COL764: Web and Information Retrieval

Assignment – 1

- by Karan Tanwar (2017CS50411)

Algorithmic Details

There are three parts to the end-to-end pipeline:

- **Parsing and cleaning the data** from the files in a directory: Here I have created a panda DataFrame in memory with each row containing clean text.
- For POS tagging, I have grouped the tags into a special syntax: person_donal_trump for <PERSON>Donald</PERSON> <PERSON>Trump</PERSON>, loc_iowa for <LOCATION>Iowa</LOCATION>. This made it easier to find tfidf of queries in later steps.
- In pre-processing library created, I decided to not stem the training data and stem the queries brutally to retrieve more relevant documents. This allowed me to create posting list faster and give me a very high nDCG of 0.51 but F1 score was dropped to 0.222. But I was not removing much stop words in the pre-processing. As an implementational choice, I increased the size of set of stop words to remove and stem the training data too (using Snowball Stemmer), as it increased the F1 score to 0.38 and reduced the size of posting list drastically (from 40MB to 24 MB in 7z format).
- To **create posting list**, posting_list_terms_frequency_in_documents() returns a dictionary for a document in {word: frequency} format. Then the program iterates over all documents and merge them into one big dictionary in {docid: {word: frequency}} format. This is the posting list. Using dictionary of dictionaries instead of dictionary of lists, really helped in reducing construction time by 17%.
- **To retrieve best k documents for specific query**, I again used the same pre-processing and calculated the tf-idf for each word in query, in all documents. I stemmed the queries (not lemmatized as it harms the word structure, and I don't want that in my implementaion) and treated it as a prefix search. Only stemming testing data seems to be reasonable if training data is large and then just treat the queries as a prefix search. For example, in posting list, there is a word 'celebrations', and in query I will pass 'celebr*'.
- Tagged terms are kept separate for queries because there were instances where only-surnames were also present. So, query=['person_donald*','person_trump*'] is not doing any harm to semantics of query and yet can search for only surnames.
- They are reduced by taking a sum of all tf-idf for a particular docid. The best min (k, num of docs containing the query words) are retrieved.
- There are functionalities for query search of tagged queries such as P:donald_trump.

Results

nDCG Score: 0.3542

F1 score: 0.3822

Size of posting list: 24 MB

Time taken for creating posting list: 367 sec at an average of 4.1 GHz CPU’s clock speed.