README(Assignment 2)

Akshaj Patil(MT19111)

1:PreProcessing Steps:

- Case folding (all terms are converted to lower case).
- Stop words removal.
- Removal of punctuations.
- Removal of numbers.
- Lemmatization.

Same pre-processing is done for data and query.

2:Assumptions:

- Meta data of the documents is considered as part of document.
- There were 3 files which my python version was unable to read I have not considered those files.
- Index.html is not considered as part of document.
- Query should be of at least 1 word.

3:Methodology:

Q1:] Here while running code for first time program will ask for idf variation 1:IDF without log 2:IDF with log.
If you press '1' raw idf will be calculated with formula "total-documents/df"
If you press '2' log idf will be calculated with formula "log(total-documents/df)"

Then program will ask user to enter his particular choice of algorithm 1:Jaccard 2:Tf-idf 3:TF-idf with Title 4:Cosine Similarity 5:Exit.

1 Jaccard)

First we made dictionary of all the documents with their name as key and list of respective lemmatized tokens as value. When query comes Query is first preprocessed then I have taken intersection and union of query terms with each document and calculated jaccard score with ((A intersection B)/(A union B)) Then sort documents according to jaccard score in descending order and then retrieve first 'k' documents.

2 Tf-Idf variants)

Here we have calculated idf of each term. And then while calculating tf-idf program will ask for which tf variant you want 1:Raw tf that is just term frequency, 2:Normalized tf by length of document (tf/(len(doc)), 3:Normalized by log (log(tf/len(doc)). In this way tf-idf is calculated in given way. Once tf-idf is calculated then sort them according to descending score and retrieve first 'k' documents.

3 Tf-idf title)

Here from index.html file we have created dictionary with file name as key and list of title elements as value. When query is preprocessed each query term is checked whether it is present in title or not. If present in title then give extra weight to the score, if not present then keep the original score. Once tf-idf is calculated considering title then sort them according to descending score and retrieve first 'k' documents.

4: Cosine Similarity

Here tf-idf vector is designed of both query and document then cosine similarity is calculated between two vectors. After calculating similarity score sort it according to descending score and retrieve first 'k' documents.

If you want to change value of 'k' then go in code and update the value.

If You want to exit the program then press 5.

Q2]

Here after running the program user first have to enter the query. We will find all miss-spelled words in the query then we will take first word and compute edit-distance with all the dictionary words and store their respective cost. After computing sort words according to cost in ascending order then retrieve first 'k' words.

Here in Q2] cost of edit-distance is of converting query words to dictionary miss-spelled words.