IR A2 README FILE

Question 1:

Pre-processing steps used-

- All the text is converted to lower case
- All email-ids are removed from the text
- Tokens which consist of letters as well as digits like play2 are removed from the text.
- All the punctuation marks are removed from the text

The above pre-processing is applied for the body text as well as the document titles text.

Part 1:

Methodology:

- A nested dictionary is prepared with document names as key and the Counter object containing counts of tokens as value. 2 dictionaries are created like this- one for the document bodies and one for the document titles.
- The Jaccard co-efficient is calculated between the query and each document b using the formula:
  JACCARD(A, B) = |A ∩ B| / |A U B|

Part 2:

Methodology:

- Here multiple dictionaries are created and pickled for using while applying different tf-idf types to find the matching score of a query with the different documents.
- A dictionary containing nt values for the different types in the vocabulary are created.
- 2 dictionaries one containing document names as key and the list of tokens in the body as list and the other containing the list of tokens in titles as list are prepared.
- The different tf-idf methods are tried and the matching score is calculated for the query
- The different tfs tried are : tf-binary, tf-raw count, tf-term frequency, tf-log normalization, tf- double normalization 0.5
- The idf used is : inverse document frequency max.
- Now the matching score is found for the query with each document by finding the sum of tf-idf values of the terms that present in the query as well as the document.
- While considering titles also the total matching score with each doc is calculated as:

  Total matching score = 0.7 * (matching score with title) + 0.3 * (matching score with body)

Assumptions:

- All the document files i.e stories files contain the story title at the top which is removed while creating the dictionaries of only body contains while handling the case of giving special attention to titles.

Variants of tf weight:

Weighting scheme	**Tf weight**
Binary	0, 1
Raw count	f(t,d)
Term frequency	F(t.d) / ∑f(t'd)
Log normalization	log(1 + f(t,d))
double normalization 0.5	0.5 + (0.5 * (f(t, d) / max(f(t', d))))

Part 3: According to the analysis done in part 2, the tf double normalization 0.5 method for tf gives the best result. In this part, I have tried the tf log normalization and tf double normalization 0.5.

Methodology:

- A matrix is prepared of the size no. of docs X vocab size of body. Here each row is a vector of the document prepared using the tf-idf values of all the terms in the vocabulary where the idf value of the term remains same across all the documents and the tf value of that term in the document is used.
- Similarly, a matrix is prepared for the titles vocabulary.
- Now, when a query is given by the user as input, first the vector of size equal to size of vocab of the body is prepared. While preparing the vector for different terms the tf values in the query are used while the idf values in the corpus are used.
- Similarly the query vector is prepared using the titles vocabulary.
- Now the cosine similarity between the query vector and the vectors of bodies and titles of different documents is calculated.
- While considering titles the total cosine similarity score with each doc is found as: Total score = 0.7 * (cosine sim with title vector) + 0.3 * (cosine sim with body vector)

Question 2:

Methodology:

- First, I have prepared a dictionary data structure of all the words present in the file english2.txt.
- Now when a sentence is entered as input by the user. The following steps are applied on the text in the sentence:

- All the text is converted to lower case.
- The tokens which are just numbers like 123 are removed from the sentence.
- If there are tokens like bad2 then they are left as it is.
- All punctuation marks except ‘ are removed.
- Now for the words not present in dictionary the edit distance algo is applied.

Assumptions:

- If user does a typo in a word in the sentence then he may mistype an additional digit along with the letters but if the punctuation is typed then the cost of deletion of punctuation is not counted.