CS F469 INFORMATION RETRIEVAL

ASSIGNMENT 1

Design Document

Topic: Plagiarism Checker

Group Members

Rohan Maheshwari (2017B4A70965H)

Giridhar Bajpai (2017B4A71451H)

Soumil Agarwal (2017B4A71606H)

Objective

This project is aimed towards creating a plagiarism checker and rank documents based on similarity matching a user provided query/answer.

Pre-processing

The given corpus has 100 documents. The average length of a file in the corpus is 208 words with 113 unique tokens.

A file in accepted from the user and preprocessing is done as follows

Step 1: The raw data in the accepted input file is processed and a list of paragraphs is returned from the input document

Step 2: The list of paragraphs are then tokenized.

Step 3: Stop words like 'the', 'is', 'me', etc are removed from the obtained tokens.

Step 4: The cleaned document undergoes a further pre-processing step called stemming. The tokens were stemmed using the Porter's algorithm.

Methodology

Built in Modules used:

1. RegexpTokenizer(): With the help of tokenize.regexp() module, tokens from string can be extracted using regular expression and the RegexpTokenizer() method.

2. stopwords.words(): used for removing the built in stop word listed in NLTK.

3. Porterstemmer(): used as a normalization technique on the obtained tokens.

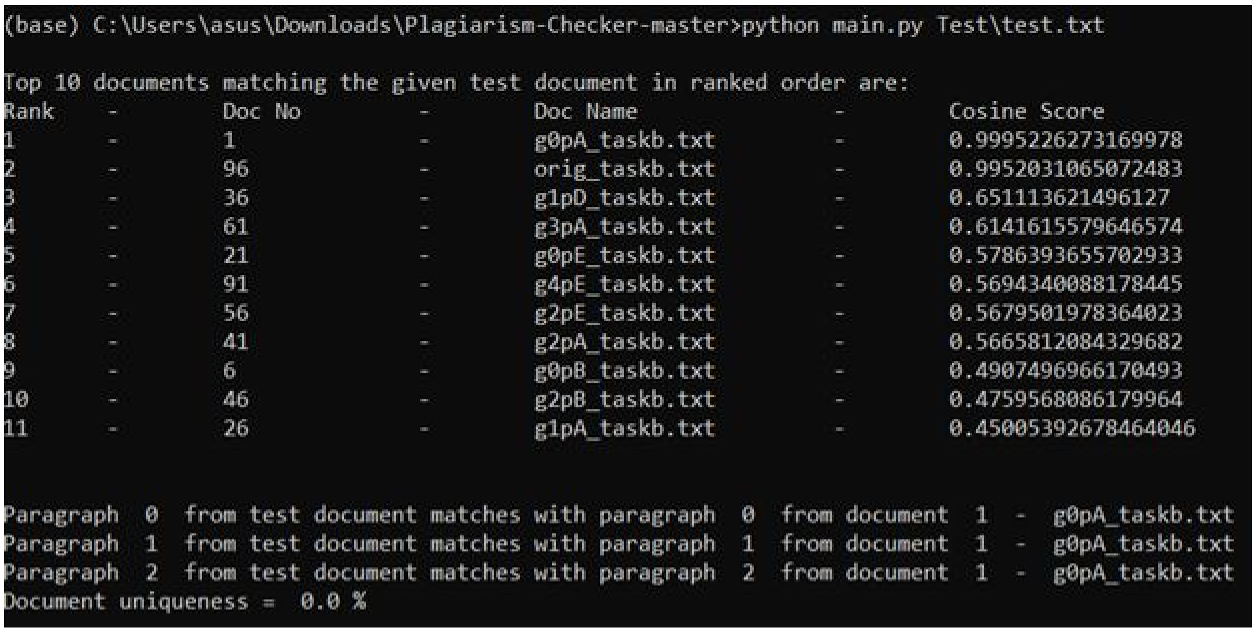
PROGRAM FLOW

1. After preprocessing was done we created the inverted index in the form of a posting list. We used a dictionary to store the word and its posting list as key value pairs.
2. We then proceeded to create the TfIDf scores for all the documents in the form of a dictionary which consisted of a word and associated list (length being number of documents) containing the tf*idf scores for the documents for that respective word. We used ltc.ltc format for calculating the scores.
3. After that processing of the test document was done in a similar way as the training corpus. We then used the TfIDf vectors to compute both matching similarity and cosine similarity between the query doc and our corpus.
4. Top 10 documents with the highest similarity score were displayed to the user. This was done by taking a test document as a query.
5. Next we found from where the paragraphs were plagiarised from (if they were) and reported the same to the user displaying the doc number name and paragraph number from that particular document. At the end we simply calculated the amount of uniqueness in the document provided by the user wrt to our training corpus.

Data Structures Used:

1. *List:* Used for storing the paragraphs, tokens and stemmed tokens.
2. *Dictionary:* Used for creating the inverted index, finding the tf-idf score for documents in given corpus and calculating the rank of documents based on similarity.
3. *Set:* Used for vocabulary of training data so as to store only a single occurrence of word that occur multiple times in the corpus. This vocabulary set was later used to create the inverted index.

SAMPLE RESULTS

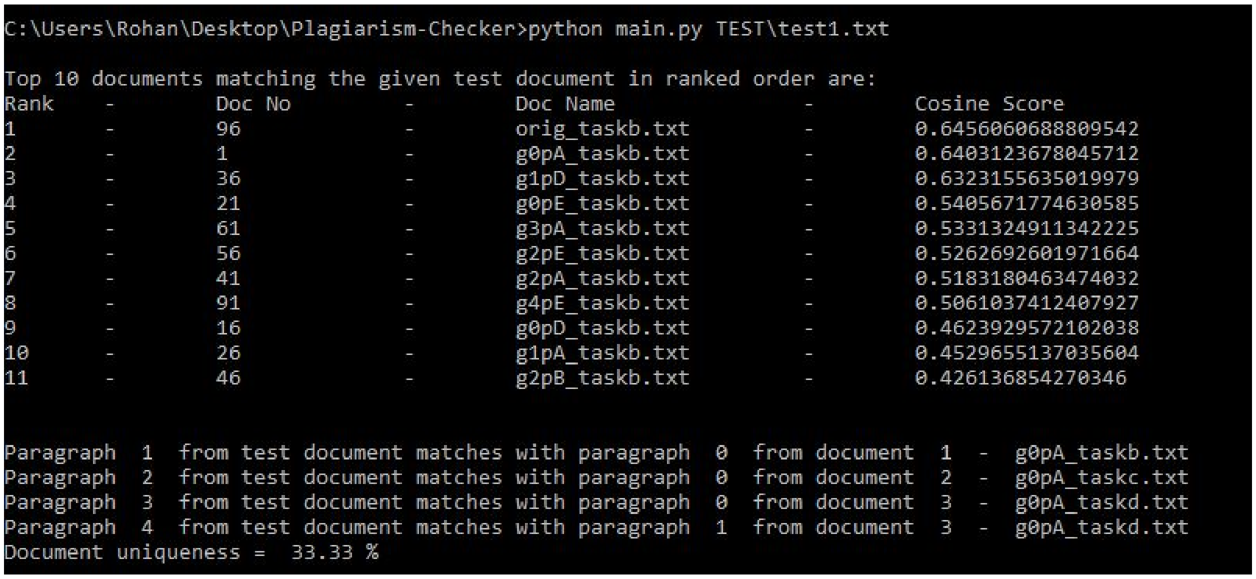


```
(base) C:\Users\asus\Downloads\Plagiarism-Checker-master>python main.py Test\test.txt

Top 10 documents matching the given test document in ranked order are:
Rank    -       Doc No          -       Doc Name            -       Cosine Score
1       -       1               -       g0pA_taskb.txt      -       0.9995226273169978
2       -       96              -       orig_taskb.txt      -       0.9952031065072483
3       -       36              -       g1pD_taskb.txt      -       0.651113621496127
4       -       61              -       g3pA_taskb.txt      -       0.6141615579646574
5       -       21              -       g0pE_taskb.txt      -       0.5786393655702933
6       -       91              -       g4pE_taskb.txt      -       0.5694340088178445
7       -       56              -       g2pE_taskb.txt      -       0.5679501978364023
8       -       41              -       g2pA_taskb.txt      -       0.5665812084329682
9       -       6               -       g0pB_taskb.txt      -       0.4907496966170493
10      -       46              -       g2pB_taskb.txt      -       0.4759568086179964
11      -       26              -       g1pA_taskb.txt      -       0.45005392678464046


Paragraph  0  from test document matches with paragraph  0  from document  1  -  g0pA_taskb.txt
Paragraph  1  from test document matches with paragraph  1  from document  1  -  g0pA_taskb.txt
Paragraph  2  from test document matches with paragraph  2  from document  1  -  g0pA_taskb.txt
Document uniqueness =  0.0 %
```

Results from copying the same document g0pA_taskb.txt from the train set into the test file which was already heavily plagiarised from orig_taskb.txt doc.As we can see our model gives accurate cosine scores.



```
C:\Users\Rohan\Desktop\Plagiarism-Checker>python main.py TEST\test1.txt

Top 10 documents matching the given test document in ranked order are:
Rank    -       Doc No          -       Doc Name            -       Cosine Score
1       -       96              -       orig_taskb.txt      -       0.6456060688809542
2       -       1               -       g0pA_taskb.txt      -       0.6403123678045712
3       -       36              -       g1pD_taskb.txt      -       0.6323155635019979
4       -       21              -       g0pE_taskb.txt      -       0.5405671774630585
5       -       61              -       g3pA_taskb.txt      -       0.5331324911342225
6       -       56              -       g2pE_taskb.txt      -       0.5262692601971664
7       -       41              -       g2pA_taskb.txt      -       0.5183180463474032
8       -       91              -       g4pE_taskb.txt      -       0.5061037412407927
9       -       16              -       g0pD_taskb.txt      -       0.4623929572102038
10      -       26              -       g1pA_taskb.txt      -       0.4529655137035604
11      -       46              -       g2pB_taskb.txt      -       0.426136854270346


Paragraph  1  from test document matches with paragraph  0  from document  1  -  g0pA_taskb.txt
Paragraph  2  from test document matches with paragraph  0  from document  2  -  g0pA_taskc.txt
Paragraph  3  from test document matches with paragraph  0  from document  3  -  g0pA_taskd.txt
Paragraph  4  from test document matches with paragraph  1  from document  3  -  g0pA_taskd.txt
Document uniqueness =  33.33 %
```

Here we wrote the first and last paragraph on our own and plagiarised other 4 paragraphs randomly from the training set which resulted in these scores.

Average Time taken for running a file of average 200 words over 10 runs was recorded to be *1.746s*.

Challenges Faced:

1. We first started to measure similarity with Matching Similarity metric(Direct sum of TfIDf values). In the case where we copy paste the same text over and over again in the test document the score gets very high for that particular document from where this part was taken. So we decided to use cosine similarity instead so as to use angle as a metric instead of distance and it fixed that problem.
2. Words included in the test document but never seen in the training corpus gave error as they were not found in the inverted index. So we dynamically assigned these new words that we encountered an empty list in the inverted index and zero scores in the TfIDf dictionary.