Information Retrieval - CSE508 - Assignment 2

Name: Debnath Kundu (MT22026), Snehal Buldeo (MT22074)

GitHub Link: https://github.com/debnathkundu/CSE508_Winter2023_A2_98

Question 1: Data Preprocessing

(i) *Relevant Text Extraction*

For each file, extract the contents between the <TITLE>...</TITLE> and <TEXT>...</TEXT> tags and concatenate the 2 strings using blank space. Discard the rest of the text and save the string obtained above in the same file.
Perform this on all 1400 files. Print contents of 5 sample files before and after performing the operation.

Assumptions:

1. All the files contain only alphabets, numbers and punctuations.
2. All files contain <TITLE>,</TITLE> ,<TEXT> and </TEXT> tags in it.
3. The file contents are stored in space separated manner

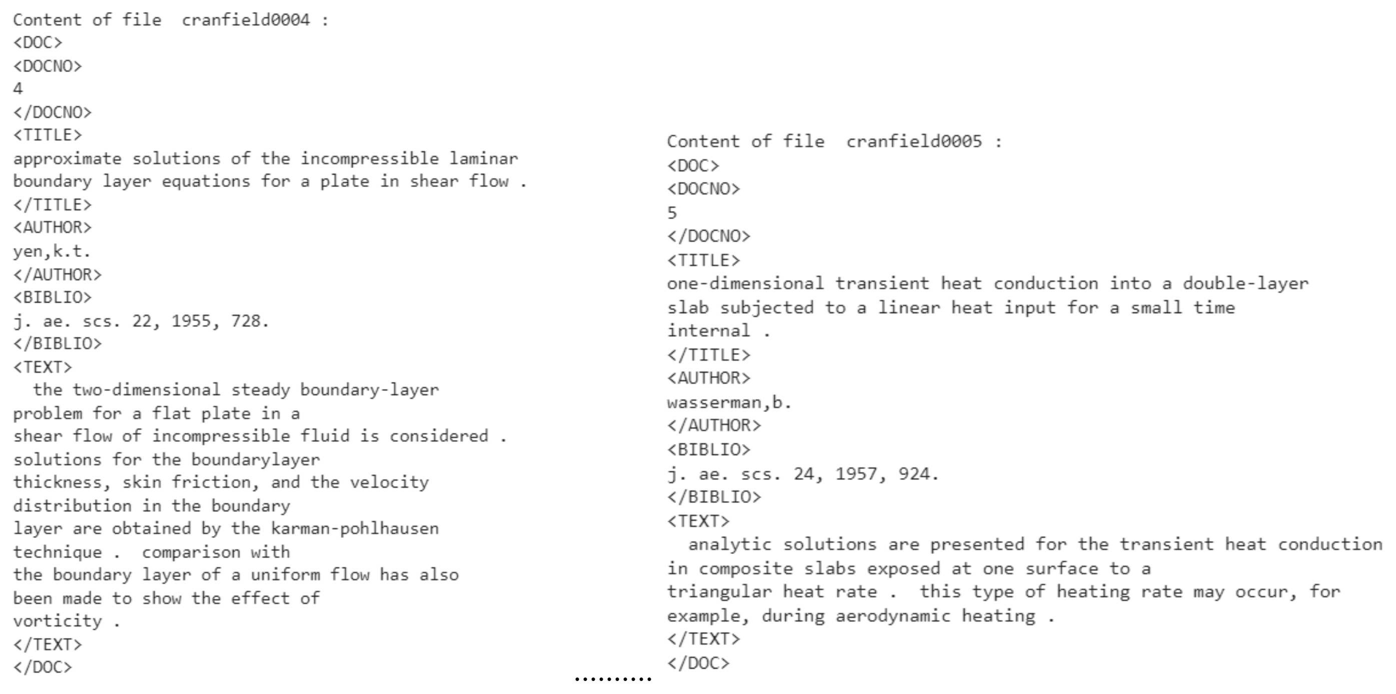
Methodology:

1. For each file in the dataset
   (i) Open the file f
   (ii) Read the file contents in a variable (here, contents)
   (iii) Get the index of <TITLE>,</TITLE> ,<TEXT> and </TEXT> tags
   (iv) extract the data between <TITLE> and </TITLE> tag, and <TEXT> and </TEXT> tag
   (v) save the strings extracted in the file, separated by a blank space

Results:
The contents of the first 5 files before text extraction:

```
Content of file  cranfield0001 :
<DOC>
<DOCNO>
1
</DOCNO>
<TITLE>
experimental investigation of the aerodynamics of a
wing in a slipstream .
</TITLE>
<AUTHOR>
brenckman,m.
</AUTHOR>
<BIBLIO>
j. ae. scs. 25, 1958, 324.
</BIBLIO>
<TEXT>
  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 .
</TEXT>
</DOC>
```

```
Content of file  cranfield0002 :
<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>

Content of file  cranfield0003 :
<DOC>
<DOCNO>
3
</DOCNO>
<TITLE>
the boundary layer in simple shear flow past a flat plate .
</TITLE>
<AUTHOR>
m. b. glauert
</AUTHOR>
<BIBLIO>
department of mathematics, university of manchester, manchester,
england
</BIBLIO>
<TEXT>
the boundary-layer equations are presented for steady
incompressible flow with no pressure gradient .
</TEXT>
</DOC>

Content of file  cranfield0004 :
<DOC>
<DOCNO>
```

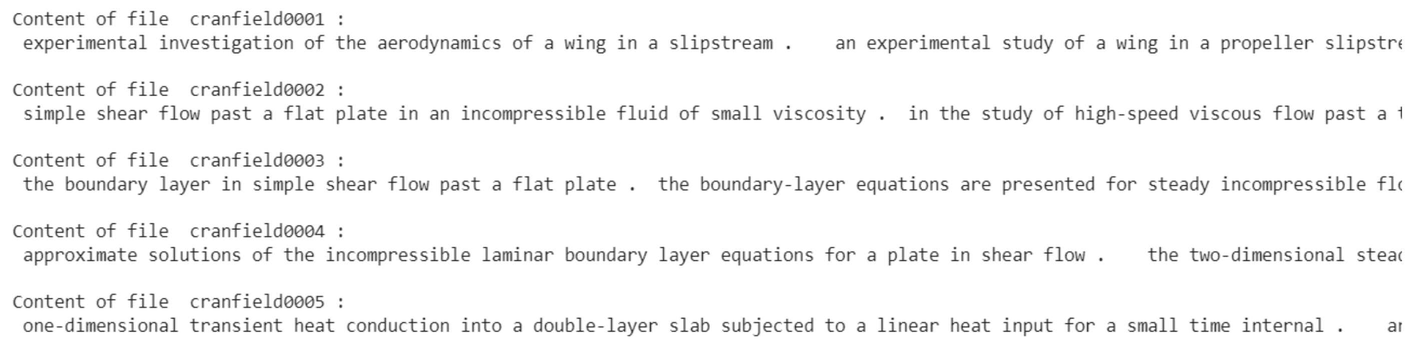


```
Content of file  cranfield0004 :
<DOC>
<DOCNO>
4
</DOCNO>
<TITLE>
approximate solutions of the incompressible laminar
boundary layer equations for a plate in shear flow .
</TITLE>
<AUTHOR>
yen,k.t.
</AUTHOR>
<BIBLIO>
j. ae. scs. 22, 1955, 728.
</BIBLIO>
<TEXT>
  the two-dimensional steady boundary-layer
problem for a flat plate in a
shear flow of incompressible fluid is considered .
solutions for the boundarylayer
thickness, skin friction, and the velocity
distribution in the boundary
layer are obtained by the karman-pohlhausen
technique .  comparison with
the boundary layer of a uniform flow has also
been made to show the effect of
vorticity .
</TEXT>
</DOC>
```

....

```
Content of file  cranfield0005 :
<DOC>
<DOCNO>
5
</DOCNO>
<TITLE>
one-dimensional transient heat conduction into a double-layer
slab subjected to a linear heat input for a small time
internal .
</TITLE>
<AUTHOR>
wasserman,b.
</AUTHOR>
<BIBLIO>
j. ae. scs. 24, 1957, 924.
</BIBLIO>
<TEXT>
  analytic solutions are presented for the transient heat conduction
in composite slabs exposed at one surface to a
triangular heat rate .  this type of heating rate may occur, for
example, during aerodynamic heating .
</TEXT>
</DOC>
```

The contents of first 5 files after text extraction:



```
Content of file  cranfield0001 :
 experimental investigation of the aerodynamics of a wing in a slipstream .    an experimental study of a wing in a propeller slipstre

Content of file  cranfield0002 :
 simple shear flow past a flat plate in an incompressible fluid of small viscosity .  in the study of high-speed viscous flow past a f

Content of file  cranfield0003 :
 the boundary layer in simple shear flow past a flat plate .  the boundary-layer equations are presented for steady incompressible flo

Content of file  cranfield0004 :
 approximate solutions of the incompressible laminar boundary layer equations for a plate in shear flow .    the two-dimensional stea

Content of file  cranfield0005 :
 one-dimensional transient heat conduction into a double-layer slab subjected to a linear heat input for a small time internal .    ar
```

(ii) Preprocessing

Carry out the following preprocessing steps on the dataset obtained above:

1. Lowercase the text

2. Perform tokenization

3. Remove stopwords

4. Remove punctuations

5. Remove blank space tokens

Print contents of 5 sample files before and after performing EACH operation.

Assumptions:

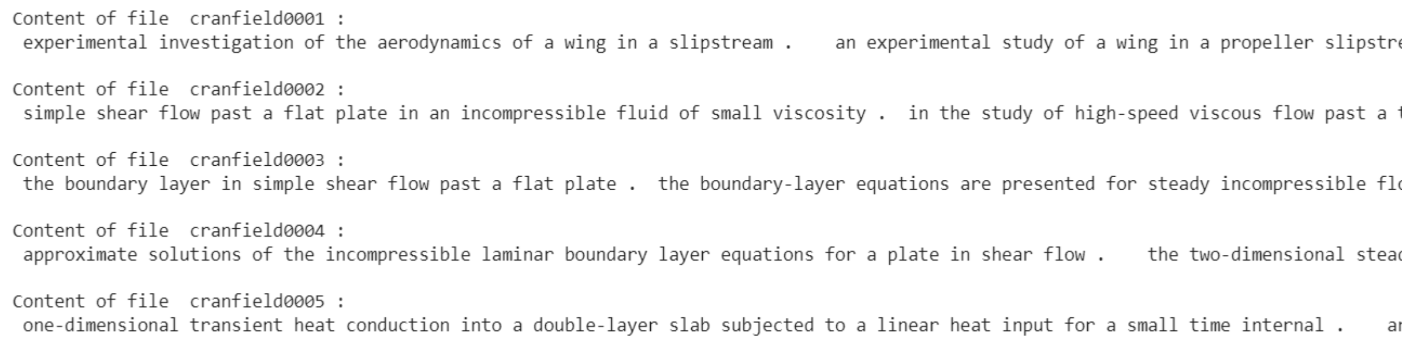
1. All the files contain only alphabets, numbers and punctuations.
2. Stemming or lemmatization is not performed hence words like history and historians are treated as different words.

Methodology:

1. For every file in the dataset
   (i) Open the file
   (ii) Read the file contents in variable(here, contents)
   (iii) Convert the contents to lowercase
   (iv) Perform tokenization on the result of previous step, using nltk.word_tokenize()

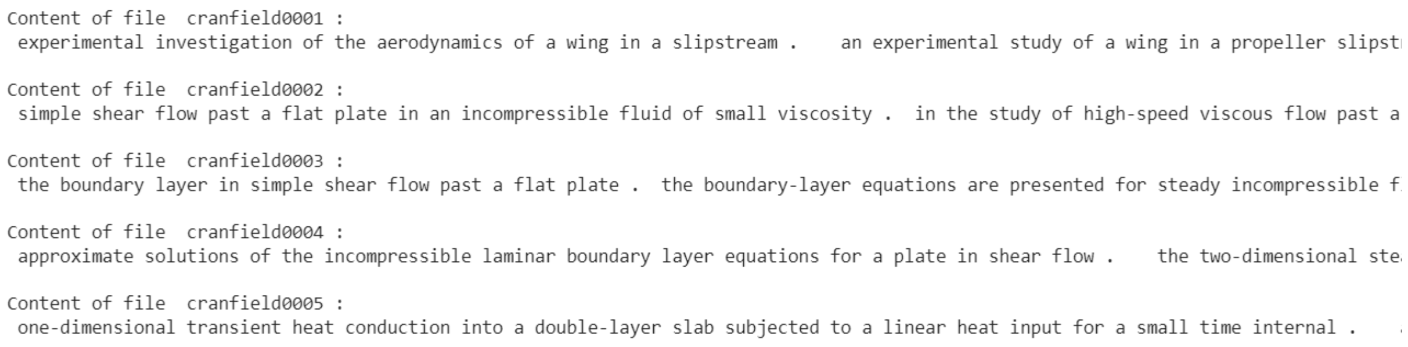
(v) Remove stop words from the list generated in the previous step
(vi) Remove punctuations from the list generated in the previous step
(vii) Remove blank space tokens from the list generated in the previous step

The file contents before performing any preprocessing:



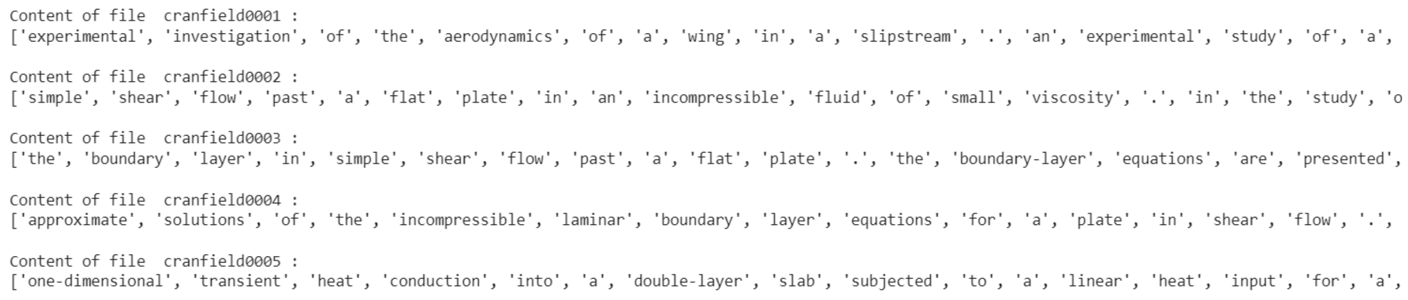
```
Content of file  cranfield0001 :
 experimental investigation of the aerodynamics of a wing in a slipstream .    an experimental study of a wing in a propeller slipstre

Content of file  cranfield0002 :
 simple shear flow past a flat plate in an incompressible fluid of small viscosity .  in the study of high-speed viscous flow past a

Content of file  cranfield0003 :
 the boundary layer in simple shear flow past a flat plate .  the boundary-layer equations are presented for steady incompressible fl

Content of file  cranfield0004 :
 approximate solutions of the incompressible laminar boundary layer equations for a plate in shear flow .    the two-dimensional stea

Content of file  cranfield0005 :
 one-dimensional transient heat conduction into a double-layer slab subjected to a linear heat input for a small time internal .    a
```

The result after converting first 5 file contents to **lowercase**:



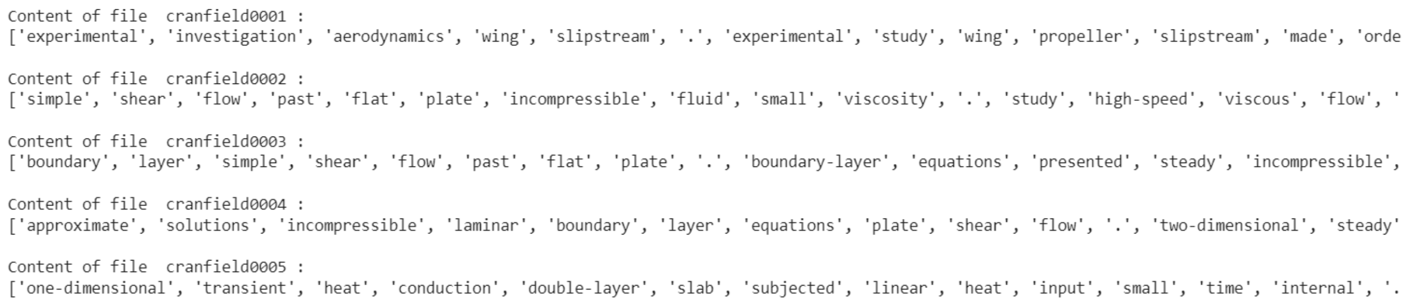
```
Content of file  cranfield0001 :
 experimental investigation of the aerodynamics of a wing in a slipstream .    an experimental study of a wing in a propeller slipst

Content of file  cranfield0002 :
 simple shear flow past a flat plate in an incompressible fluid of small viscosity .  in the study of high-speed viscous flow past a

Content of file  cranfield0003 :
 the boundary layer in simple shear flow past a flat plate .  the boundary-layer equations are presented for steady incompressible f

Content of file  cranfield0004 :
 approximate solutions of the incompressible laminar boundary layer equations for a plate in shear flow .    the two-dimensional ste

Content of file  cranfield0005 :
 one-dimensional transient heat conduction into a double-layer slab subjected to a linear heat input for a small time internal .
```

The result after performing **tokenization** on the first 5 file contents:



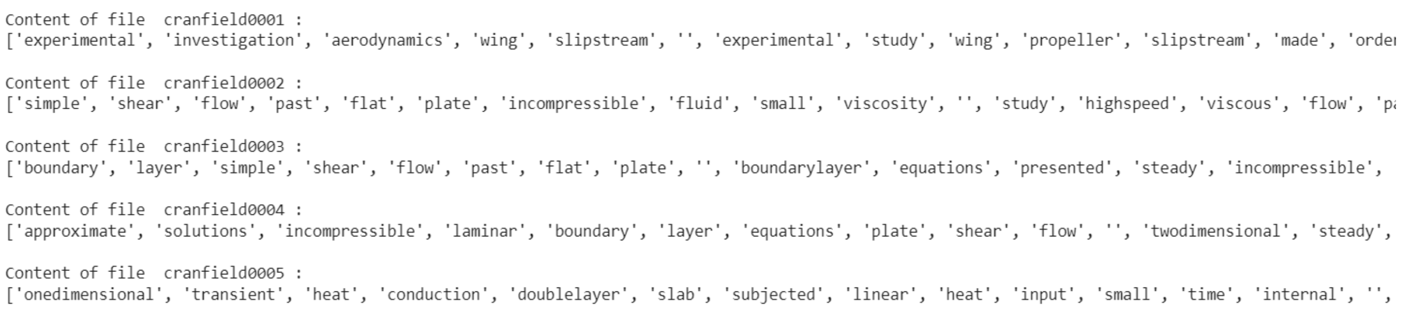
```
Content of file  cranfield0001 :
['experimental', 'investigation', 'of', 'the', 'aerodynamics', 'of', 'a', 'wing', 'in', 'a', 'slipstream', '.', 'an', 'experimental', 'study', 'of', 'a',

Content of file  cranfield0002 :
['simple', 'shear', 'flow', 'past', 'a', 'flat', 'plate', 'in', 'an', 'incompressible', 'fluid', 'of', 'small', 'viscosity', '.', 'in', 'the', 'study', 'o

Content of file  cranfield0003 :
['the', 'boundary', 'layer', 'in', 'simple', 'shear', 'flow', 'past', 'a', 'flat', 'plate', '.', 'the', 'boundary-layer', 'equations', 'are', 'presented',

Content of file  cranfield0004 :
['approximate', 'solutions', 'of', 'the', 'incompressible', 'laminar', 'boundary', 'layer', 'equations', 'for', 'a', 'plate', 'in', 'shear', 'flow', '.',

Content of file  cranfield0005 :
['one-dimensional', 'transient', 'heat', 'conduction', 'into', 'a', 'double-layer', 'slab', 'subjected', 'to', 'a', 'linear', 'heat', 'input', 'for', 'a',
```

The result after **removing the stop words** from the first 5 file contents:



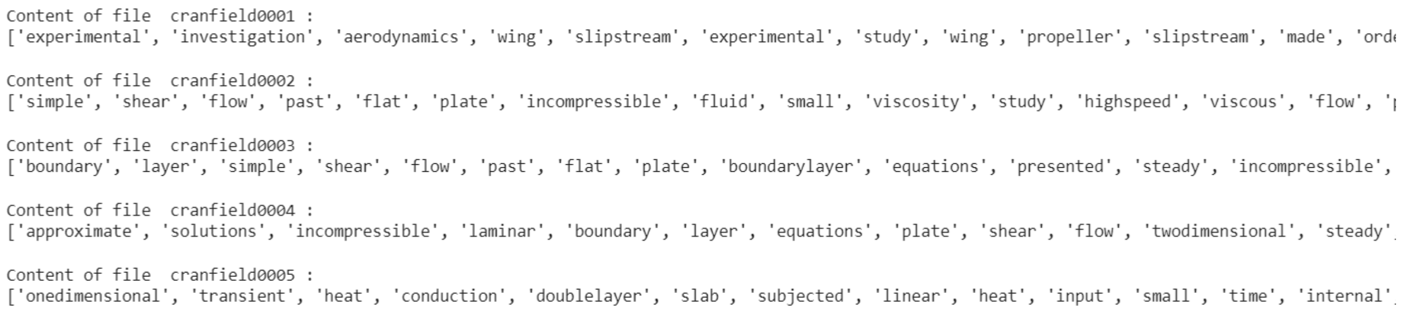
```
Content of file  cranfield0001 :
['experimental', 'investigation', 'aerodynamics', 'wing', 'slipstream', '.', 'experimental', 'study', 'wing', 'propeller', 'slipstream', 'made', 'orde

Content of file  cranfield0002 :
['simple', 'shear', 'flow', 'past', 'flat', 'plate', 'incompressible', 'fluid', 'small', 'viscosity', '.', 'study', 'high-speed', 'viscous', 'flow', '

Content of file  cranfield0003 :
['boundary', 'layer', 'simple', 'shear', 'flow', 'past', 'flat', 'plate', '.', 'boundary-layer', 'equations', 'presented', 'steady', 'incompressible',

Content of file  cranfield0004 :
['approximate', 'solutions', 'incompressible', 'laminar', 'boundary', 'layer', 'equations', 'plate', 'shear', 'flow', '.', 'two-dimensional', 'steady'

Content of file  cranfield0005 :
['one-dimensional', 'transient', 'heat', 'conduction', 'double-layer', 'slab', 'subjected', 'linear', 'heat', 'input', 'small', 'time', 'internal', '.
```

The result after **removing the punctuations** from the first 5 file contents:



```
Content of file  cranfield0001 :
['experimental', 'investigation', 'aerodynamics', 'wing', 'slipstream', '', 'experimental', 'study', 'wing', 'propeller', 'slipstream', 'made', 'orde

Content of file  cranfield0002 :
['simple', 'shear', 'flow', 'past', 'flat', 'plate', 'incompressible', 'fluid', 'small', 'viscosity', '', 'study', 'highspeed', 'viscous', 'flow', 'pa

Content of file  cranfield0003 :
['boundary', 'layer', 'simple', 'shear', 'flow', 'past', 'flat', 'plate', '', 'boundarylayer', 'equations', 'presented', 'steady', 'incompressible',

Content of file  cranfield0004 :
['approximate', 'solutions', 'incompressible', 'laminar', 'boundary', 'layer', 'equations', 'plate', 'shear', 'flow', '', 'twodimensional', 'steady',

Content of file  cranfield0005 :
['onedimensional', 'transient', 'heat', 'conduction', 'doublelayer', 'slab', 'subjected', 'linear', 'heat', 'input', 'small', 'time', 'internal', '',
```

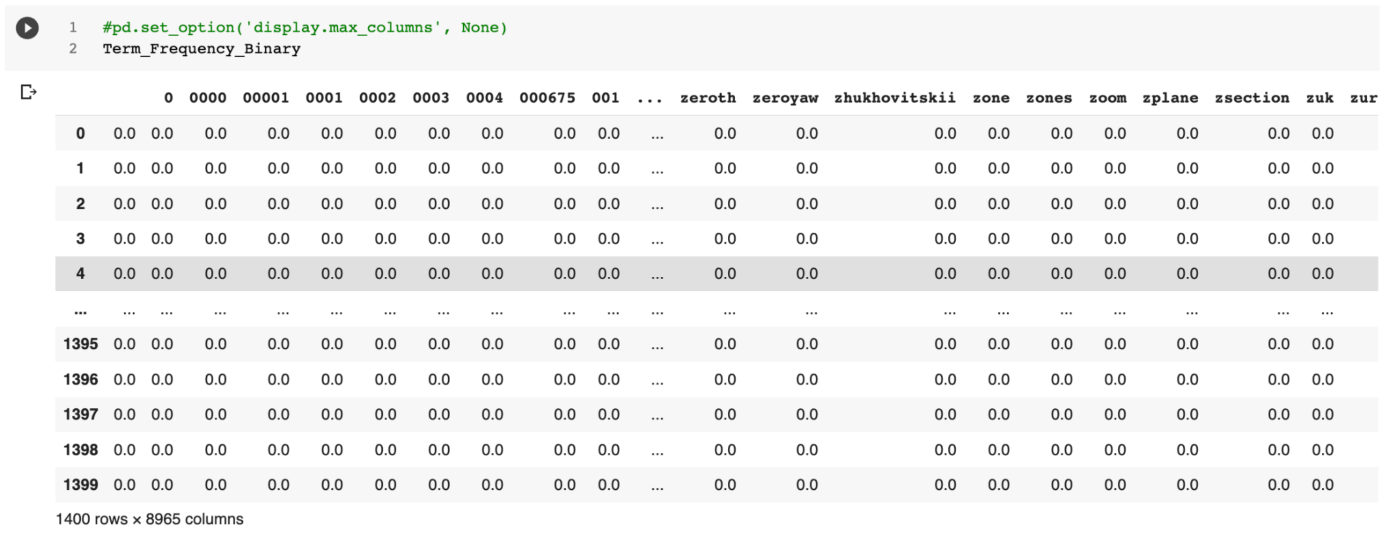
The result after **removing the white spaces** from the first 5 file contents:



```
Content of file  cranfield0001 :
['experimental', 'investigation', 'aerodynamics', 'wing', 'slipstream', 'experimental', 'study', 'wing', 'propeller', 'slipstream', 'made', 'orde

Content of file  cranfield0002 :
['simple', 'shear', 'flow', 'past', 'flat', 'plate', 'incompressible', 'fluid', 'small', 'viscosity', 'study', 'highspeed', 'viscous', 'flow', 'p

Content of file  cranfield0003 :
['boundary', 'layer', 'simple', 'shear', 'flow', 'past', 'flat', 'plate', 'boundarylayer', 'equations', 'presented', 'steady', 'incompressible',

Content of file  cranfield0004 :
['approximate', 'solutions', 'incompressible', 'laminar', 'boundary', 'layer', 'equations', 'plate', 'shear', 'flow', 'twodimensional', 'steady'

Content of file  cranfield0005 :
['onedimensional', 'transient', 'heat', 'conduction', 'doublelayer', 'slab', 'subjected', 'linear', 'heat', 'input', 'small', 'time', 'internal'
```

(ii) TF-IDF Matrix [25 points]

Weighting Scheme	**TF Weight**
Binary	*0,1*

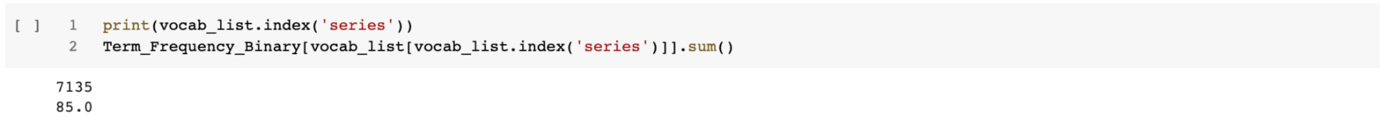


```
#pd.set_option('display.max_columns', None)
Term_Frequency_Binary
```

		0	0000	00001	0001	0002	0003	0004	000675	001	...	zeroth	zeroyaw	zhukhovitskii	zone	zones	zoom	zplane	zsection	zuk	zur
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1395	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1396	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1397	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1398	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1399	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

1400 rows × 8965 columns

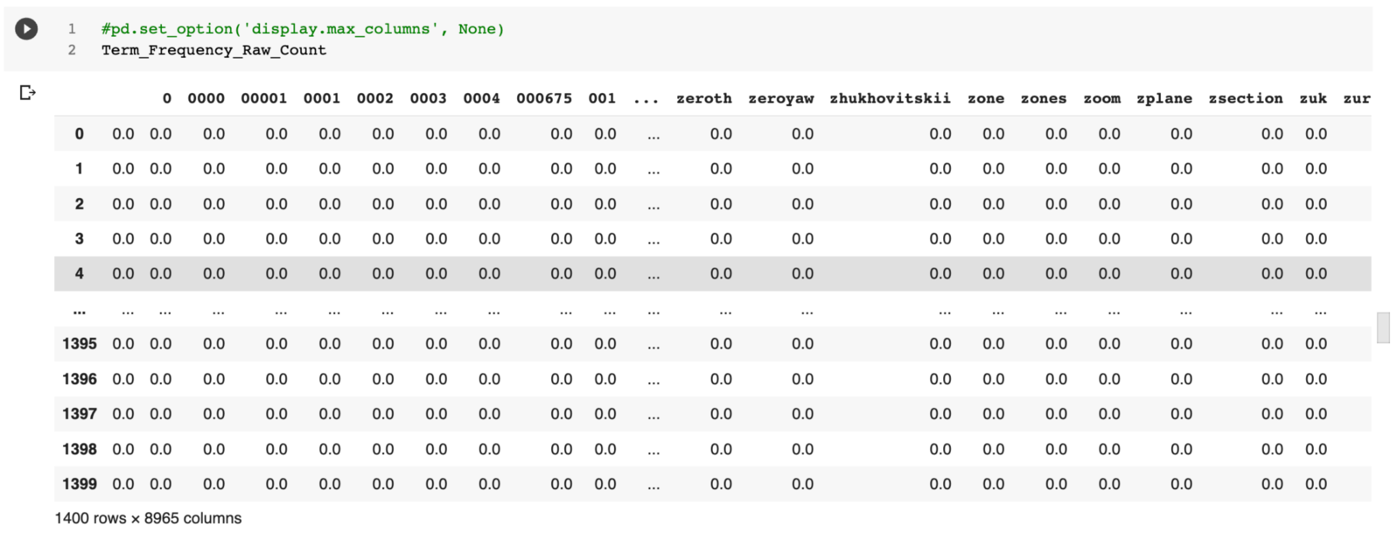
Check the frequency of any term in the binary matrix: Used to calculate iDF



```
[ ]
print(vocab_list.index('series'))
Term_Frequency_Binary[vocab_list[vocab_list.index('series')]].sum()
```

```
7135
85.0
```

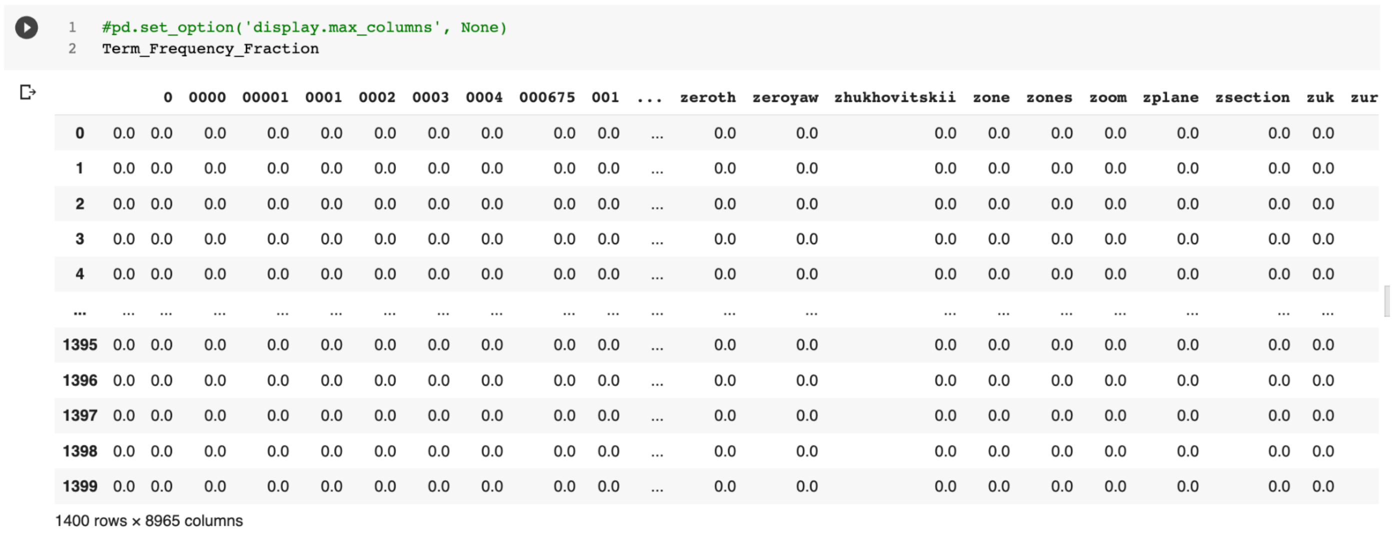
Raw count *$f(t,d)$*



```
#pd.set_option('display.max_columns', None)
Term_Frequency_Raw_Count
```

```
         0  0000  00001  0001  0002  0003  0004  000675  001  ...  zeroth  zeroyaw  zhukhovitskii  zone  zones  zoom  zplane  zsection  zuk  zur
   0   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   1   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   2   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   3   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   4   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
 ...   ...   ...    ...   ...   ...   ...   ...     ...  ...  ...     ...      ...            ...   ...    ...   ...     ...       ...  ...
1395   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1396   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1397   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1398   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1399   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0

1400 rows × 8965 columns
```

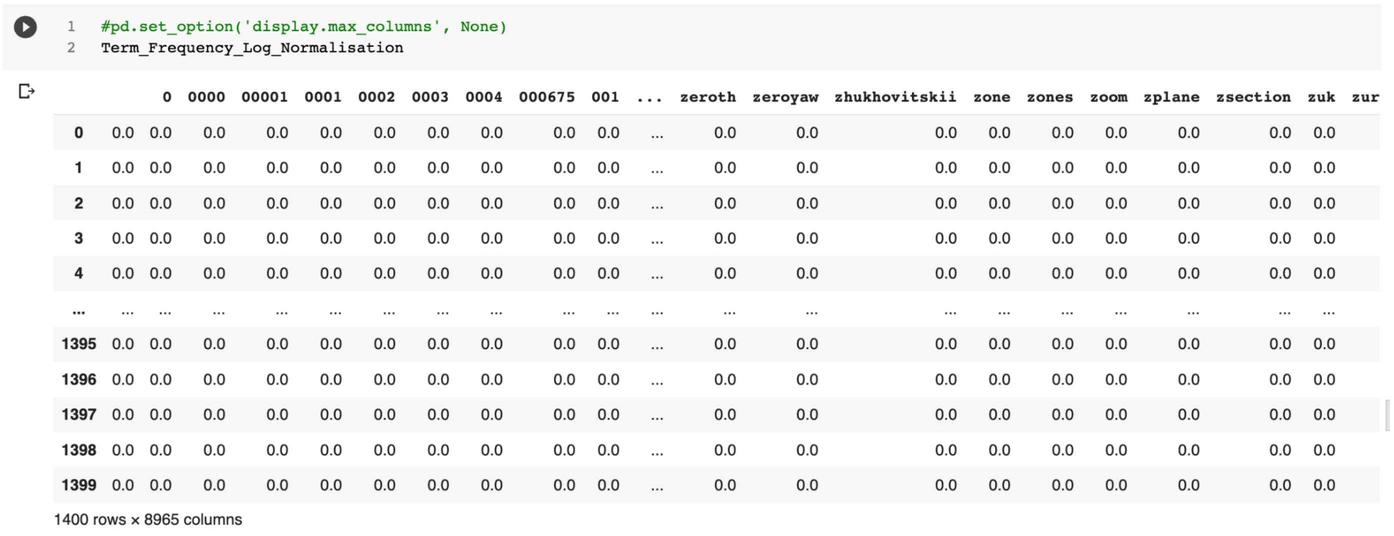
Term frequency *$f(t,d) / \Sigma f(t, d)$*



```
#pd.set_option('display.max_columns', None)
Term_Frequency_Fraction
```

```
         0  0000  00001  0001  0002  0003  0004  000675  001  ...  zeroth  zeroyaw  zhukhovitskii  zone  zones  zoom  zplane  zsection  zuk  zur
   0   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   1   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   2   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   3   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   4   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
 ...   ...   ...    ...   ...   ...   ...   ...     ...  ...  ...     ...      ...            ...   ...    ...   ...     ...       ...  ...
1395   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1396   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1397   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1398   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1399   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0

1400 rows × 8965 columns
```

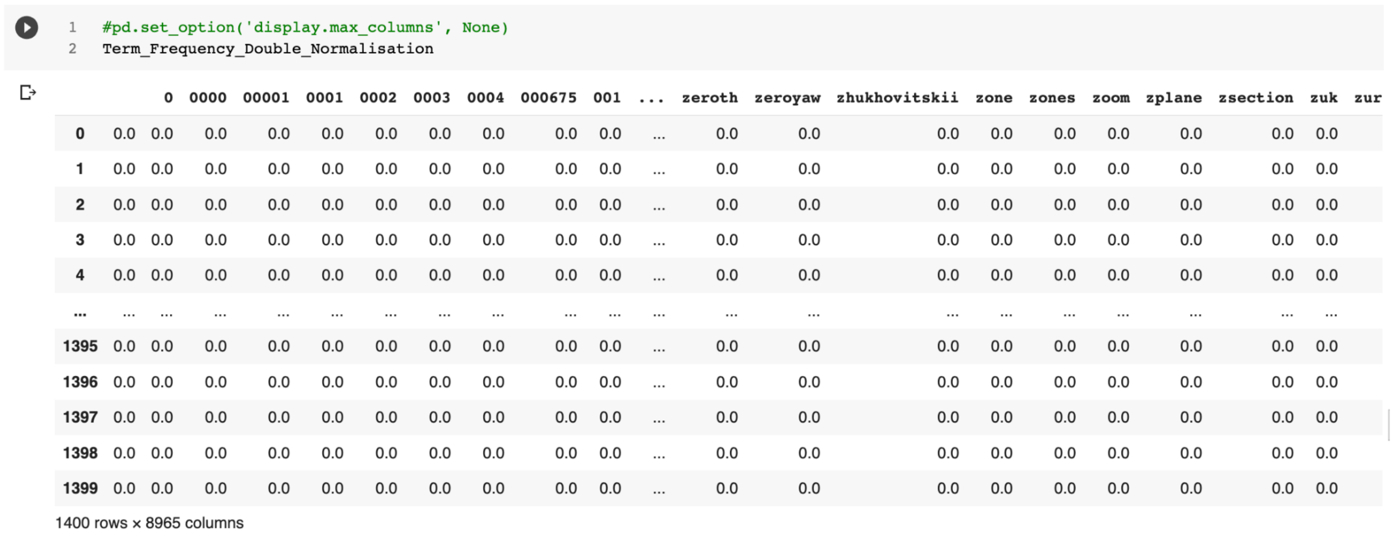
Log normalization *$\log(1+f(t,d))$*



```
#pd.set_option('display.max_columns', None)
Term_Frequency_Log_Normalisation
```

```
         0  0000  00001  0001  0002  0003  0004  000675  001  ...  zeroth  zeroyaw  zhukhovitskii  zone  zones  zoom  zplane  zsection  zuk  zur
   0   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   1   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   2   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   3   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   4   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
 ...   ...   ...    ...   ...   ...   ...   ...     ...  ...  ...     ...      ...            ...   ...    ...   ...     ...       ...  ...
1395   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1396   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1397   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1398   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1399   0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0

1400 rows × 8965 columns
```

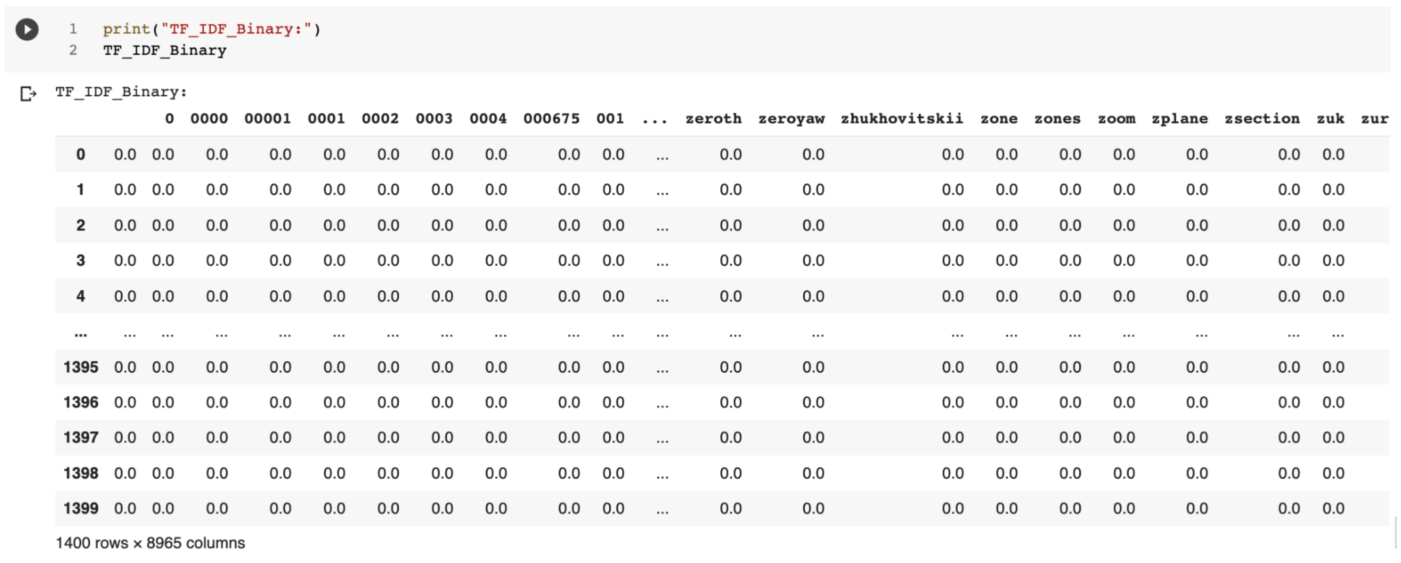
Double normalization   *0.5+0.5*(f(t,d)/ max(f(t‘,d))*



```python
#pd.set_option('display.max_columns', None)
Term_Frequency_Double_Normalisation
```

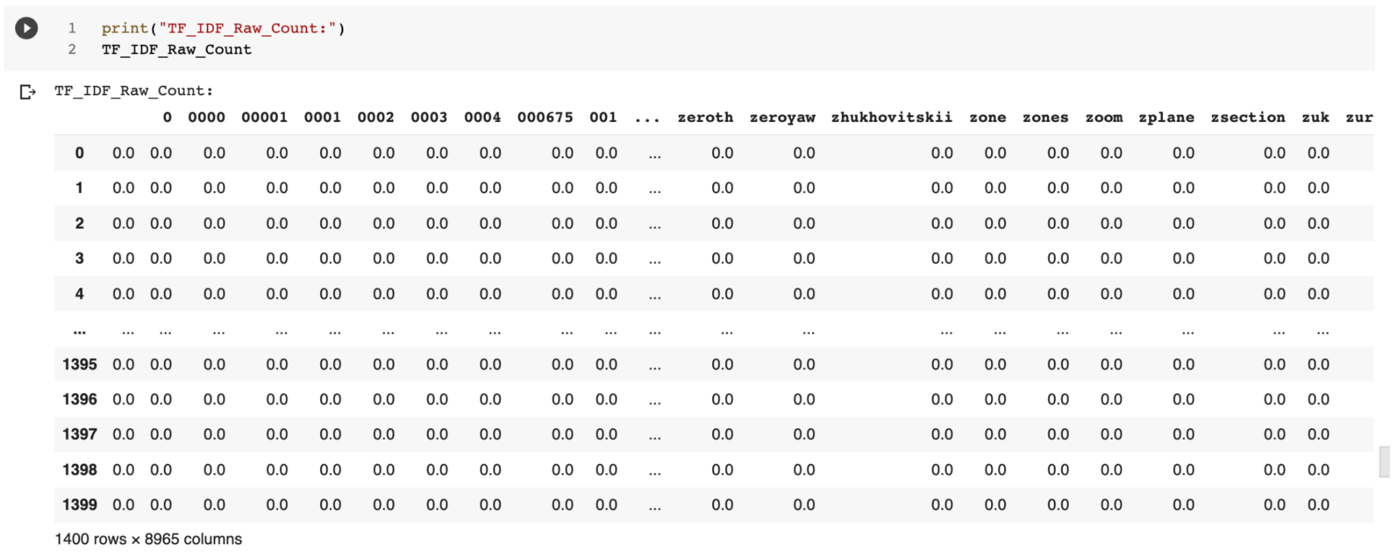
```
          0  0000  00001  0001  0002  0003  0004  000675  001  ...  zeroth  zeroyaw  zhukhovitskii  zone  zones  zoom  zplane  zsection  zuk  zur
   0    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   1    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   2    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   3    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   4    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
 ...    ...   ...    ...   ...   ...   ...   ...     ...  ...  ...  ...     ...      ...            ...   ...    ...   ...     ...       ...  ...
1395    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1396    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1397    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1398    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1399    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0

1400 rows × 8965 columns
```

(iii)Fill in the tf-idf values for each term in the vocabulary in the matrix.



```python
print("TF_IDF_Binary:")
TF_IDF_Binary
```

```
TF_IDF_Binary:
          0  0000  00001  0001  0002  0003  0004  000675  001  ...  zeroth  zeroyaw  zhukhovitskii  zone  zones  zoom  zplane  zsection  zuk  zur
   0    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   1    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   2    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   3    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   4    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
 ...    ...   ...    ...   ...   ...   ...   ...     ...  ...  ...  ...     ...      ...            ...   ...    ...   ...     ...       ...  ...
1395    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1396    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1397    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1398    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1399    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0

1400 rows × 8965 columns
```



```python
print("TF_IDF_Raw_Count:")
TF_IDF_Raw_Count
```

```
TF_IDF_Raw_Count:
          0  0000  00001  0001  0002  0003  0004  000675  001  ...  zeroth  zeroyaw  zhukhovitskii  zone  zones  zoom  zplane  zsection  zuk  zur
   0    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   1    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   2    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   3    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
   4    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
 ...    ...   ...    ...   ...   ...   ...   ...     ...  ...  ...  ...     ...      ...            ...   ...    ...   ...     ...       ...  ...
1395    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1396    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1397    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1398    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0
1399    0.0   0.0    0.0   0.0   0.0   0.0   0.0     0.0  0.0  0.0  ...     0.0      0.0            0.0   0.0    0.0   0.0     0.0       0.0  0.0

1400 rows × 8965 columns
```

```
print("TF_IDF_Fraction:")
TF_IDF_Fraction
```

```
TF_IDF_Fraction:
```

	0	0000	00001	0001	0002	0003	0004	000675	001	...	zeroth	zeroyaw	zhukhovitskii	zone	zones	zoom	zplane	zsection	zuk	zur
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1395	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1396	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1397	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1398	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1399	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

1400 rows × 8965 columns

```
print("TF_IDF_Log_Normalisation:")
TF_IDF_Log_Normalisation
```

```
TF_IDF_Log_Normalisation:
```

	0	0000	00001	0001	0002	0003	0004	000675	001	...	zeroth	zeroyaw	zhukhovitskii	zone	zones	zoom	zplane	zsection	zuk	zur
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1395	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1396	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1397	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1398	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1399	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

1400 rows × 8965 columns

```
[ ]
print("TF_IDF_Double_Normalisation:")
TF_IDF_Double_Normalisation
```

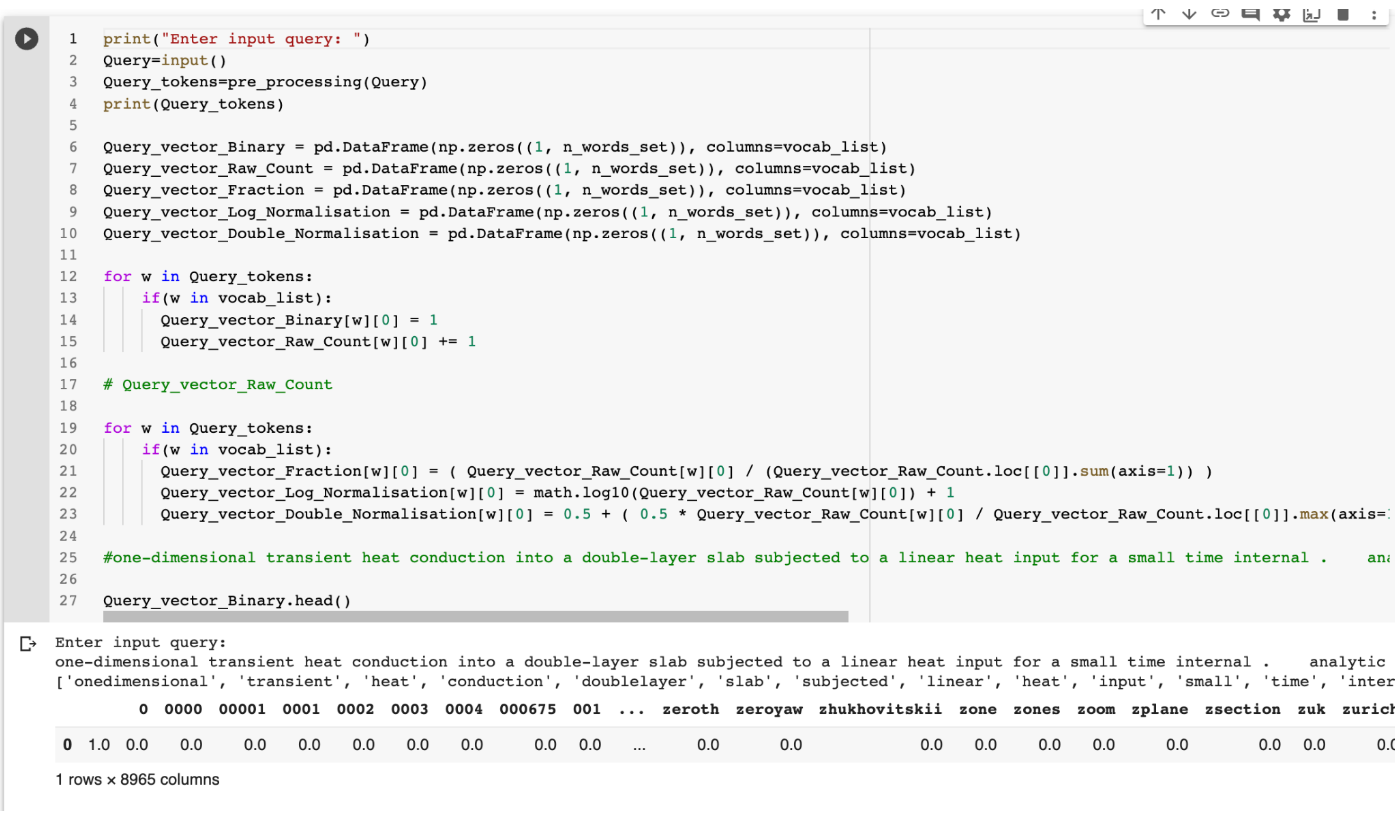
```
TF_IDF_Double_Normalisation:
```

	0	0000	00001	0001	0002	0003	0004	000675	001	...	zeroth	zeroyaw	zhukhovitskii	zone	zones	zoom	zplane	zsection	zuk	zur
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1395	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1396	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1397	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1398	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1399	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

1400 rows × 8965 columns

4. Construct the query vector of size vocab.

For the input query, a separate query vector is created for respective weighting schemes. Below attached are the contents of the Query_vector_Binary.



```python
print("Enter input query: ")
Query=input()
Query_tokens=pre_processing(Query)
print(Query_tokens)

Query_vector_Binary = pd.DataFrame(np.zeros((1, n_words_set)), columns=vocab_list)
Query_vector_Raw_Count = pd.DataFrame(np.zeros((1, n_words_set)), columns=vocab_list)
Query_vector_Fraction = pd.DataFrame(np.zeros((1, n_words_set)), columns=vocab_list)
Query_vector_Log_Normalisation = pd.DataFrame(np.zeros((1, n_words_set)), columns=vocab_list)
Query_vector_Double_Normalisation = pd.DataFrame(np.zeros((1, n_words_set)), columns=vocab_list)

for w in Query_tokens:
    if(w in vocab_list):
        Query_vector_Binary[w][0] = 1
        Query_vector_Raw_Count[w][0] += 1

# Query_vector_Raw_Count

for w in Query_tokens:
    if(w in vocab_list):
        Query_vector_Fraction[w][0] = ( Query_vector_Raw_Count[w][0] / (Query_vector_Raw_Count.loc[[0]].sum(axis=1)) )
        Query_vector_Log_Normalisation[w][0] = math.log10(Query_vector_Raw_Count[w][0]) + 1
        Query_vector_Double_Normalisation[w][0] = 0.5 + ( 0.5 * Query_vector_Raw_Count[w][0] / Query_vector_Raw_Count.loc[[0]].max(axis=

#one-dimensional transient heat conduction into a double-layer slab subjected to a linear heat input for a small time internal .    ana

Query_vector_Binary.head()
```

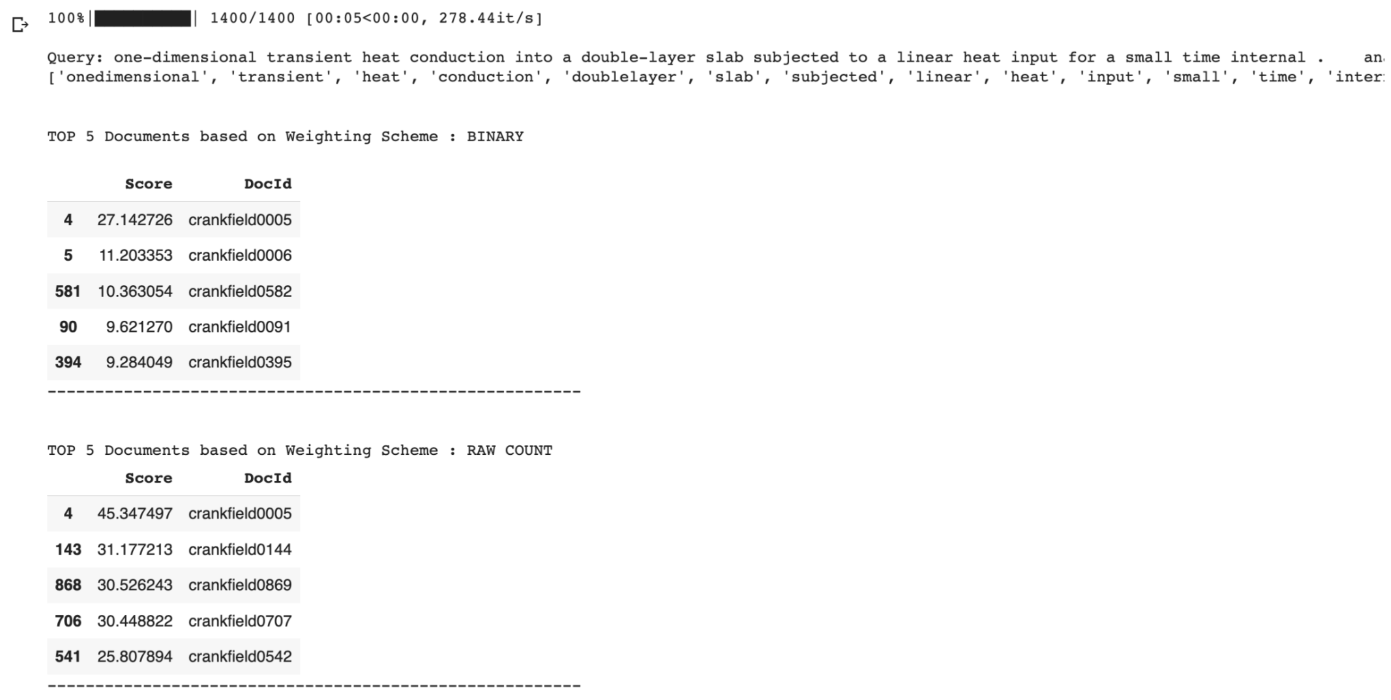
```
Enter input query:
one-dimensional transient heat conduction into a double-layer slab subjected to a linear heat input for a small time internal .    analytic
['onedimensional', 'transient', 'heat', 'conduction', 'doublelayer', 'slab', 'subjected', 'linear', 'heat', 'input', 'small', 'time', 'inter
```

	0	0000	00001	0001	0002	0003	0004	000675	001	...	zeroth	zeroyaw	zhukhovitskii	zone	zones	zoom	zplane	zsection	zuk	zurich
0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

1 rows × 8965 columns

(v) Compute the TF-IDF score for the query using the TF-IDF matrix. Report the top 5 relevant documents based on the score.

The TF-IDF Scores for respective weighting schemes are displayed in the data frame itself.



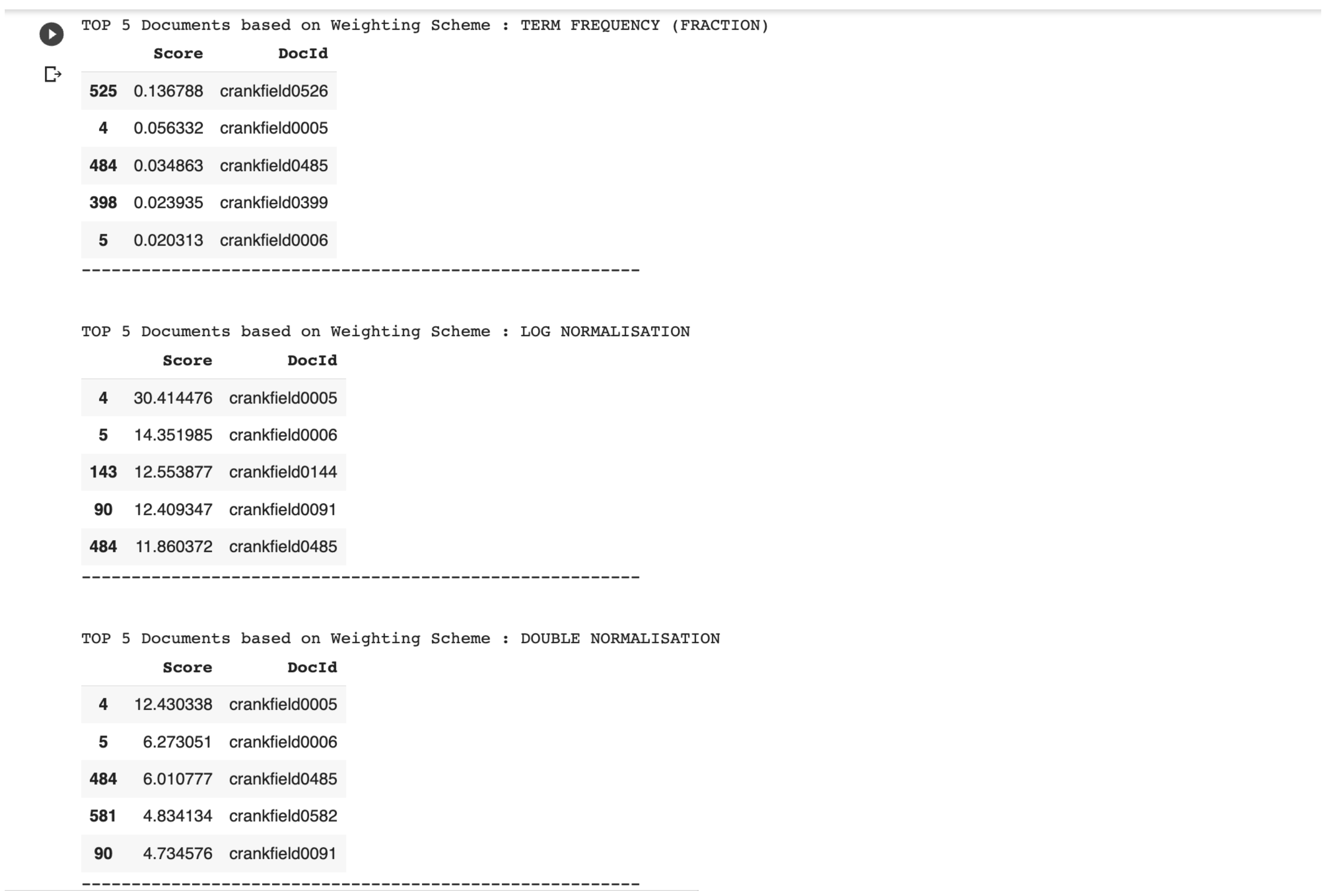
```
100%|██████████| 1400/1400 [00:05<00:00, 278.44it/s]

Query: one-dimensional transient heat conduction into a double-layer slab subjected to a linear heat input for a small time internal .    an
['onedimensional', 'transient', 'heat', 'conduction', 'doublelayer', 'slab', 'subjected', 'linear', 'heat', 'input', 'small', 'time', 'inter


TOP 5 Documents based on Weighting Scheme : BINARY
```

	Score	DocId
4	27.142726	crankfield0005
5	11.203353	crankfield0006
581	10.363054	crankfield0582
90	9.621270	crankfield0091
394	9.284049	crankfield0395

```
--------------------------------------------------------

TOP 5 Documents based on Weighting Scheme : RAW COUNT
```

	Score	DocId
4	45.347497	crankfield0005
143	31.177213	crankfield0144
868	30.526243	crankfield0869
706	30.448822	crankfield0707
541	25.807894	crankfield0542

```
--------------------------------------------------------
```



```
TOP 5 Documents based on Weighting Scheme : TERM FREQUENCY (FRACTION)
```

	Score	DocId
525	0.136788	crankfield0526
4	0.056332	crankfield0005
484	0.034863	crankfield0485
398	0.023935	crankfield0399
5	0.020313	crankfield0006

```
--------------------------------------------------------


TOP 5 Documents based on Weighting Scheme : LOG NORMALISATION
```

	Score	DocId
4	30.414476	crankfield0005
5	14.351985	crankfield0006
143	12.553877	crankfield0144
90	12.409347	crankfield0091
484	11.860372	crankfield0485

```
--------------------------------------------------------


TOP 5 Documents based on Weighting Scheme : DOUBLE NORMALISATION
```

	Score	DocId
4	12.430338	crankfield0005
5	6.273051	crankfield0006
484	6.010777	crankfield0485
581	4.834134	crankfield0582
90	4.734576	crankfield0091

```
--------------------------------------------------------
```

(vi) Use all 5 weighting schemes for term frequency calculation and report the TF-IDF score and results for each scheme separately.

The TF-IDF Scores for respective weighting schemes are displayed in the data frame itself. Refer above screenshots.

(vii) Pros and Cons of using each weighting scheme to determine the relevance of documents

Binary (0,1):

Pros:

1. Simple to implement.
2. Efficient in computation.

Cons:

1. Frequency of occurrence of a term in a document is lost, hence not a robust weighting scheme. It is not recommended for tasks where the frequency of occurrence of a term in a document is important.
2. The method also does not capture the semantic correlation between different terms in a document but merely registers their presence or absence.
3. No information about rare terms.
4. It is rather difficult to model sparse representations.

Raw count f(t,d):

Pros:

1. Simple to implement.
2. Frequency of occurrence of a term in a document is stored. It forms a basis for a lot of other weighting schemes where the frequency of occurrence a term in a document is important.
3. **"Bag-of-words"** model is a tool for feature generation. It is possible to calculate several different measures that can be used to characterize the text.

Cons:

1. No information on the positional index. The method also does not capture the semantic correlation between different terms in a document.
2. No information about rare terms.
3. Relevance of a document does not increase proportionally with term count.
4. Raw Count isn't the best representation of the text because common words such as "the", "a", and "to" are almost always the terms with the highest frequency in the text.
5. It is rather difficult to model sparse representations.

Term frequency f(t,d) / Σ(t, d):

Pros:

1. Term weighting indicates how important each individual word is to the document and within the document collection.
2. Can be used further to assign weights to different sections of the document such as Title/Body.
3. It represents the relative frequency of term t within document d,

Cons:

1. Still no information of positional index. The method does not capture the semantic correlation between different terms in a document.
2. No information about rare terms.
3. Relevance of a document does not increase proportionally with term count.
4. Raw Count isn't the best representation of the text because common words such as "the", "a", and "to" are almost always the terms with the highest frequency in the text.

Log normalization log(1+f(t,d)):

Pros:

1. Dampen the effect of the term that has a high frequency.
2. Brings non-linearity in the scoring process.
3. We also add 1 to the log(tf) because when tf is equal to 1, the log(1) is zero. By adding one, we distinguish between tf=0 and tf=1.

Cons:

1. For frequent terms, we want high +ve weights for the words, but lower weights than the rare terms.
2. The method also does not capture the semantic correlation between different terms in a document.

Double normalization 0.5+0.5*(f(t,d)/ max(f(t',d)):

Pros:

1. Also called **"Augmented frequency**", is used to prevent a bias towards longer documents, e.g. raw frequency divided by the raw frequency of the most frequently occurring term in the document.

2. Brings non-linearity to the scoring process.
3. It is therefore critical to balance between over-weighting and under-weighting.

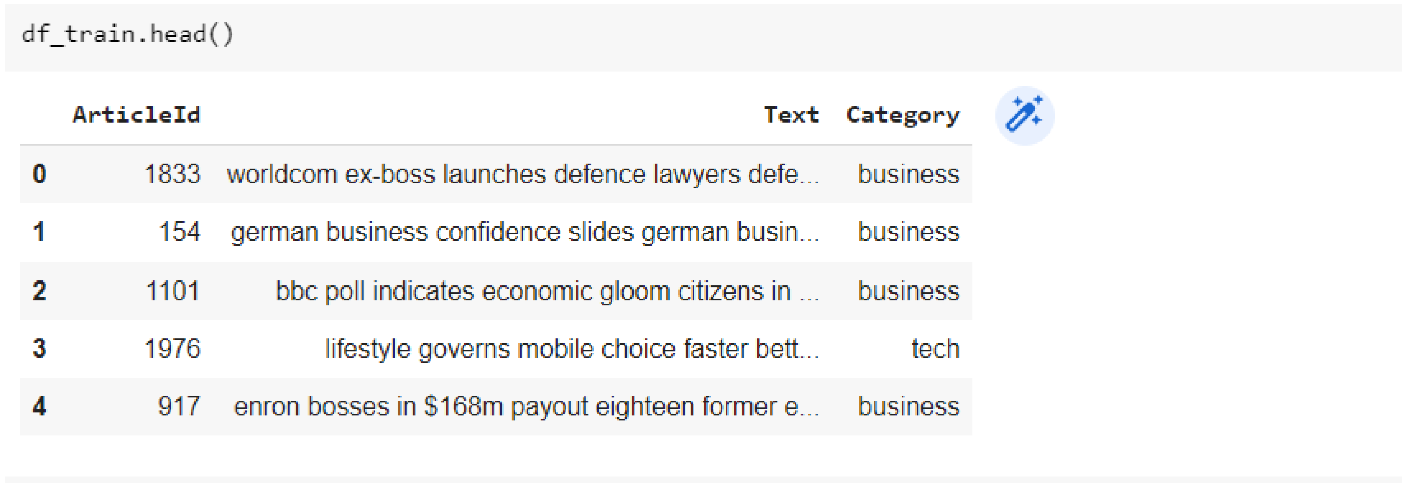
Cons:

1. Sublinear scaling and bias terms shrink the ratios between weights of different terms. If the scaling functions scale down the weights too much or the bias term is too large, ratios between term weights become too small. Hence under-weighting occurs.
2. Regularization techniques may be required to prevent under/overweighting, thus, adding to overhead.

Question 2: Naive Bayes Classifier with TF-ICF[40 marks]

1. Preprocessing (2 points)

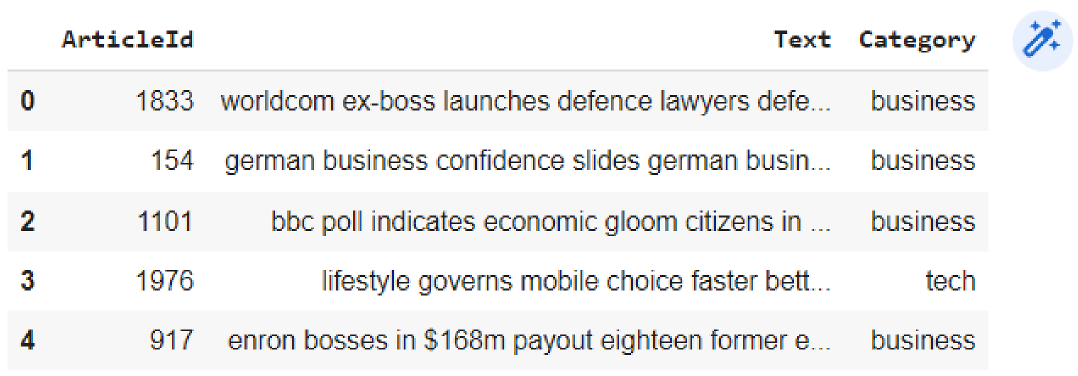
The file contents before performing any preprocessing:



```
df_train.head()
```

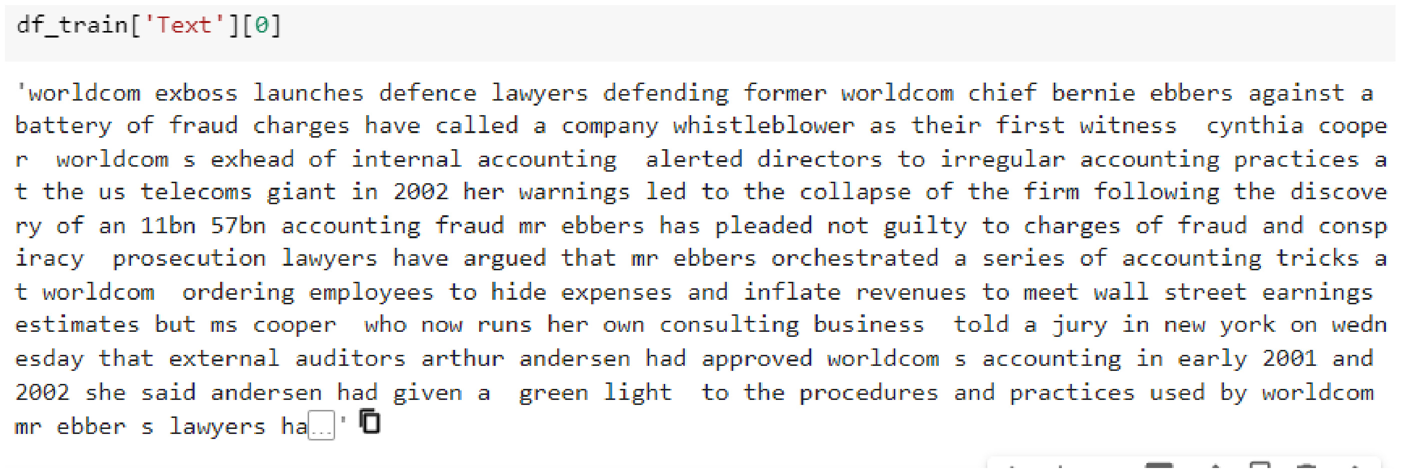
	ArticleId	Text	Category
0	1833	worldcom ex-boss launches defence lawyers defe...	business
1	154	german business confidence slides german busin...	business
2	1101	bbc poll indicates economic gloom citizens in ...	business
3	1976	lifestyle governs mobile choice faster bett...	tech
4	917	enron bosses in $168m payout eighteen former e...	business

The result after converting the file contents to **lowercase**:



	ArticleId	Text	Category
0	1833	worldcom ex-boss launches defence lawyers defe...	business
1	154	german business confidence slides german busin...	business
2	1101	bbc poll indicates economic gloom citizens in ...	business
3	1976	lifestyle governs mobile choice faster bett...	tech
4	917	enron bosses in $168m payout eighteen former e...	business

The result after **removing punctuations** from the file contents:



```
df_train['Text'][0]
```

```
'worldcom exboss launches defence lawyers defending former worldcom chief bernie ebbers against a
battery of fraud charges have called a company whistleblower as their first witness  cynthia coope
r  worldcom s exhead of internal accounting  alerted directors to irregular accounting practices a
t the us telecoms giant in 2002 her warnings led to the collapse of the firm following the discove
ry of an 11bn 57bn accounting fraud mr ebbers has pleaded not guilty to charges of fraud and consp
iracy  prosecution lawyers have argued that mr ebbers orchestrated a series of accounting tricks a
t worldcom  ordering employees to hide expenses and inflate revenues to meet wall street earnings
estimates but ms cooper  who now runs her own consulting business  told a jury in new york on wedn
esday that external auditors arthur andersen had approved worldcom s accounting in early 2001 and
2002 she said andersen had given a  green light  to the procedures and practices used by worldcom
mr ebber s lawyers ha...'
```

The result after **removing the stop words** from the first 5 file contents:

```
df_train['Text'][0]
```

```
'worldcom exboss launches defence lawyers defending former worldcom chief bernie ebbers battery fr
aud charges called company whistleblower first witness cynthia cooper worldcom exhead internal acc
ounting alerted directors irregular accounting practices us telecoms giant 2002 warnings led colla
pse firm following discovery 11bn 57bn accounting fraud mr ebbers pleaded guilty charges fraud con
spiracy prosecution lawyers argued mr ebbers orchestrated series accounting tricks worldcom orderi
ng employees hide expenses inflate revenues meet wall street earnings estimates ms cooper runs con
sulting business told jury new york wednesday external auditors arthur andersen approved worldcom
accounting early 2001 2002 said andersen given green light procedures practices used worldcom mr e
bber lawyers said unaware fraud arguing auditors alert problems ms cooper also said shareholder me
etings mr ebbers often passed technical questions company finance chief giving brief answers prose
cution star witness f...'
```

The result after **performing lemmatization** on the file contents:

```
df_train['Text'][0]
```

```
'worldcom exboss launch defence lawyer defending former worldcom chief bernie ebbers battery fraud
charge called company whistleblower first witness cynthia cooper worldcom exhead internal accounti
ng alerted director irregular accounting practice u telecom giant 2002 warning led collapse firm f
ollowing discovery 11bn 57bn accounting fraud mr ebbers pleaded guilty charge fraud conspiracy pro
secution lawyer argued mr ebbers orchestrated series accounting trick worldcom ordering employee h
ide expense inflate revenue meet wall street earnings estimate m cooper run consulting business to
ld jury new york wednesday external auditor arthur andersen approved worldcom accounting early 200
1 2002 said andersen given green light procedure practice used worldcom mr ebber lawyer said unawa
re fraud arguing auditor alert problem m cooper also said shareholder meeting mr ebbers often pass
ed technical question company finance chief giving brief answer prosecution star witness former wo
rldcom financial chi...'
```

2. Splitting the dataset (5 points)

Result after performing train-test split of 70:30

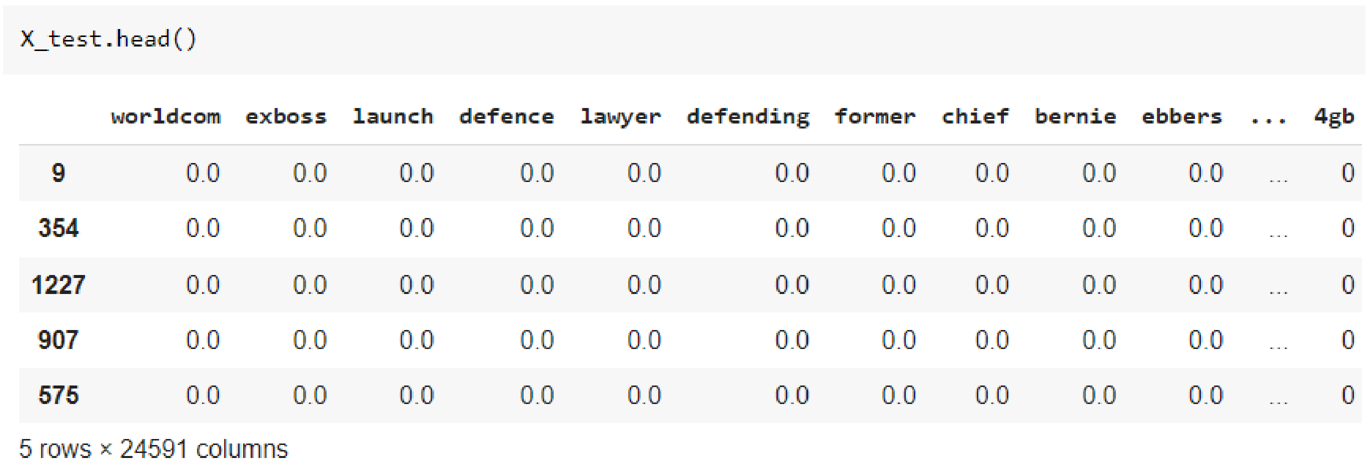
X_train :

```
X_train.head()
```

	worldcom	exboss	launch	defence	lawyer	defending	former	chief	bernie	ebbers	...	4gb
240	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0
1305	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0
1042	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0
1426	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0
1364	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0

5 rows × 24591 columns

X_test :

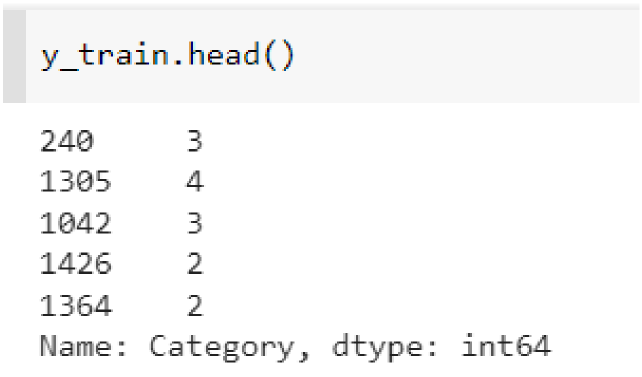


```
X_test.head()
```

	worldcom	exboss	launch	defence	lawyer	defending	former	chief	bernie	ebbers	...	4gb
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0
354	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0
1227	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0
907	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0
575	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0

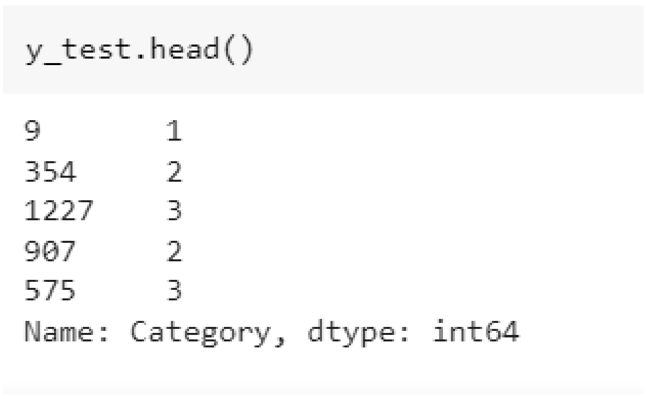
5 rows × 24591 columns

y_train and y_test :



```
y_train.head()
```

```
240     3
1305    4
1042    3
1426    2
1364    2
Name: Category, dtype: int64
```

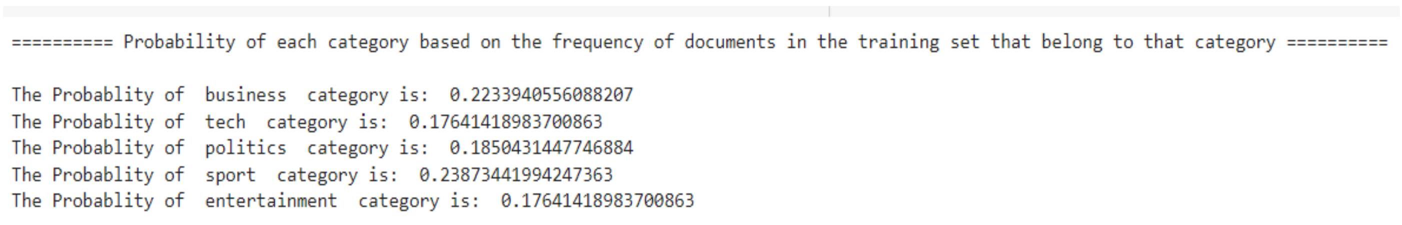


```
y_test.head()
```

```
9       1
354     2
1227    3
907     2
575     3
Name: Category, dtype: int64
```

3. Training the Naive Bayes classifier with TF-ICF (10 points)

Calculate the probability of each category based on the frequency of documents in the training set that belong to that category :

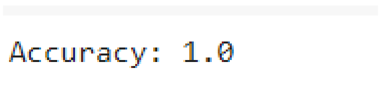


```
========== Probability of each category based on the frequency of documents in the training set that belong to that category ==========

The Probablity of  business  category is:  0.2233940556088207
The Probablity of  tech  category is:  0.17641418983700863
The Probablity of  politics  category is:  0.1850431447746884
The Probablity of  sport  category is:  0.23873441994247363
The Probablity of  entertainment  category is:  0.17641418983700863
```

4. Testing the Naive Bayes classifier with TF-ICF (10 points)

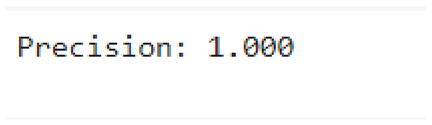
Results :
After testing the data on the test set the following values were obtained.
The train test split for this purpose was 70:30.

Accuracy obtained:



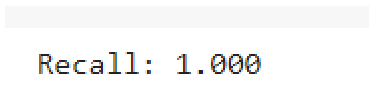
```
Accuracy: 1.0
```

Precision obtained :



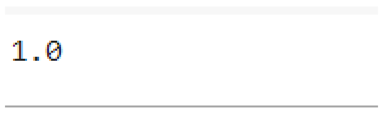
```
Precision: 1.000
```

Recall value :



```
Recall: 1.000
```

F1 score obtained was:

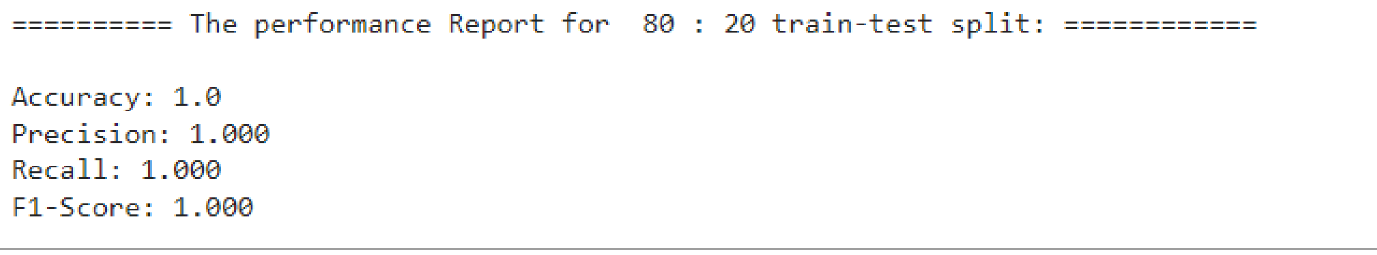


```
1.0
```

5. Improving the classifier (10 points)

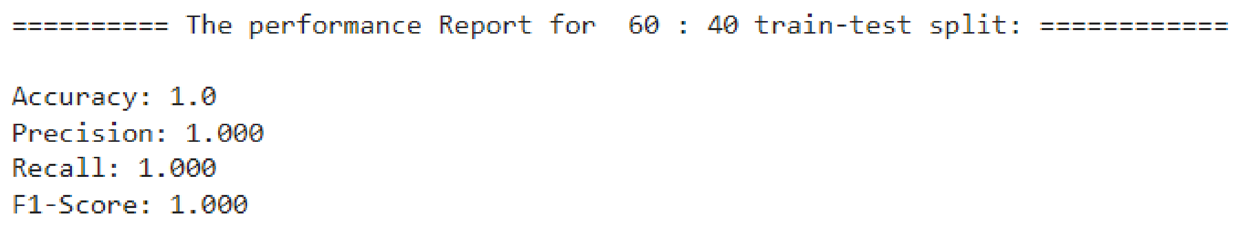
Performance on different **train-test splits** was observed (80:20,40:60,50:50)

1. Performance of TF-ICF on 80:20 train-test split



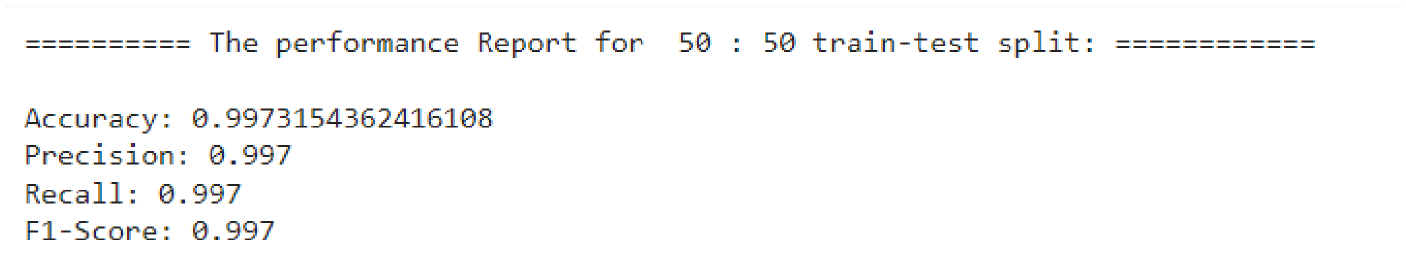
```
========== The performance Report for  80 : 20 train-test split: ============

Accuracy: 1.0
Precision: 1.000
Recall: 1.000
F1-Score: 1.000
```

2. Performance of TF-ICF on 60:40 train-test split



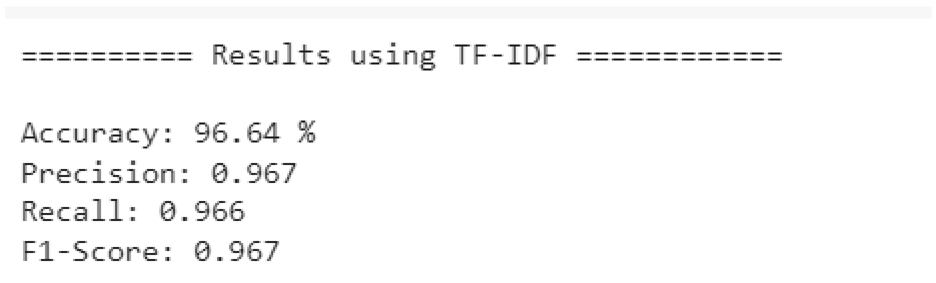
```
========== The performance Report for  60 : 40 train-test split: ============

Accuracy: 1.0
Precision: 1.000
Recall: 1.000
F1-Score: 1.000
```

3. Performance of TF-ICF on 50:50 train-test split



```
========== The performance Report for  50 : 50 train-test split: ============

Accuracy: 0.9973154362416108
Precision: 0.997
Recall: 0.997
F1-Score: 0.997
```

Performance **using TF-IDF vectorizer**



```
========== Results using TF-IDF ============

Accuracy: 96.64 %
Precision: 0.967
Recall: 0.966
F1-Score: 0.967
```

6. Conclusion (3 points)

The performance of TF-ICF was 100% and TF-IDF was 96.6%.
The performance on different train-test splits were almost similar as mentioned in the above sections.
Multinomial Naive Bayes algorithm was used for values generated from TF-IDF vectorizer due to its ability to handle feature counts that are zero (0).
Gaussian Naive Bayes and Multinomial Naive Bayes performed similar on the values from TF-ICF vectorizer.

Question 3: Ranked-Information Retrieval and Evaluation [20 points]

(i) Create a file that rearranges the query-URL pairs in order of the maximum DCG (discounted cumulative gain).

Methodology :

1. Extract relevance judgment column from the dataset with qid = 4
2. Sort the column in decreasing order
3. Store the values in a text file

Results :

The number of such files that can be created

```
The total number of files with maximum DCG are:
 3371779216260740779631533494051750829989294944163825523424606832661840337744948047866557044988313600000000000000000000000
```

The **maximum DCG** is achieved when the relevance scores are sorted in decreasing order. i.e the document with the highest relevance score is ranked first.

Assumptions :

1. The relevance judgment values provided are correct
2. The base of the logarithm is taken as 2.

Methodology :

1. Extract relevance judgment column from the dataset with qid = 4
2. Sort the column in decreasing order
3. Calculate DCG on the sorted judgment values, using the formula below:

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)} = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2(i+1)}$$

4. NDCG for all query-url pairs with qid=4 is calculated using the formula below:

$$\mathrm{nDCG_p} = \frac{DCG_p}{IDCG_p},$$

Where IDCGp is the max_DCG/ DCG (ground truth) .

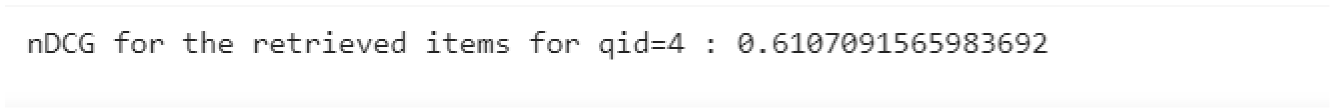
Results :

The maximum DCG over the entire dataset where qid =4:

```
#calculate the max_dcg

max_dcg_qid4 = dcg(ground_truth_qid4)

print("The maximum dcg for all the query-url pairs with qid = 4: ",max_dcg_qid4)
```

```
The maximum dcg for all the query-url pairs with qid = 4:  20.989750804831445
```

nDCG for all query-url pairs with qid=4 :



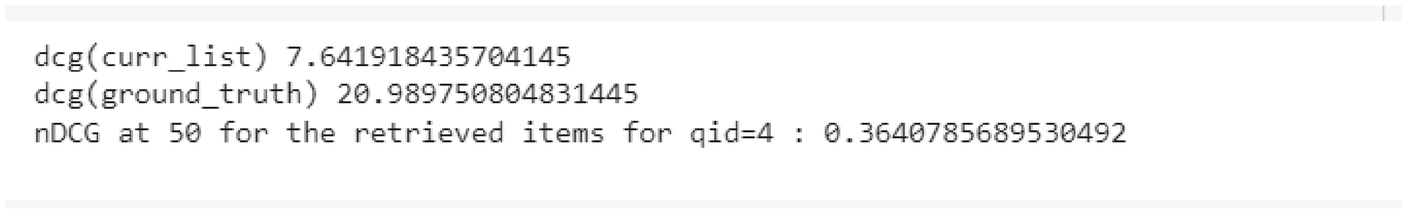
```
nDCG for the retrieved items for qid=4 : 0.6107091565983692
```

(ii) Compute the nDCG (normalized discounted cumulative gain) for the dataset. This involves calculating nDCG at position 50 and for the entire dataset.

Methodology :

1. Extract the first 50 rows from the dataset.
2. Calculate NDCG of the extracted list of relevance judgment values.

Results :
nDCG at 50 for all query-url pairs with qid=4 :



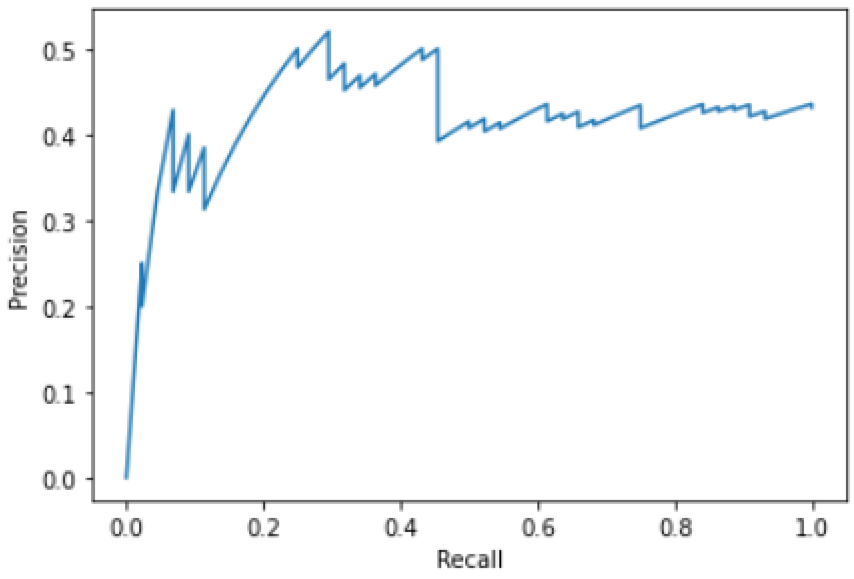
```
dcg(curr_list) 7.641918435704145
dcg(ground_truth) 20.989750804831445
nDCG at 50 for the retrieved items for qid=4 : 0.3640785689530492
```

(iii) Plot a Precision-Recall curve for the query "qid:4".
The curve should help visualize the trade-off between precision and recall as the model's threshold for relevance is adjusted.
For the third objective, assume a model that ranks URLs based on the value of feature 75, which represents the sum of TF-IDF on the whole document. URLs with higher feature 75 values are considered more relevant.

Methodology :

1. Extract relevance judgment values from the dataset with qid = 4
2. Extract feature 75 values from the dataset with qid =4
3. The feature 75 values are considered predicted values and relevance judgment values are the true values
4. From point 3, recall and precision values are calculated incrementally using true and predicted values
5. A plot is generated with recall values at the x-axis and precision values in the y-axis.

Results :
Visualization of the trade-off between Precision and Recall as the model's threshold for relevance is adjusted.



Precision
0.0
0.1
0.2
0.3
0.4
0.5
Recall
0.0
0.2
0.4
0.6
0.8
1.0