Information Retrieval

Assignment 2

Prabhat Kumar
MT17036

Assumptions:

- IDF value of token the query has been considered as 1.
- Intersection of Posting List has not been considered as a factor in Ranking of Documents as it was often the case that the importance of Title Matching was lost, when giving weightage to those documents where all token in the query was found. As results with Title, should be given preference, Intersection of Posting list was not performed. Also, after intersection, searching results of query was found to take more time to execute.
- Spell check has been performed according to the library Autocorrect.
- Index.html has been modified for ease of extraction of File Titles.
- TF_IDF matrix has been created by the formula

$$tf - idf\ score = (1 + \log_{10} TF) \times (\log_{10}(\frac{N}{DF}))$$

Where,

TF = Term Frequency in the Document
N = Total Number of Documents
DF = Document Frequency of Term

Methodology:

- Each document is preprocessed by tokenization, followed by removal of punctuations. Further more, Stopwords are removed with respect to NLTK Library.
- Documents are converted into lower case to create uniformity among all.
- Numerical Tokens such as 99, 1990 are converted into respective Word representation.
- Document titles are then extracted using *title_extractor.py* which extracts title and word count from *index.html.*
- TF-IDF matrix is created and saved as *tf_dict.sav.*
- TF-IDF and Cosine Query Matching has been done according to its respective methods.

Spell Correction

Spell Correction have been performed by using the *Autocorrect* library.

Numerical Queries

As mentioned before, each numerical token has been converted into its verbal form by using function *number_to_words()* from *Inflect* library.

Title Extraction and Weightage

- Title extraction has been done by title_extractor.py and have been saved as dictionary with file name as the key and value as a list containing 2 elements.
- First, being the word count of the file as per index.html, followed by a list of tokens of Title of the file value is associated with.

- For each matching token of a title's token list, TF value of the particular word with respect to File is increased by value equal to sqrt(word count)
- Increasing TF by sqrt(word count) makes sure the that documents with less occurrences of words, don't get added benefit as compared to one with large documents as this can be the case with static increase in TF count.

Caching

- Caching of Queries has been done with Least Recently Used algorithm.
- This implies, the cache contains the results of last 20 queries executed irrespective of hit and miss of cache.

References

- NLTK, http://www.nltk.org
- AutoCorrect, https://pypi.python.org/pypi/autocorrect/0.2.0
- Inflect, https://pypi.python.org/pypi/inflect