IR Assignment 2 Report

Drishya Uniyal MT21119

Shashank Rustagi MT21081

Group 5

Question 1

Dataset: Same as in assignment 1. (Humor,Hist,Media,Food)

- **Preprocessing**
  - ★ Open the file with ISO-8859-1 encoding.
  - ★ Delete special characters.
  - ★ Delete numbers.
  - ★ Tokenise the words.
  - ★ Convert to lower case.
  - ★ Removal of stop words using nltk.
  - ★ Lastly, a set of all the words is created.

● Methodology

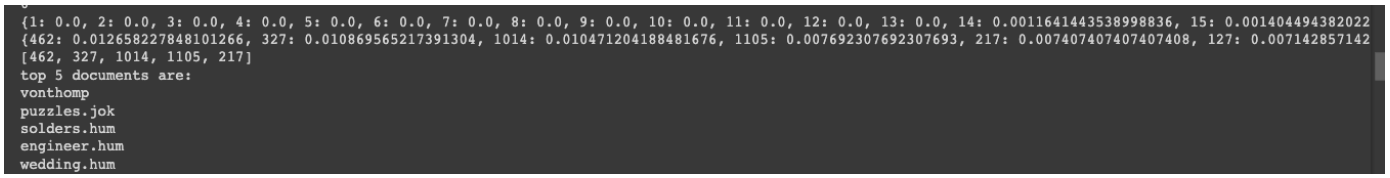
Part A Jaccard Coefficient

- Reading all the files in the folder.
- Making functions to find union, intersection and calculate jaccard coefficient.
  - Jaccard Coefficient = Intersection of (doc,query) / Union of (doc,query)
- Create dictionary to store jaccard coefficient with respective doc-id in key:value format.
- Calculate the top 5 documents according to value of jaccard coefficient.
- Process the input query and call the funcions.
- Print the top 5 documents.

Results:

Query: lion stood thoughtfully for a moment. (preprocessing done first)

Output:



Part B

TF-IDF Matrix

Preprocessing

1. Open the file with ISO-8859-1 encoding.
2. Delete special characters.
3. Delete numbers.
4. Tokenise the words.
5. Convert to lower case.
6. Removal of stop words using nltk.
7. Lastly, a set of all the words is created.

Methodology

1. Reading all the files in the folder.
2. TF-IDF calculation. Find document frequency. Make a dictionary that will contain a number of documents for each term in key-value format.
3. Find inverse document frequency.
   IDF(word)=log(total no. of documents/document frequency(word)+1)
4. Calculate term frequency.
5. Make a function to calculate TF-IDF value = tf*idf.
6. Binary Term frequency calculation. Always be 1 for all values because no word with 0 freq is added in the list; words with list freq 1 are added to the list.
7. Calculate raw count term frequency.
8. Calculate term frequency.
9. Calculate log term frequency.
10. Calculate double log normalisation.
11. Get the top 5 documents for each.
12. Process the query and get results.

Advantages: -

Easy to compute basic metric to extract the most descriptive terms in a document

Normalization : Log is said to be used because it “dampens” the effect of IDF as data increases it does not affect the scores using log as it ormalizes the data

Disadvantages: -

Normalisation : based on the bag-of-words (BoW) model, therefore it does not capture position in text, semantics.

Results.

Part B

Query: lion stood thoughtfully for a moment

Output:

```
top 5 documents for binary tfidf are:
pizzawho.hum
boneles2.txt
murphys.txt
tpquotes.txt
collected_quotes.txt
top 5 documents for raw_count tfidf are:
lions.cat
lion.jok
lion.txt
barney.txt
boneles2.txt
top 5 documents for termfreq tfidf are:
lion.txt
lion.jok
lions.cat
solders.hum
boneles2.txt
top 5 documents for log tfidf are:
lions.cat
lion.jok
lion.txt
barney.txt
boneles2.txt
top 5 documents for doublelog tfidf are:
pizzawho.hum
boneles2.txt
lion.jok
lions.cat
lion.txt
```

Q2.

Ranked Information Retriecal and Evaluation

Part A

1. Open the file using encoding as uniocde_escape. Read through the lines. Process through the qid:4.
2. Sort these lines based on the first column as a relevance score for max dcg and store in a file. After applying to qid:4 and finding max DCG, we obtain a total of **1!17!29!59!** files.
3. Save the file as maxDCG.csv.

Part B

1. Calculate the dcg for 50 documents for URL with qid:4 and the same calculate idcg for 50 documents after sorting the lines by its relevance score. Apply the formula.
2. Calculate ndcg=dcg/idcg. Same for all the data with qid:4.

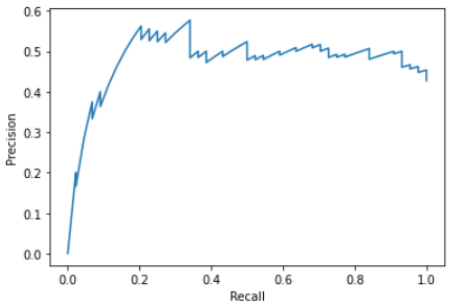
Part C

1. For values of feature 75 and qid:4, calculate precision and recall and store these values in a list, and by using the matplotlib library plot the Precision-Recall curve.

Results:

Total files that can be obtained= **1!17!29!59!**
nDCG at 50 = 0.35
nDCG at whole dataset = 0.578

```
[0 1 3 2]
59
26
1
17
The number of maximum DCG ordering possible for qid:4 will be : 19893497375938370599826047614905329896936840170566570588205180312704857992695193482412686565431050240000000000000000000000
nDCG at 50: 0.35612494416255847
nDCG for whole dataset: 0.5784691984582591
```



Q3.

Naive Bayes Classifier

Two files are included in the operation, the first file consists of preprocessing taking the five classes comp.graphics, sci.med, talk.politics.misc,rec.sport.hockey, sci.space having the same folder name.

Pre-processing:

1. Strip the data for each file
2. Convert into lower case
3. Tokenise the data
4. Converting the numbers into a word like 12 -> twelve
5. Remove punctuations.
6. Removing single alphabet.
7. Remove stop words
8. Apply lemmatisation
9. Porter stemmer.

Division of data: The data was split randomly using a NumPy library, using ratios 0.5, 0.7, and 0.8 as ratios.

TF-ICF: 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 class frequency technique as given in the question was used.

Results:

Confusion Matrixes

```
Counter({'talk.politics.misc': 538, 'rec.sport.hockey': 518, 'comp.graphics': 501, 'sci.med': 488, 'sci.space': 461})
Accuracy at 0.5 and at feature value 10 :  0.7794707297514034
[[315   1   0   5   0]
 [170 470   0  36   2]
 [  0   0 512   7   0]
 [  0   0   0 187   0]
 [ 14  11   0 304 460]]
Accuracy at 0.5 and at feature value 20 :  0.8448275862068966
[[381   4   0   6   0]
 [102 462   0  77   2]
 [  0   1 512   4   0]
 [  0   4   0 293   1]
 [ 16  11   0 159 459]]
Accuracy at 0.5 and at feature value 40 :  0.9125902165196471
[[400   6   0   3   3]
 [ 85 452   0  21   2]
 [  1   3 510   2   1]
 [  0  13   2 459   1]
 [ 13   8   0  54 455]]
Accuracy at 0.5 and at feature value 60 :  0.9366479550922213
[[431  10   0   5   2]
 [ 52 449   0  16   2]
 [  2   1 511   2   1]
 [  2  15   1 489   1]
 [ 12   7   0  27 456]]
Counter({'rec.sport.hockey': 715, 'sci.med': 705, 'sci.space': 696, 'comp.graphics': 695, 'talk.politics.misc': 685})
Accuracy at 0.7 and at feature value 10 :  0.8384308510638298
[[190   0   0   3   1]
 [112 281   0 102   4]
 [  0   0 295   3   0]
 [  1   1   0 196  11]
 [  2   3   0   0 299]]
Accuracy at 0.7 and at feature value 20 :  0.8856382978723404
[[229   2   0   3   2]
 [ 71 277   0  51   4]
 [  2   2 295  23   2]
 [  1   1   0 227   3]
 [  2   3   0   0 304]]
```

```
Accuracy at 0.7 and at feature value 40 :  0.932845744680851
[[256   4   0   3   3]
 [ 45 271   0  18   3]
 [  1   1 295   7   2]
 [  0   6   0 275   1]
 [  3   3   0   1 306]]
Accuracy at 0.7 and at feature value 60 :  0.9454787234042553
[[272   6   0   4   2]
 [ 27 264   0  10   3]
 [  1   1 295   5   1]
 [  2  11   0 284   2]
 [  3   3   0   1 307]]
Counter({'sci.med': 799, 'talk.politics.misc': 791, 'comp.graphics': 785, 'rec.sport.hockey': 777, 'sci.space': 775})
Accuracy at 0.8 and at feature value 10 :  0.7670083876980429
[[209 111   0  16   4]
 [  2 110   0   3   0]
 [  0   0 201 107   0]
 [  0   0   0  98   0]
 [  4   2   0   1 205]]
Accuracy at 0.8 and at feature value 20 :  0.8089468779123952
[[207  98   0  12   4]
 [  2 120   0   4   0]
 [  3   2 201  73   1]
 [  0   0   0 136   0]
 [  3   3   0   0 204]]
Accuracy at 0.8 and at feature value 40 :  0.8872320596458527
[[203  59   0   5   4]
 [  1 151   0   7   0]
 [  3   2 201  20   0]
 [  0   8   0 192   0]
 [  8   3   0   1 205]]
Accuracy at 0.8 and at feature value 60 :  0.9151910531220876
[[205  42   0   2   4]
 [  2 168   0   5   0]
 [  2   4 201  14   0]
 [  1   6   0 203   0]
 [  5   3   0   1 205]]
```

% OF TRAINING DATA CONSIDERED	FEATURES(TF-ICF BASED) SELECTED/CLASS	ACCURACY
50 %	10	77.94707297514034 %
50 %	20	84.48275862068965 %
50 %	40	91.25902165196472 %
50 %	60	93.66479550922213 %
70 %	10	83.84308510638297 %
70 %	20	88.56382978723404 %
70 %	40	93.2845744680851 %
70 %	60	94.54787234042553 %
80 %	10	76.70083876980429 %
80 %	20	80.89468779123952 %
80 %	40	88.72320596458528 %
80 %	60	91.51910531220877 %