IR Assignment 2

Shalini Bhardwaj( MT19045 )

Ques1: Retrieve Ranked Documents

**1.Preprocessing**

Preprocessing steps: Nltk library is used for preprocessing

1. To Lowercase: query is transformed to lower case

2. Punctuation is removed.

3. Stop words are removed

4. Tokenization : converted to tokens

5. Lemmatization : converting to root dictionary word

Both query and dataset are preprocessed.

Files from all folders are kept in one place and then read.

**2.Methodology**

1.All the above preprocessing steps are performed initially.

2.While lemmetizing documents. Idf score is calculated for each term by calculating in how many documents term has arrived.

3.Following metrices score is calculated :

1.Jaccard Coefficient

2.Tf-idf score

3.Tf-idf score with title weightage

4.Cosine Similarity

5.Cosine Similarity with title weightage

4.User can choose from the above scores to retrieve ranked documents.

5.While calculating tf-idf and cosine similarity user is asked to choose different variation of tf and idf.

6. Tf variation:

1. count of term in doc/length of doc

2. count of term

3. 1+log(1+count of term in doc)

Idf variation:

1. Number of total docs/number of docs where term is present
2. Log(Number of total docs/number of docs where term is present +1)
3. Log(Number of total docs/number of docs where term is present)

7. If a term is present in title of document then that document is given more priority . If term is not present in title then score is multiplied by 0.7

8.Title are accumulated in a text file.Then dictionary is built having filename as key and tokenize title of document as its list.

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 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 user will be asked for which tf and idf variant you want . 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 .txt file is used.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.

5.Cosine Similarity with title

Here wigthage to query vector is given if in title term is present. Rest same as cosine similarity.

User is asked to enter value of k.

3.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.
* In title I have assumed that if term is in title then it would be given more weightage.
* If term not present in any document its idf is done 0.
* Different formulas are appropriate for different types of queries.
* Sum of zone weightage is kept 1

Q2] EDIT DISTANCE

Preprocessing :

1. To Lowercase: query is transformed to lower case

2. Punctuation is removed.

3.Numbers are removed.

4. Tokenization : converted to tokens

Methodology:

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 cost of edit-distance is of converting query word to dictionary word.

Assumption:

If punctuation is in query .It is removed and terms are concatenated