Readme(Assignment 4)

(MT19111) Akshaj Patil

1:PreProcessing Steps:

• Case folding (all terms are converted to lower case).
• Stop words removal.
• Removal of punctuations.
• Numbers are converted to Words.
• Lemmatization.
Same pre-processing is done for data and query.

2:Assumptions:

- Meta data of the documents is considered as part of document.
- Query should be of at least 1 word.
- k is taken to be 100.
- p is set as 20 and p documents belonging to the ground truth from each iteration is marked as relevant.

3:Methodology:

Q1:]Here we preprocessed all the files. We first constructed inverted index of all the terms with docId as a key and tf-idf of that doc as a value.

Then We made document vector of all documents with dimension as vocab size.

When User runs the code. It will ask user to enter query. When users enter query query will be preprocessed and will be converted into list of lemmatized words. Then user have to enter number of docs he want to retrieve. Then we made a Query vector with dimension vocab size. Here to calculate cosine score I only considered those documents in which there is at least one query term present. If any of the query term is not present in vocab then code will not return any document. Then User have to enter querys ground truth folder. Then Cosine score will be calculated between query and documents and document will re-arranged according to decreasing order of cosine score.

Then first 20 documents which are retrieved and are in ground truth will be marked as relevant and query vector will be updated by Rocchio’s algorithm.

$$\begin{aligned}\dot{\vec{q}}_m &= \alpha\vec{q}_0 + \beta\mu(D_r) - \gamma\mu(D_{nr}) \\ &= \alpha\vec{q}_0 + \beta\frac{1}{|D_r|}\sum_{\vec{d}_j\in D_r}\vec{d}_j - \gamma\frac{1}{|D_{nr}|}\sum_{\vec{d}_j\in D_{nr}}\vec{d}_j\end{aligned}$$

And again cosine similarity is performed on updated query vector with all docs and documents are retrieved in decreasing order of cosine score. Here ‘*’ is appended in front of document which is marked as relevant. And above iteration is repeated 4 times. TSNE graph is plot to see changes in query vector.

Functions:

def calculateIdf(df): To calculate IDF of terms in vocab.

def calculateCosineScore(queryVectorTemp,docVectorTemp): Calculate cosine score between two vectors.

def calculateCosine(k,queryVector): Display score and top k documents.

def select_ground_truth(): Let user to select ground truth folder.

def findCentroid(queryVec,itr): find centroid of relevant and centroid of non relevant documents.

def updateQueryVec(queryVec): Apply Rocchio algorithm and update query vector accordingly.

def plotPreRecMap(finalDocVecDic,itr): Calculate precision, MAP and Recall and plot PR-curve.

def plot_tsne(): Plot TSNE Graph for all query vectors.

Pickle:

Code for pickle is commented because my pickle is of 825mb and net connectivity is not good.

All preprocessing steps will be completed in 6 to 7mins.

HOW TO RUN MY CODE:

There are 11 code blocks.

1st and 2nd block contains all functions. Run first two blocks.

3rd block contains all header files run 3rd block.

4th block is of reading the file and preprocessing it. run 4th block

5th block is of calculating idf and building query vector. Run 5th block.

6th and 7th block is of dumping pickle and retrieving it. since it is commented you can skip these blocks.

Run 8th block then Enter query. Then enter number of docs you want to retrieve. Then enter ground truth folder. And you will get result.

9th block is of giving feedback and iterate it 4 times. Run this block and wait there is automated feedback mechanism which will give feedback at every iteration and show result accordingly. At the end you can see result of 4 iterations and at last precision-Recall curve for all 4 iterations.

10th block is of TSNE run this block to see result.

11th block plots graph for MAP. Don’t run this block until and unless you finished with all your queries. Once you finish running all above blocks(8th to 10th) to all queries in query-set then at last run this 11th block. You will see graph of MAP.