ASSIGNMENT-03

README

How to Run the Code?

1. Extract the files of 20_newsgroups.
2. Must have NLTK.
3. Open Jupyter Notebook and make sure that files are available in the same folder as jupyter notebook and then start running the code from the beginning.
4. Must install pickle file named A4_Q1.pkl,Assignment4_Q1.pkl containg dictionary.

Assumptions:

1. Query terms are not repeated.
2. All queries words spellings are correct.
3. No word in present according to mobile dictionary.
4. Can only run on one query at a time.
5. Everytime new top k are retrieved irrespective of previous iteration.
6. The all documents that are not marked as relevant considered as non-relevant documents.

Preprocessing Steps:

Normalization:

- converting all text to the same case (upper or lower).
- removing punctuations

Stop Words:

- We may omit very common words such as the, a, to, of, be from unigram inverted index not from positional index.

Stemming:

- Used porter stemmer to stem the words. It is faster than lemmatization and does a good enough job of stemming related words to the same stem.

Tokenization:

- Cut character sequence into word tokens.

Num2Words:

- Convert number to words.

Single Character:

- Remove a character of length 1.

Removal of Header.

Methodology:

1. *Loaded documents from folder i.e 20_newsgroup folder.*
2. *A dictionary is created* **dic** *for storing the paths of each file.*
3. *Preprocessing of data and query is done using* **preprocess_data** *and* **preprocess_query** *methods respectively.*

4. *Made a dictionary named* **text_dic** *containing docId and tf for each word in a corpus.*
5. *Pickle a text_dic and stored in Assignment4_Q1.pkl.*
6. **tf_dict** *is maintained having tf values for each term in a corpus.*
7. **idf** *dictionary is maintained having idf values for each term in a corpus.*
8. **tf_idf** *dictionary is maintained having tf*idf values for each term in a corpus.*
9. *User enter the query and* ***K****. Then is query is preprocessed. It's tf _idf values for each term are computed and stored in dictionary named* **query_tf_idf**.
10. *For each query term it's documents are fetched and then took the union of all those documents using* **union** *method.*
11. *For each query, a query vector is generated using* **queryVector** *method of length equal to vocabulary.*
12. *Created a dictionary named* **docs_dic** *contating documents vector having tf_idf values for each vocab term if present otherwise 0.*
13. *Pickle a doc_dic and stored in A4_Q1.pkl.*
14. *Cosine similarity between each document and query vector is found using* **cosine_sim** *method.*
15. *Found top k and mapping of documents with it's actual name using* **mapp_doc** *function.*
16. *User feedback is taken using* **user_feedback** *function returing relevant and nonrelevant documents marked by user.*
17. *Documents are marked star using* **top_retrieved** *function.*
18. **calc_centroid** *is used for calculating centroid of relevant as well as nonrelevant documents.*
19. *Modified query is computed using* **modified_query** *method.*
20. *Found precision and recall and plotted it's curve in each iteration.*
21. *MAP is computed using* ***compute_MAP*** *function.*
22. *TSNE plot is plotted using function* **plot_TSNE**.

References:

- *https://stackoverflow.com/*
- *https://github.com/williamscott701/Information-Retrieval/tree/master/Assignment%203*