0212, cranfield0213, cranfield0214, cranfield0215, cranfield0216, cranfield0217,
cranfield0218, cranfield0219, cranfield0220, cranfield0221, cranfield0222, cranfield0223, cranfield0224, cranfield0225, cranfield0226, cranfield0227,
cranfield0228, cranfield0229, cranfield0230, cranfield0231, cranfield0232, cranfield0233, cranfield0234, cranfield0235, cranfield0236, cranfield0237,
cranfield0238, cranfield0239, cranfield0240, cranfield0241, cranfield0242, cranfield0243, cranfield0244, cranfield0245, cranfield0246, cranfield0247,
cranfield0248, cranfield0249, cranfield0250, cranfield0251, cranfield0252, cranfield0253, cranfield0254, cranfield0255, cranfield0256, cranfield0257,
cranfield0258, cranfield0259, cranfield0260, cranfield0261, cranfield0262, cranfield0263, cranfield0264, cranfield0265, cranfield0266, cranfield0267,
cranfield0268, cranfield0269, cranfield0270, cranfield0271, cranfield0272, cranfield0273, cranfield0274, cranfield0275, cranfield0276, cranfield0277,
cranfield0278, cranfield0279, cranfield0280, cranfield0281, cranfield0282, cranfield0283, cranfield0284, cranfield0285, cranfield0286, cranfield0287,
cranfield0288, cranfield0289, cranfield0290, cranfield0291, cranfield0292, cranfield0293, cranfield0294, cranfield0295, cranfield0296, cranfield0297,
cranfield0298, cranfield0299, cranfield0300, cranfield0301, cranfield0302, cranfield0303, cranfield0304, cranfield0305, cranfield0306, cranfield0307,
cranfield0308, cranfield0309, cranfield0310, cranfield0311, cranfield0312, cranfield0313, cranfield0314, cranfield0315, cranfield0316, cranfield0317,
cranfield0318, cranfield0319, cranfield0320, cranfield0321, cranfield0322, cranfield0323, cranfield0324, cranfield0325, cranfield0326, cranfield0327,
cranfield0328, cranfield0329, cranfield0330, cranfield0331, cranfield0332, cranfield0333, cranfield0334, cranfield0335, cranfield0336, cranfield0337,
cranfield0338, cranfield0339, cranfield0340, cranfield0341, cranfield0342, cranfield0343, cranfield0344, cranfield0345, cranfield0346, cranfield0347,
cranfield0348, cranfield0349, cranfield0350, cranfield0351, cranfield0352, cranfield0353, cranfield0354, cranfield0355, cranfield0356, cranfield0357,
cranfield0358, cranfield0359, cranfield0360, cranfield0361, cranfield0362, cranfield0363, cranfield0364, cranfield0365, cranfield0366, cranfield0367,
cranfield0368, cranfield0369, cranfield0370, cranfield0371, cranfield0372, cranfield0373, cranfield0374, cranfield0375, cranfield0376, cranfield0377,
cranfield0378, cranfield0379, cranfield0380, cranfield0381, cranfield0382, cranfield0383, cranfield0384, cranfield0385, cranfield0386, cranfield0387,
cranfield0388, cranfield0389, cranfield0390, cranfield0391, cranfield0392, cranfield0393, cranfield0394, cranfield0395, cranfield0396, cranfield0397,
cranfield0398, cranfield0399, cranfield0400, cranfield0401, cranfield0402, cranfield0403, cranfield0404, cranfield0405, cranfield0406, cranfield0407,
cranfield0408, cranfield0409, cranfield0410, cranfield0411, cranfield0412, cranfield0413, cranfield0414, cranfield0415, cranfield0416, cranfield0417,
cranfield0418, cranfield0419, cranfield0420, cranfield0421, cranfield0422, cranfield0423, cranfield0424, cranfield0425, cranfield0426, cranfield0427,
cranfield0428, cranfield0429, cranfield0430, cranfield0431, cranfield0432, cranfield0433, cranfield0434, cranfield0435, cranfield0436, cranfield0437,
cranfield0438, cranfield0439, cranfield0440, cranfield0441, cranfield0442, cranfield0443, cranfield0444, cranfield0445, cranfield0446, cranfield0447,
cranfield0448, cranfield0449, cranfield0450, cranfield0451, cranfield0452, cranfield0453, cranfield0454, cranfield0455, cranfield0456, cranfield0457,
cranfield0458, cranfield0459, cranfield0460, cranfield0461, cranfield0462, cranfield0463, cranfield0464, cranfield0465, cranfield0466, cranfield0467,
cranfield1168, cranfield1169, cranfield1170, cranfield1171, cranfield1172, cranfield1173, cranfield1174, cranfield1175, cranfield1176, cranfield1177,
cranfield1178, cranfield1179, cranfield1180, cranfield1181, cranfield1182, cranfield1183, cranfield1184, cranfield1185, cranfield1186, cranfield1187,
cranfield1188, cranfield1189, cranfield1190, cranfield1191, cranfield1192, cranfield1193, cranfield1194, cranfield1195, cranfield1196, cranfield1197,
cranfield1198, cranfield1199, cranfield1200, cranfield1201, cranfield1202, cranfield1203, cranfield1204, cranfield1205, cranfield1206, cranfield1207,
cranfield1208, cranfield1209, cranfield1210, cranfield1211, cranfield1212, cranfield1213, cranfield1214, cranfield1215, cranfield1216, cranfield1217,
cranfield1218, cranfield1219, cranfield1220, cranfield1221, cranfield1222, cranfield1223, cranfield1224, cranfield1225, cranfield1226, cranfield1227,
cranfield1228, cranfield1229, cranfield1230, cranfield1231, cranfield1232, cranfield1233, cranfield1234, cranfield1235, cranfield1236, cranfield1237,
cranfield1238, cranfield1239, cranfield1240, cranfield1241, cranfield1242, cranfield1243, cranfield1244, cranfield1245, cranfield1246, cranfield1247,
cranfield1248, cranfield1249, cranfield1250, cranfield1251, cranfield1252, cranfield1253, cranfield1254, cranfield1255, cranfield1256, cranfield1257,
cranfield1258, cranfield1259, cranfield1260, cranfield1261, cranfield1262, cranfield1263, cranfield1264, cranfield1265, cranfield1266, cranfield1267,
cranfield1268, cranfield1269, cranfield1270, cranfield1271, cranfield1272, cranfield1273, cranfield1274, cranfield1275, cranfield1276, cranfield1277,
cranfield1278, cranfield1279, cranfield1280, cranfield1281, cranfield1282, cranfield1283, cranfield1284, cranfield1285, cranfield1286, cranfield1287,
cranfield1288, cranfield1289, cranfield1290, cranfield1291, cranfield1292, cranfield1293, cranfield1294, cranfield1295, cranfield1296, cranfield1297,
cranfield1298, cranfield1299, cranfield1300, cranfield1301, cranfield1302, cranfield1303, cranfield1304, cranfield1305, cranfield1306, cranfield1307,
cranfield1308, cranfield1309, cranfield1310, cranfield1311, cranfield1312, cranfield1313, cranfield1314, cranfield1315, cranfield1316, cranfield1317,
cranfield1318, cranfield1319, cranfield1320, cranfield1321, cranfield1322, cranfield1323, cranfield1324, cranfield1325, cranfield1326, cranfield1327,
cranfield1328, cranfield1329, cranfield1330, cranfield1331, cranfield1332, cranfield1333, cranfield1334, cranfield1335, cranfield1336, cranfield1337,
cranfield1338, cranfield1339, cranfield1340, cranfield1341, cranfield1342, cranfield1343, cranfield1344, cranfield1345, cranfield1346, cranfield1347,
cranfield1348, cranfield1349, cranfield1350, cranfield1351, cranfield1352, cranfield1353, cranfield1354, cranfield1355, cranfield1356, cranfield1357,
cranfield1358, cranfield1359, cranfield1360, cranfield1361, cranfield1362, cranfield1363, cranfield1364, cranfield1365, cranfield1366, cranfield1367,
cranfield1368, cranfield1369, cranfield1370, cranfield1371, cranfield1372, cranfield1373, cranfield1374, cranfield1375, cranfield1376, cranfield1377,
cranfield1378, cranfield1379, cranfield1380, cranfield1381, cranfield1382, cranfield1383, cranfield1384, cranfield1385, cranfield1386, cranfield1387,
cranfield1388, cranfield1389, cranfield1390, cranfield1391, cranfield1392, cranfield1393, cranfield1394, cranfield1395, cranfield1396, cranfield1397,
cranfield1398, cranfield1399, cranfield1400,
No. of comparisons required for query  1 :  1
```

QUES 3:

BIGRAM INVERTED INDEX:

To find the bigrams first we create a tuple containing two adjacent words. Then we are creating a dictionary as in ques 2 bi_postings={}. In this dictionary tuple of adjacent words is a key and value is a list which contains the document ids in which the tuple is present.
On each distinct tuple, we have created a for loop. If both words in tuple are already in our dictionary postings, we append the document id to the list if it is. We shall create a new value pair if the tuple has just appeared.
After creating all the posting lists for all the terms, we store them using the "bigram_ postings" Python pickle.

Positional index:

To store the document ids and locations of a given word, we created a dictionary named posInd ={}.
The current word serves as the dictionary's key, and the value is a list, where the first element of the list denotes frequency and the second element contains the dictionary. The dictionary's keys are the document ids in which the word appears, and its value is a list of the locations in the specified document where the current word can be found.
After creating all the posting lists for all the terms, we store them using the "positional_ postings" Python pickle.

To do a search in the bigram inverted index, we turn the query into a collection of tuples, or sets of two words, and then we process it by invoking the bigram_query() method, which performs an AND merge boolean query on every tuple in the collection before printing the results.

The process positional_query() method is called by the query and gets the tokens processed. It then checks the positional postings dictionary to see if the tokens are in adjacent positions and stores them in the list if they are.

For each query the output will be in the following format:

i. Number of documents retrieved using bigram inverted index

ii. Names of documents retrieved using bigram inverted index

iii. Number of documents retrieved using positional index

iv. Names of documents retrieved using positional index

```
Query  1 : flow past flat
Number of documents retrieved for query  1  using bigram inverted index: 7
Names of documents retrieved for query  1  using bigram inverted index:
cranfield0002,cranfield0003,cranfield0388,cranfield0389,cranfield1186,cranfield0308,cranfield0663,
Number of documents retrieved for query  1  using positional index: 7
Number of documents retrieved for query  1  using positional index:
cranfield0002,cranfield0003,cranfield0308,cranfield0388,cranfield0389,cranfield0663,cranfield1186,
```

Comparison

When a query is processed using the positional index we check for adjacent positions whereas using the bigram inverted index we check for combinations of words together which might lead to different documents where the words occur together but the total query does not occur together.

```
Query  1 : design highspeed flight
Number of documents retrieved for query  1  using bigram inverted index: 1
Names of documents retrieved for query  1  using bigram inverted index: cranfield0012,
Number of documents retrieved for query  1  using positional index: 0
Number of documents retrieved for query  1  using positional index:
```

Here in document number 12 the words design highspeed flight are not together but (design,highspeed) and (highspeed,flight) are present in different locations so using bigram inverted index it is printing that the query is present and using positional index it is shows not present.