CSE508 Information Retrieval
Winter 2023
Assignment-1

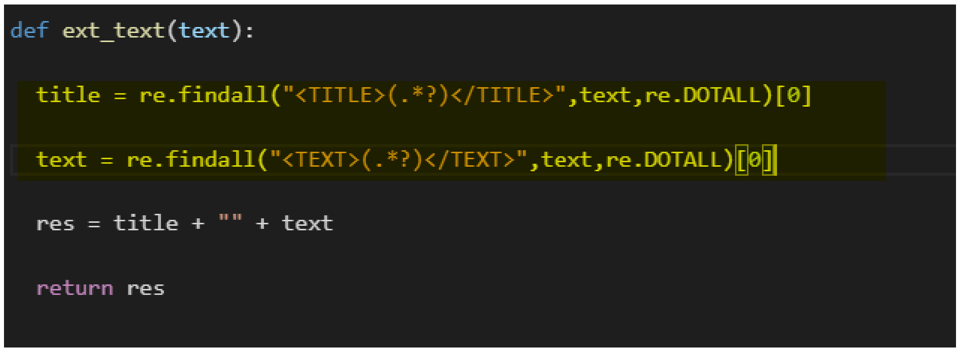
Submitted by:
Asmita Mukherjee (MT21115)
Sehban Fazili (MT21143)
Amaan Khan (2020388)

Q1. Data Preprocessing

A. Relevant Text Extraction:

Methodology:

First, we iterated through all the files in the dataset and read the text in them. We use regex in the following way in order to detect the text between the <TITLE>...</TITLE> and <TEXT>...</TEXT> tags.



```
def ext_text(text):

  title = re.findall("<TITLE>(.*?)</TITLE>",text,re.DOTALL)[0]

  text = re.findall("<TEXT>(.*?)</TEXT>",text,re.DOTALL)[0]

  res = title + "" + text

  return res
```

Results:

-----------Original-File-----------

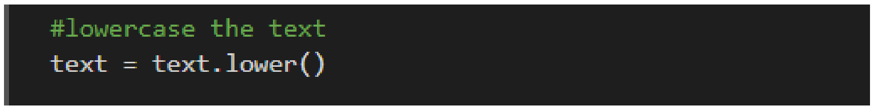
<DOC> <DOCNO> 2 </DOCNO> <TITLE> simple shear flow past a flat plate in an incompressible fluid of small viscosity . </TITLE> <AUTHOR> ting-yili </AUTHOR> <BIBLIO> department of aeronautical engineering, rensselaer polytechnic institute troy, n.y. </BIBLIO> <TEXT> in the study of high-speed viscous flow past a two-dimensional body it is usually necessary to consider a curved shock wave emitting from the nose or leading edge of the body . consequently, there exists an inviscid rotational flow region between the shock wave and the boundary layer . such a situation arises, for instance, in the study of the hypersonic viscous flow past a flat plate . the situation is somewhat different from prandtl's classical boundary-layer problem . in prandtl's original problem the inviscid free stream outside the boundary layer is irrotational while in a hypersonic boundary-layer problem the inviscid free stream must be considered as rotational . the possible effects of vorticity have been recently discussed by ferri and libby . in the present paper, the simple shear flow past a flat plate in a fluid of small viscosity is investigated . it can be shown that this problem can again be treated by the boundary-layer approximation, the only novel feature being that the free stream has a constant vorticity . the discussion here is restricted to two-dimensional incompressible steady flow . </TEXT> </DOC>

-----------New-File-----------

simple shear flow past a flat plate in an incompressible fluid of small viscosity . in the study of high-speed viscous flow past a two-dimensional body it is usually necessary to consider a curved shock wave emitting from the nose or leading edge of the body . consequently, there exists an inviscid rotational flow region between the shock wave and the boundary layer . such a situation arises, for instance, in the study of the hypersonic viscous flow past a flat plate . the situation is somewhat different from prandtl's classical boundary-layer problem . in prandtl's original problem the inviscid free stream outside the boundary layer is irrotational while in a hypersonic boundary-layer problem the inviscid free stream must be considered as rotational . the possible effects of vorticity have been recently discussed by ferri and libby . in the present paper, the simple shear flow past a flat plate in a fluid of small viscosity is investigated . it can be shown that this problem can again be treated by the boundary-layer approximation, the only novel feature being that the free stream has a constant vorticity . the discussion here is restricted to two-dimensional incompressible steady flow .

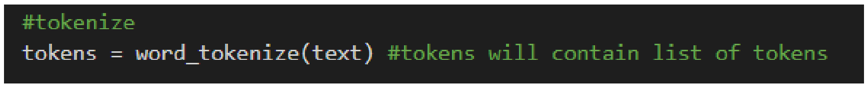
B. *Preprocessing:*

1. Lowercase the text:



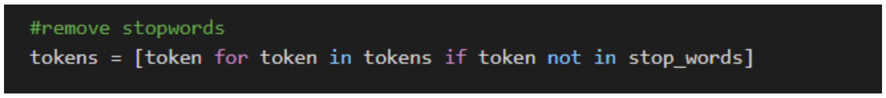
```
#lowercase the text
text = text.lower()
```

2. Perform tokenization:



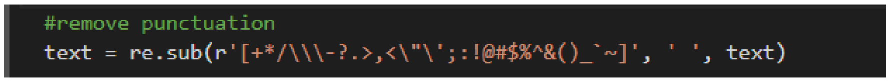
```
#tokenize
tokens = word_tokenize(text) #tokens will contain list of tokens
```

3. Remove stopwords:



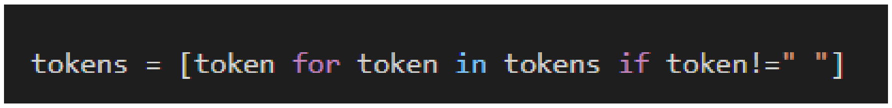
```
#remove stopwords
tokens = [token for token in tokens if token not in stop_words]
```

4. Remove punctuation:



```
#remove punctuation
text = re.sub(r'[+*/\\\-?.>,<\"\';:!@#$%^&()_`~]', ' ', text)
```

5. Remove blank space tokens:



```
tokens = [token for token in tokens if token!=" "]
```

Results:

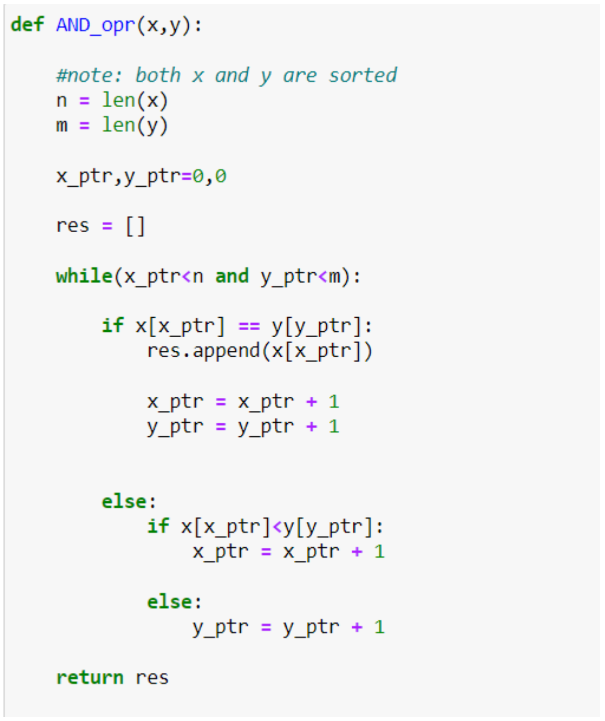
---------Before Preprocessing---------

experimental investigation of the aerodynamics of a wing in a slipstream .   an experimental study of a wing in a propeller slipstream was made in order to determine the spanwise distribution of the lift increase due to slipstream at different angles of attack of the wing and at different free stream to slipstream velocity ratios . the results were intended in part as an evaluation basis for different theoretical treatments of this problem .   the comparative span loading curves, together with supporting evidence, showed that a substantial part of the lift increment produced by the slipstream was due to a /destalling/ or boundary-layer-control effect . the integrated remaining lift increment, after subtracting this destalling lift, was found to agree well with a potential flow theory .   an empirical evaluation of the destalling effects was made for the specific configuration of the experiment .

---------After Preprocessing---------
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/ boundary-layer-control effect integrated remaining lift increment subtracting destalling lift found agree well potential flow theory empirical evaluation destalling effects made specific configuration experiment

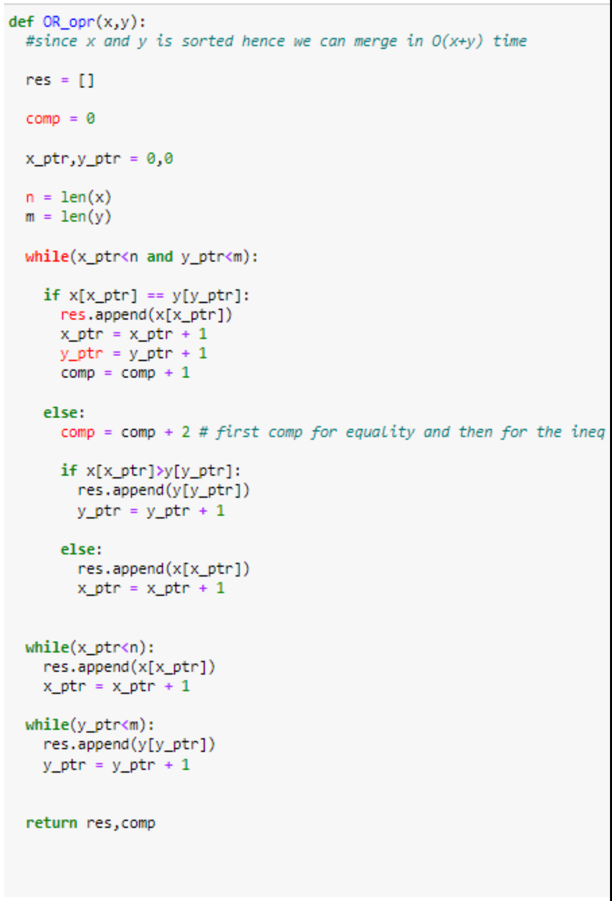
Q2. Boolean Queries:
Methodology:
- Implemented a unigram inverted index after pre-processing the given files.
    - Make a list of all the tokens in the corpus.
    - For each token, iterate the document and check if the token is present.
    - If present, add the document Id associated with the token.
- Determining the number of documents matching the query and the number of comparisons required for this task.
    - First, we preprocess the query and obtain the tokens in them
    - For each of the tokens, we associate the operator
    - For each query, we calculate from left to right
    - For each of the operators, we return the doc Id and the number of comparisons.
- The different operations are as follows:

AND Operation



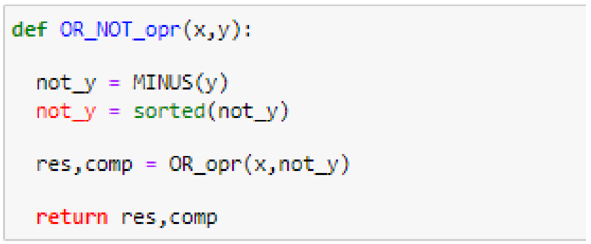
```python
def AND_opr(x,y):

    #note: both x and y are sorted
    n = len(x)
    m = len(y)

    x_ptr,y_ptr=0,0

    res = []

    while(x_ptr<n and y_ptr<m):

        if x[x_ptr] == y[y_ptr]:
            res.append(x[x_ptr])

            x_ptr = x_ptr + 1
            y_ptr = y_ptr + 1


        else:
            if x[x_ptr]<y[y_ptr]:
                x_ptr = x_ptr + 1

            else:
                y_ptr = y_ptr + 1

    return res
```

OR Operation



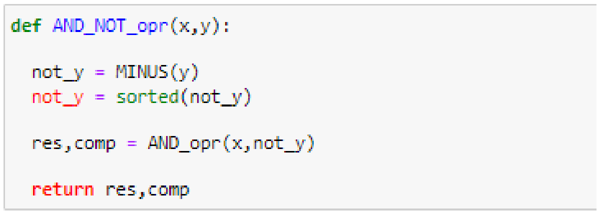
```python
def OR_opr(x,y):
  #since x and y is sorted hence we can merge in O(x+y) time

  res = []

  comp = 0

  x_ptr,y_ptr = 0,0

  n = len(x)
  m = len(y)

  while(x_ptr<n and y_ptr<m):

    if x[x_ptr] == y[y_ptr]:
      res.append(x[x_ptr])
      x_ptr = x_ptr + 1
      y_ptr = y_ptr + 1
      comp = comp + 1

    else:
      comp = comp + 2 # first comp for equality and then for the ineq

      if x[x_ptr]>y[y_ptr]:
        res.append(y[y_ptr])
        y_ptr = y_ptr + 1

      else:
        res.append(x[x_ptr])
        x_ptr = x_ptr + 1


  while(x_ptr<n):
    res.append(x[x_ptr])
    x_ptr = x_ptr + 1

  while(y_ptr<m):
    res.append(y[y_ptr])
    y_ptr = y_ptr + 1


  return res,comp
```

OR NOT Operation



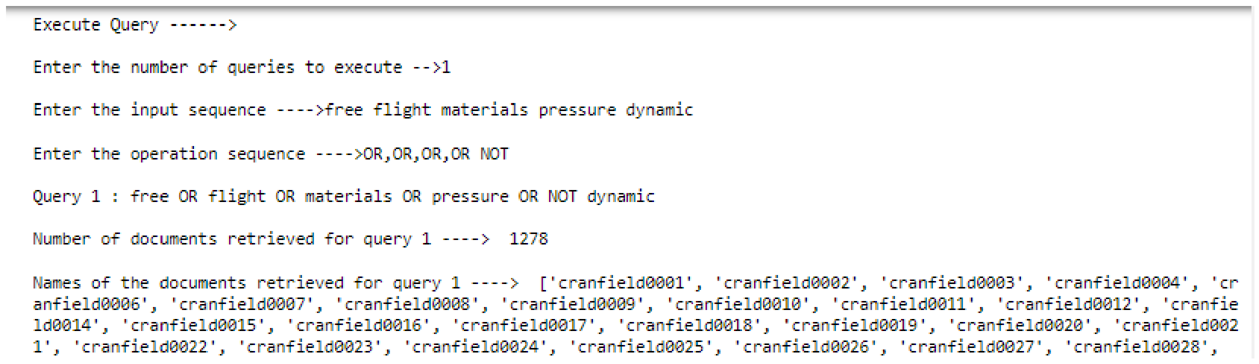
```python
def OR_NOT_opr(x,y):

  not_y = MINUS(y)
  not_y = sorted(not_y)

  res,comp = OR_opr(x,not_y)

  return res,comp
```

AND NOT Operation

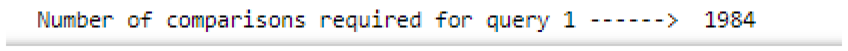


```python
def AND_NOT_opr(x,y):

  not_y = MINUS(y)
  not_y = sorted(not_y)

  res,comp = AND_opr(x,not_y)

  return res,comp
```

Result:



```
Execute Query ------>

Enter the number of queries to execute -->1

Enter the input sequence ---->free flight materials pressure dynamic

Enter the operation sequence ---->OR,OR,OR,OR NOT

Query 1 : free OR flight OR materials OR pressure OR NOT dynamic

Number of documents retrieved for query 1 ---->  1278

Names of the documents retrieved for query 1 ---->  ['cranfield0001', 'cranfield0002', 'cranfield0003', 'cranfield0004', 'cr
anfield0006', 'cranfield0007', 'cranfield0008', 'cranfield0009', 'cranfield0010', 'cranfield0011', 'cranfield0012', 'cranfie
ld0014', 'cranfield0015', 'cranfield0016', 'cranfield0017', 'cranfield0018', 'cranfield0019', 'cranfield0020', 'cranfield002
1', 'cranfield0022', 'cranfield0023', 'cranfield0024', 'cranfield0025', 'cranfield0026', 'cranfield0027', 'cranfield0028',
```



```
Number of comparisons required for query 1 ------>  1984
```

Q3. Phrase Queries:

A. Bigram inverted index:

   Methodology:

   - First Implemented a bigram inverted index after pre-processing the given files.
     - Make a list of all the bi-grams in the corpus.
     - For each bi-gram, iterate the document and check if the bi-gram is present.
     - If present, add the document Id associated with the bi-gram.

B. Positional Index:

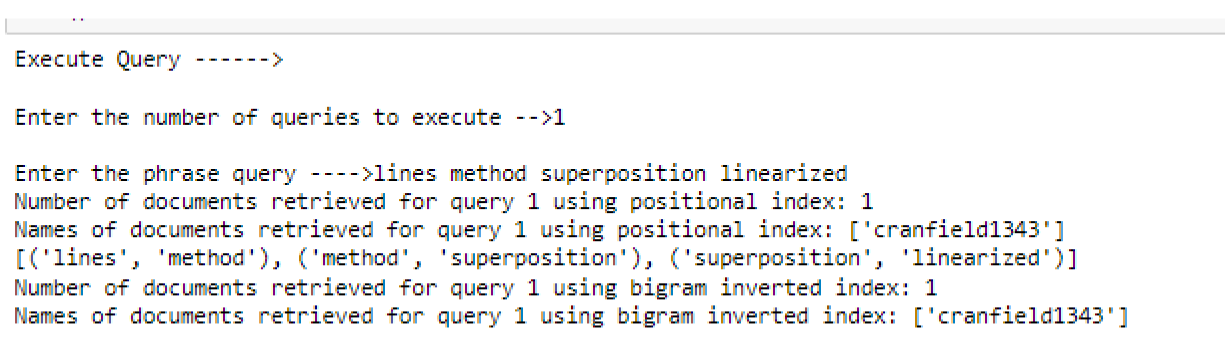
   Methodology:

   - Make a list of all tokens in the corpus
   - For each token, iterate each document and check if the token is present.
   - If present, record its position.

C. Methodology:

   - Positional Index
     - Preprocess the query in order to obtain the tokens
     - For each token, obtain the list from the positional index
     - Do positional intersect of each of the lists
   - Bigram
     - Preprocess the query in order to obtain the bigram
     - For each bigram obtain the list of document Ids.
     - Do AND operation from left to right for the above-obtained bigram list.

Results:



```
Execute Query ------>

Enter the number of queries to execute -->1

Enter the phrase query ---->lines method superposition linearized
Number of documents retrieved for query 1 using positional index: 1
Names of documents retrieved for query 1 using positional index: ['cranfield1343']
[('lines', 'method'), ('method', 'superposition'), ('superposition', 'linearized')]
Number of documents retrieved for query 1 using bigram inverted index: 1
Names of documents retrieved for query 1 using bigram inverted index: ['cranfield1343']
```

Comment:

The size of the bigram index is comparatively larger than the positional indexes.

In the current query both the positional index and the bigram index return the same result for the phrase query.
However bigram indexes can have the issue of false positives, i.e it may return the documents that may not contain the exact phrase , since it also considers overlaps.

For ex for the above query , it could have returned those documents that contained “lines method method superposition superposition linearized” since even this would have fulfilled the condition of the AND operation.