Information Retrieval

Assignment – 2 Report, Group - 25

Question 1

Preprocessing

For preprocessing, the main steps followed were:-:

- Converting the text to lower text.
- Word tokenization
- Removal of stop words from the text
- Removal of punctuation and space tokens
- Stemming over the token to convert tokens to their base words.

Methodology

We implemented the positional index using the dictionary in which for each word, its docs list and position of the word in each doc were stored. We stored the positional index in a file named store.dat to make our system a little more efficient as now, for every query, we do not have to make the positional index from scratch.

Question 2(a)

Preprocessing

- Converting the word of the text to lower case
- Then word Tokenization
- Removing stopwords
- Removing punctuations
- Word stemming

Methodology

- First, need to read each docs taking the query from the user.
- converting the query part by preprocessing. The converting the query part by preprocessing.
- Converting each doc by preprocessing.
- Now taking the intersection of doc and query.
- Now taking the Union of doc and query
- The Jaccard coefficient of each doc = intersection (doc and query) / Union (doc and query)
- After that taking the top 5 documents.
- And print their file path and file name with thor Jaccard coefficient

Question 2(b) & (c)

Preprocessing

In a preprocessing part we have used these processing for both story directory file content and query string

- Converting all character in the lower case
- Removing all punctuation from a string
- Tokenizing the string
- Removing the stopping words
- Applying the Lemmatization

Methodology

- First, we have read all file in a story directory iteratively and preprocess the file content, and then create the unigram data structure for maintain the record of occurrence of unique term in a document
- Now we have to perform TF-IDF in 5 different types of weighting schemes. So, first we have created the TF matrix of all 5 schemes
- Now we are performing the product operation between TF matrix and IDF value and update matrix as value of TF-IDF
- Now Taking input from the user and we have performed the preprocessing steps of a query string and after that we had created the query vector of vocabulary size

- Now we have performed the operation of finding the TF-IDF score of each document in which the query term exists into the document, and pick out the top 5 documents according to the highest TF-IDF score
- Now we have performed the operation of finding the Cosine similarity between query vector and document vector, pick out the top 5 documents according to the highest cosine similarity score

Pros and cons of all 5 weighting schemes

Pros:

- Binary

  It gives you some sense by marking (0,1) whether the particular term present in this document or not.

- Raw Count

  It will account and prioritize the term which occurred many times in a document

- Term Frequency

  It will account and prioritize the term occurred in a document relative to the document length

- Log Normalization

  It will account prioritize the rare term also in a document which is occurred less in document but are more informative

- Double Normalization

  It will account and prioritize the term occurred in a document relative to the maximum occurrence of term in document

Cons:

- Binary

It does not care about occurrence frequency of a term in a document, if a term1 is present many times in a document and a term2 present only one time, this will treat both terms in the same way.

- Raw Count

  It does not account the how many total term occurrences in a document (which is the length of document)

- Term Frequency

  It does not care about the rare term in a document which is occurred less in document but are more informative

- Log Normalization

  It does not consider the length of a document, if a document is large or small

- Double Normalization

  It does not consider the length of a document, how term occurred relative to document length

Question 3

Preprocessing

- Importing .txt file
- Making array of lines having qid:4
- Index finding
- Sorting in descending order to rank
- Finding precision and recall
- Plotting curve

Methodology

- First we have imported the .txt file and made the data frame with pandas
- Then retrieved list of indexes of URLs having qids ==4
- Made the array storing the data of relevance
- Counted the frequency of particular relevance i.e. of 0,1,2,3
- Sorted the array of URLs in descending order i.e. more relevant at first

- For maximum, no of file counted the factorial of the sorted array
- Now considered the value of feature 75 as relevance value
- Made an array with relevant data and made a judgment table whether it is relevant or not i.e. if its relevant then the value given is 1 else 0

$$\text{precision} = \frac{\text{tp}}{\text{tp} + \text{fp}} = \frac{\text{retrieved and relevant documents}}{\text{all retrieved documents}}$$

- *Mathematical definition of precision*

$$\text{recall} = \frac{\text{tp}}{\text{tp} + \text{fn}} = \frac{\text{retrieved and relevant documents}}{\text{all relevant documents}}$$

- *Mathematical definition of recall*
- Predicted the precision and recall for each value and stored in x-axis and y-axis
- Plotted the precision-recall curve

Any assumptions

value of feature 75 (sum of TF-IDF on the whole document) i.e. the higher the value, the more relevant the URL also any non zero relevance judgment value is considered as relevant.