[CSE508] Information Retrieval: Assignment 3

- *Shreyash Arya (2015097)*

PRE-PROCESSING:

Pre-processing followed is similar to the previous assignment. The following steps are taken (in the mentioned sequence):

- Removing HTML tags using BeautifulSoup inbuilt module.
- Fixing the contractions (Eg. don't → do not) using contractions python library.
- Tokenization using NLTK's word tokenizer.
- Removing non-ascii characters using Unicode.
- Lowercasing the document data.
- Removing punctuations using regex (r'([^\w\s])|_+').
- Replacing numbers with words using the num2words library.
- Stemming using NLTK's Porter Stemmer.
- Removing stopwords using the NLTK's English stop-words dictionary.

Question 1

The TF-IDF based vector space document retrieval has been done similar to the previous assignment.

- In this case, the query and documents are both considered as vectors and then the cosine similarity is calculated between the two. The k having the highest cosine similarity is returned.
- The query vector is created (of the length of the size of vocabulary = 38896) which contains the frequency of the term in the query (where the term matches the vocabulary word) and others are kept zero.
- The documents are also converted to vector as the query (with the same length) and contain the TF-IDF scores for each vocabulary term and for each document.
- Finally, the cosine similarity (which is already normalized by the length of the query and the document) is measured between the created vectors and the places where the terms are not present, the similarity score becomes zero in the vector and only the relevant query terms are considered for the TF-IDF values.
- Top k (user-input) results are returned.

Rocchio Algorithm and Implementation:

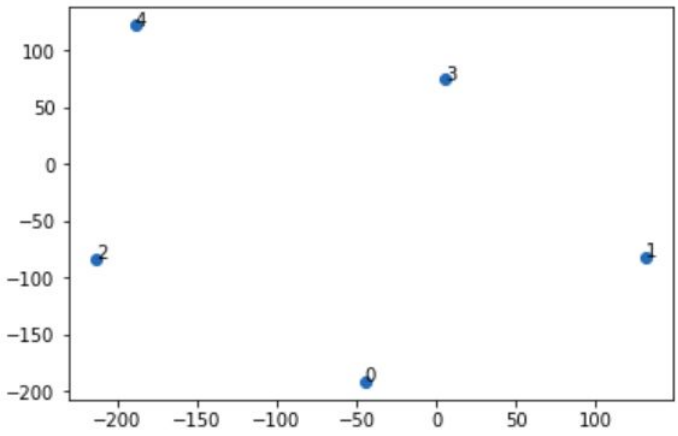
The same implementation is followed as mentioned in the class lectures:

$$\vec{q}_m = \alpha\vec{q}_0 + \beta\mu(D_r) - \gamma\mu(D_{nr})$$

where α = 0.1, β = 0.75 and γ = 0.25.

The initial query is taken for the top 10 documents and then the new query is modified according to the above-mentioned equation.

Each modified query with the original query is stored and plotted in 2D dimension using TSNE. We can see that after modifying the query, the query vector moves (annotated according to each query number) in the space according to the user feedback about the documents. The results are mentioned in the python notebook for each iteration with the final TSNE plot.

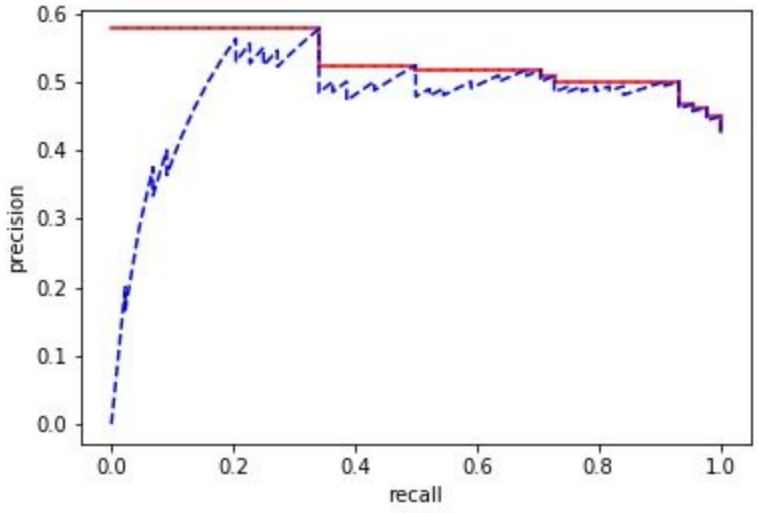


Question 2

The data is used as provided in the question by extracting all the rows with ‘qid = 4’. There in total are 103 such lines. Then the 75th feature is extracted that is the sum of TF-IDF for the whole document.

The same approach followed as in the lecture slides. The data is sorted according to the TF-IDF scores and then k ranging from 1 to 103 (for top k ranked list) was used for calculating the precision and recall for the PR Curve. Precision is calculated as the Correctly_Retrieved (TP) / Total_Retrieved (TP + FP) and Recall = Correctly_Retrieved (TP) / Total Relevant (TP + FN) . Any non zero relevance judgment value is considered to be relevant.

Finally, the calculated precision-recall values are plotted with the interpolated values as well as mentioned in the lecture slides. The maximum value to the current value is considered for interpolation. The results and final plot is present in the python notebook.



As we can see in the interpolated graph that as the more documents are retrieved, precision decreases and recall increases. Till k=4, the true positives are = 0 hence we get a increasing graph and then the normal pattern is followed.

References

Lecture Slides from class.
https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/
https://matplotlib.org/users/pyplot_tutorial.html
https://stackoverflow.com/questions/14432557/matplotlib-scatter-plot-with-different-text-at-each-data-point
https://stackoverflow.com/questions/39836953/how-to-draw-a-precision-recall-curve-with-interpolation-in-python
https://matplotlib.org/gallery/shapes_and_collections/scatter.html