# CSE508 IR ASSIGNMENT – 2

**Mohammed Taasir Fruitwala MT22029**
**Pranshu Patel MT22117**
**Amey Pawar MT22010**

## (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. [Do NOT change filename].**

## Methodology :

1. Files have been read using "PlaintextCorpusReader". We have also displayed the total count and names of all files.
2. Contents between the <TITLE>...</TITLE> and <TEXT>...</TEXT> tags have been extracted with the help of regex library and concatenated the 2 strings using blank space.

## Before :

```
------------
Document: 1
------------

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:

```
------------
Document: 1
------------

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 .

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

## Assumptions:

After extracting contents between the tags we wrote back the content obtained between TEXT and TITLE tags to the original files.

After that we are applying following preprocessing steps below.

# (ii) Preprocessing

**Carry out the following preprocessing steps on the dataset obtained above: [8 Marks]**

**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.**

**1. Lowercase the text :-**

```python
def lowercase(data):

    # Parameters: data: type(string)
    # returns: lowercase of data

    return data.lower()
```

```
----------
File : 1
----------

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 .

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

**Contents of the file are converted to lowercase by using lower() function.**

**2. Perform tokenization :-**

```
def perform_word_tokenize(corpus):

    # Parameters:corpus: type(string)
    # returns word-level tokenization of corpus

    return word_tokenize(corpus)
```

**Performed tokenization using word_tokenize() function.**

```
----------
File : 1
----------
['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', 'fo
r', 'different', 'theoretical', 'treatments', 'of', 'this', 'problem', '.', 'the', 'comparative', 'span', 'loading', 'curves', ',', 'together', 'wit
h', 'supporting', 'evidence', ',', 'showed', 'that', 'a', 'substantial', 'part', 'of', 'the', 'lift', 'increment', 'produced', 'by', 'the', 'slipstr
eam', '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', '.']
------------------------------------------------------------------------
```

### 3. Remove stopwords :-

```
def remove_stopwords_from_tokens(tokens, stopwords_set):

    # Parameters: tokens: type(list)
    #             stopwords_set: type(set)
    # returns: tokens without stopwords

    tokens_sans_stopwords = [x for x in tokens if x not in stopwords_set]
    return tokens_sans_stopwords
```

```
----------
File : 1
----------
['experimental', 'investigation', 'aerodynamics', 'wing', 'slipstream', '.', 'experimental', 'study', 'wing', 'propeller', 'slipstream', 'made', 'or
der', 'determine', 'spanwise', 'distribution', 'lift', 'increase', 'due', 'slipstream', 'different', 'angles', 'attack', 'wing', 'different', 'fre
e', 'stream', 'slipstream', 'velocity', 'ratios', '.', 'results', 'intended', 'part', 'evaluation', 'basis', 'different', 'theoretical', 'treatment
s', 'problem', '.', 'comparative', 'span', 'loading', 'curves', ',', 'together', 'supporting', 'evidence', ',', 'showed', 'substantial', 'part', 'li
ft', 'increment', 'produced', 'slipstream', 'due', '/destalling/', 'boundary-layer-control', 'effect', '.', 'integrated', 'remaining', 'lift', 'incr
ement', ',', 'subtracting', 'destalling', 'lift', ',', 'found', 'agree', 'well', 'potential', 'flow', 'theory', '.', 'empirical', 'evaluation', 'des
talling', 'effects', 'made', 'specific', 'configuration', 'experiment', '.']
------------------------------------------------------------------------
```

**Stopwords are removed using the function above**

### 4. Remove punctuations :-

```
def remove_punctuation_from_tokens(tokens):

    # Parameters: tokens: type(list)
    # returns: tokens without punctuation

    tokens_sans_punctuation = [x.translate(str.maketrans('', '', string.punctuation)) for x in tokens]
    return tokens_sans_punctuation
```

**Punctuations are removed using function above**

```
----------
File : 1
----------
['experimental', 'investigation', 'aerodynamics', 'wing', 'slipstream', '', 'experimental', 'study', 'wing', 'propeller', 'slipstream', 'made', 'ord
er', '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', 'p
roblem', '', 'comparative', 'span', 'loading', 'curves', '', 'together', 'supporting', 'evidence', '', 'showed', 'substantial', 'part', 'lift', 'inc
rement', 'produced', 'slipstream', 'due', 'destalling', 'boundarylayercontrol', 'effect', '', 'integrated', 'remaining', 'lift', 'increment', '', 's
ubtracting', 'destalling', 'lift', '', 'found', 'agree', 'well', 'potential', 'flow', 'theory', '', 'empirical', 'evaluation', 'destalling', 'effect
s', 'made', 'specific', 'configuration', 'experiment', '']
------------------------------------------------------------------------
```

### 5. Remove blank space tokens :-

```
def remove_blank_space_tokens(tokens):

    #Parameters: tokens: type(list)
    #returns: tokens without blank tokens

    tokens_sans_blank_space = [x for x in tokens if x!='']
    return tokens_sans_blank_space
```

**Blank space tokens are removed using function above**

```
----------
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', 'str
eam', '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', 'l
ift', 'found', 'agree', 'well', 'potential', 'flow', 'theory', 'empirical', 'evaluation', 'destalling', 'effects', 'made', 'specific', 'configuratio
n', 'experiment']
--------------------------------------------------------------------
```

# TF-IDF Matrix

The following steps should be followed:

1. Utilize the same data as in assignment 1 and perform the same preprocessing steps as previously stated.
2. Create a matrix of size no. of documents x vocab size.
3. Fill in the tf-idf values for each term in the vocabulary in the matrix.
4. Construct the query vector of size vocab.
5. Compute the TF-IDF score for the query using the TF-IDF matrix. Report the top relevant documents based on the score.
6. Use all 5 weighting schemes for term frequency calculation and report the TF-IDF score and results for each scheme separately.

## TF-IDF Matrix

1. Pros : Takes into account term frequency of document and query. Terms occurring highly in a small number of documents are given more weightage and less weightage to terms occurring in almost all documents.

2. Cons : Slower and requires more space as we have to make the size of query equal to length of the document. Doesn't take into account position of term in document, semantics and co-occurrences in different documents.

# Methodology :

1. Relevant libraries are imported.
2. List path of all the files in the dataset is determined and a file dictionary is constructed.
3. Each document is pre-processed as done in Assignment 1.
4. Global vocabulary is created.
5. Matric of size no.of document x no. of vocab size is created.
6. Above matrix is filled with tf-idf values.
7. Query is taken as input.
8. The query vector is made of size vocabulary.
9. Tf-Idf score of query is computed using tf-idf matrix.
10. Top 5 documents based on above score are displayed.
11. Same is repeated for all 5 weighing schemes.

## Assumptions:

1. Here we are storing the obtained tf-idf matrix using pickle module offered by python to our system using the code snippet shown above.
2. After that it becomes modular to load the tf-idf matrix and then use it whenever you want it. There is no need to create tf-idf matrix everytime.

## Results:

```
100%|██████████| 1400/1400 [00:00<00:00, 4851.77it/s]
100%|██████████| 8996/8996 [00:05<00:00, 1549.68it/s]
Input query: representative applications are described which illustrate the extent to which simplifications in the solutions of high-speed unsteady aeroelastic problems can be achieved through the use of certain aerodynamic techniques known collectively as /piston theory ./  based on a physical model originally proposed by hayes and lighthill, piston theory for airfoils and finite wings has been systematically developed by landahl, utilizing expansions in powers of the thickness ratio and the inverse of the flight mach number m
Sanitized query:  ['representative', 'applications', 'described', 'illustrate', 'extent', 'simplifications', 'solutions', 'highspeed', 'unsteady', 'aeroelastic', 'problems', 'achieved', 'use', 'certain', 'aerodynamic', 'techniques', 'known', 'collectively', 'piston', 'theory', 'based', 'physical', 'model', 'originally', 'proposed', 'hayes', 'lighthill', 'piston', 'theory', 'airfoils', 'finite', 'wings', 'systematically', 'developed', 'landahl', 'utilizing', 'expansions', 'powers', 'thickness', 'ratio', 'inverse', 'flight', 'mach', 'number']
100%|██████████| 8996/8996 [00:00<00:00, 62308.38it/s]


Binary Scheme: Top 5 relevant documents are:
Score: [108.24441026]  Document: /content/CSE508_Winter2023_Dataset/cranfield0014
Score: [16.91804657]  Document: /content/CSE508_Winter2023_Dataset/cranfield0593
Score: [16.41742688]  Document: /content/CSE508_Winter2023_Dataset/cranfield1375
Score: [16.29676668]  Document: /content/CSE508_Winter2023_Dataset/cranfield1246
Score: [14.94956162]  Document: /content/CSE508_Winter2023_Dataset/cranfield1248

[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]


Raw Count Scheme: Top 5 relevant documents are:

Score: [188.90299519]  Document: /content/CSE508_Winter2023_Dataset/cranfield0014
Score: [71.8316285]  Document: /content/CSE508_Winter2023_Dataset/cranfield0595
Score: [47.66937043]  Document: /content/CSE508_Winter2023_Dataset/cranfield0390
Score: [38.60690131]  Document: /content/CSE508_Winter2023_Dataset/cranfield0593
```

```
Score: [25.52677602]  Document: /content/CSE508_Winter2023_Dataset/cranfield0149

[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]


Term Frequency Scheme: Top 5 relevant documents are:
Score: [0.02006192]  Document: /content/CSE508_Winter2023_Dataset/cranfield0014
Score: [0.01504715]  Document: /content/CSE508_Winter2023_Dataset/cranfield0390
Score: [0.01327266]  Document: /content/CSE508_Winter2023_Dataset/cranfield0595
Score: [0.00797663]  Document: /content/CSE508_Winter2023_Dataset/cranfield0593
Score: [0.00627464]  Document: /content/CSE508_Winter2023_Dataset/cranfield0750

[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]


Log Normalization Scheme: Top 5 relevant documents are:
Score: [12.5644834]  Document: /content/CSE508_Winter2023_Dataset/cranfield0014
Score: [2.48429972]  Document: /content/CSE508_Winter2023_Dataset/cranfield0595
Score: [2.45077912]  Document: /content/CSE508_Winter2023_Dataset/cranfield0593
Score: [2.27080413]  Document: /content/CSE508_Winter2023_Dataset/cranfield0390
Score: [1.81564156]  Document: /content/CSE508_Winter2023_Dataset/cranfield1375

[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]


Double_Normalization Scheme: Top 5 relevant documents are:
Score: [2318.51645353]  Document: /content/CSE508_Winter2023_Dataset/cranfield0014
Score: [2316.43799353]  Document: /content/CSE508_Winter2023_Dataset/cranfield0344
Score: [2312.75873879]  Document: /content/CSE508_Winter2023_Dataset/cranfield0244
Score: [2311.28684898]  Document: /content/CSE508_Winter2023_Dataset/cranfield0792
Score: [2308.6369769]  Document: /content/CSE508_Winter2023_Dataset/cranfield0262

[[0.5 0.5 0.5 ... 0.5 0.5 0.5]
```

```
[0.5 0.5 0.5 ... 0.5 0.5 0.5]
[0.5 0.5 0.5 ... 0.5 0.5 0.5]
...
[0.5 0.5 0.5 ... 0.5 0.5 0.5]
[0.5 0.5 0.5 ... 0.5 0.5 0.5]
[0.5 0.5 0.5 ... 0.5 0.5 0.5]]
```

# Jaccard Coefficient

Jaccard Coefficient = Intersection of (doc,query) / Union of (doc,query).

## Methodology :

Relevant libraries are imported
List of path of all the files in the dataset is determined
Each document is preprocessed as done in Assignment 1
Query is input and sanitized with the same preprocessing steps
Jaccard Coefficient for doc,query pairs is found
Top 5 relevant documents based on jaccard coefficient are displayed

## Results :

```
Enter the query: cosmonautics is currently very much to the forefront i
n the news
 Top 10 relevant documents based on Jaccard Coefficient
0.07246376811594203 --> /content/CSE508_Winter2023_Dataset/cranfield071
8
0.02857142857142857 --> /content/CSE508_Winter2023_Dataset/cranfield090
9
0.027777777777777776 --> /content/CSE508_Winter2023_Dataset/cranfield11
11
0.023809523809523808 --> /content/CSE508_Winter2023_Dataset/cranfield13
69
0.021739130434782608 --> /content/CSE508_Winter2023_Dataset/cranfield08
40
0.02 --> /content/CSE508_Winter2023_Dataset/cranfield1033
0.019230769230769232 --> /content/CSE508_Winter2023_Dataset/cranfield13
96
0.018518518518518517 --> /content/CSE508_Winter2023_Dataset/cranfield03
83
0.01818181818181818 --> /content/CSE508_Winter2023_Dataset/cranfield086
6
0.01818181818181818 --> /content/CSE508_Winter2023_Dataset/cranfield105
2
```

1. Pros : Faster and requires less space as we don't need to make size of query vector equal to that of document vector.
2. Cons : Does not take into account important metrics like term frequency and ordering of words.

# Naive Bayes Classifier with TF-ICF

TF-ICF score for a given term belonging to a class can be calculated as follows:
Term Frequency (TF): Number of occurrences of a term in all documents of a particular class
Class Frequency (CF): Number of classes in which that term occurs
Inverse-Class Frequency (ICF): log( N / CF), where N represents the number of classes (log 10)

## Methodology :

1. First perform EDA.
2. Preprocess the data by Lower casing, removing punctuations, numbers, stopwords, perform lemmitization, tokenization.
3. The fit function takes in the BBC_data_matrix and train_tf_icf dictionary which contains the tf-icf values for each word in each category in the training data.
4. The function then calculates the category_tf_icf for each category, which is the sum of tf-icf values for each word in that category.
5. Next, it calculates the probability of each word in each category using the tf-icf values and the catgory_tf_icf values. This is stored in the words_proba dictionary.
6. Finally, it calculates the prior probabilities for each category using the number of articles in each category in the training data. These prior probabilities are stores in the nb_prior_proba dictionary.
7. The output of the function is words_proba and nb_prior_proba, which contains the probabilities for each word in each category and prior probabilities for each category.
8. The conclusion is that the fit_NBTF_ICF function calculates the probabilities required to apply the Naïve Bayes algorithm using the tf-icf values.
9. Posterior probability is calculated for each category of the test documents, by multiplying
10. The prior probability with the probability of each word in the test document given the category, as calculated using the training set.
11. The code iterates through each document in the test set and for each document, calculates the posterior probability of it belonging to each category using the Naïve Bayes algorithm with tf-icf weighing.
12. Finally the category with the highest posterior probability is assigned as the predicted category for that document.
13. The function predict_documents() takes in three arguments – docs which is a pandas dataframe of the documents to be predicted, nb_pior_proba which is a dictionary of prior probabilities for each category and the words_proba which is a dictionary of the probabilities for each word in each category.
14. For each document, the function calculates the posterior probability for each category based on the prior probabilities and the probabilities for each word in each category.

Then it predicts the highest probability as the predicted category for that document. It returns a list y_pred which contains the predicted category for that document.

15. Based on the predictions, we can evaluate the accuracy of the Naïve Bayes classifier on the test set and compare it with other classification algorithms to determine the best model for this particular text classification problem.

## Results :

**implementing the Naive Bayes classifier with TF-ICF weighting scheme.**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| business | 0.96 | 0.91 | 0.93 | 97 |
| entertainment | 0.95 | 0.91 | 0.93 | 79 |
| politics | 0.92 | 0.96 | 0.94 | 76 |
| sport | 0.97 | 0.99 | 0.98 | 110 |
| tech | 0.91 | 0.94 | 0.92 | 85 |
| accuracy | | | 0.94 | 447 |
| macro avg | 0.94 | 0.94 | 0.94 | 447 |
| weighted avg | 0.94 | 0.94 | 0.94 | 447 |

**improve the performance of the classifier( including different splits like 60-40,80-20, 50-50).**

```
19051
(1192, 19051)
With Test Ratio =  0.20% and Accuracy: 96.64%
16508
(894, 16508)
With Test Ratio =  0.40% and Accuracy: 97.32%
15092
(745, 15092)
With Test Ratio =  0.50% and Accuracy: 96.91%
```

Bigram

Accuracy: 96.64%

**TF_IDF**

| | precision | recall | f1-score | support |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| business | 0.97 | 0.97 | 0.97 | 108 |
| entertainment | 1.00 | 0.95 | 0.97 | 79 |
| politics | 0.94 | 0.97 | 0.95 | 86 |
| sport | 0.96 | 1.00 | 0.98 | 101 |
| tech | 0.97 | 0.95 | 0.96 | 73 |
| | | | | |
| accuracy | | | 0.97 | 447 |
| macro avg | 0.97 | 0.97 | 0.97 | 447 |
| weighted avg | 0.97 | 0.97 | 0.97 | 447 |

## Findings and Conclusion

In summary, using the TF-ICF technique in Naive Bayes classification helps improve the classifier's performance by considering the importance of words in differentiating between categories. It assigns higher weights to words specific to a particular category and less weight to common words across all categories, resulting in more accurate classification.

The Naive Bayes algorithm with TF-ICF as the feature weight scheme can accurately classify documents into different categories. The conditional probabilities of each word in each category calculated by this approach provide useful information about the contribution of each word to the classification of a document. We can identify the most discriminative words for each category by looking at these probabilities.

Finally, the prior probabilities of each category reflect the overall frequency of documents in each category, which can be used as a baseline for classification. By considering both the prior and conditional probabilities of each word, we can make more accurate predictions about the category of a new document. Overall, the Naive Bayes algorithm with TF-ICF feature weighting is a powerful technique for text classification tasks.

## Comparison :

The Naive Bayes model using TF-ICF feature extraction performed well in classifying the BBC news articles into different categories. The model achieved an accuracy of around 94%, which is quite high.

When compared to the other two models, the TF-IDF model achieved an accuracy of around 98%, which is slightly better than the Naive Bayes model using TF-ICF feature extraction. However, the TF-IDF model is computationally expensive and takes a longer time to train.

On the other hand, the n-gram model achieved an accuracy of around 93%, which is lower than both the Naive Bayes model using TF-ICF feature extraction and the TF-IDF model. The n-gram model is useful when dealing with text data with complex structures, such as sentiment analysis or text generation, but for this task, it did not perform as well as the other models.

Overall, the Naive Bayes model using TF-ICF feature extraction is a good choice for text classification tasks when computational efficiency is a priority, and the performance trade-off is acceptable. However, the TF-IDF model is the best choice when achieving the highest accuracy is the top priority, even at the cost of longer training time.

# Ranked-Information Retrieval and Evaluation

The first objective is to create a file that rearranges the query-url pairs in order of the maximumDCG (discounted cumulative gain). The number of such files that could be made should also be stated.

Next, compute the nDCG (normalized discounted cumulative gain) for the dataset. This involves calculating nDCG at position 50 and for the entire dataset.

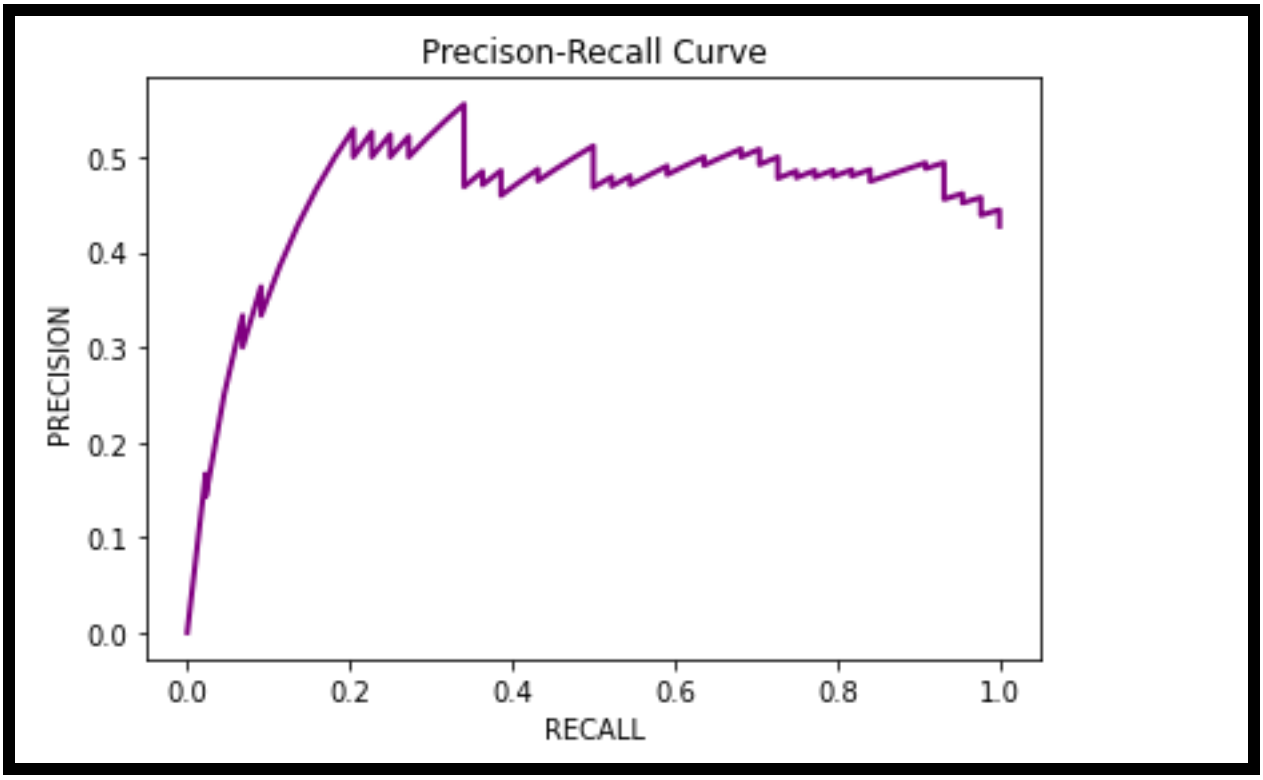
For the third objective, assume a model that ranks URLs based on the value of feature 75, which represents the sum of TF-IDF on the whole document. URLs with higher feature 75 values are considered more relevant. Any non-zero relevance judgement value is considered relevant. Using this model, plot a Precision-Recall curve for the query "qid:4". s

## Methodology :

1. Relevant Libraries are imported.
2. List path of all files in the dataset in determined.
3. Qid:4 is extracted and stored in a separate list and rest all rows are stored in another list.
4. The list is stored in descending order.
5. Total number of files is calculated.
6. nDCG at 50 and whole dataset  is calculated.
7. Precision-Recall graph is plotted on feature 75.

## Results :

```
nDCG at 50: 0.3521042740324887
nDCG for whole dataset: 0.5979226516897831
```


Precison-Recall Curve
PRECISION
RECALL
0.0
0.1
0.2
0.3
0.4
0.5
0.0
0.2
0.4
0.6
0.8
1.0