CS F469: Information Retrieval

Open-Source Search Engine

1. Ujjwal Raizada	2017A7PS1398H
2. Satyam Mani	2017A7PS0277H
3. Daksh Yashlaha	2017A7PS0218H
4. Pranjal Gupta	2017A7PS0124H

Open-Source Search Engine

Design Document

This search engine indexes all the documentation in open source projects on GitHub and then uses inverted Index to store inverted document frequency. A compound statement is by default processed as AND and intersection is performed on the individual results to generate the final result.

- Python3 is the language of choice because of rich in-built data structures and libraries for data pre-processing.
- NLTK library is used for data pre-processing and tokenization.
- SortedSet is used to store inverted document frequency to reduce search complexity to factor of log(n).
- pickle library is used to persistently store the index.

Component:

1) **Pre-processing:** Firstly the document is converted to a string stream, then all words are converted to lower-case for consistency, then NLTK library is used to remove stop words then stem all the remaining words. Then frequency ratio of each word is calculated (number of occurence / total number of words).

2) **Index:** Index is a dict of SortedSet of pairs:

```
[word-1]: [(freq_ratio, doc_name), ....(freq_ratio, doc_name)]  # SortedSet
[word-2]: [(freq_ratio, doc_name), ....(freq_ratio, doc_name)]  # SortedSet
.
.
.
[word-n]: [(freq_ratio, doc_name), ....(freq_ratio, doc_name)]  # SortedSet
```

   - Dict data structure is used, because it searches for a given word in O(1) amortized complexity and is very easy to use.
   - SortedSet is used because each document should exist only once in each word and always sorted property makes is easier to get the best results.
   - Index is stored persistently using pickle to save pre-processing time each time the search engine is loaded.

3) **Search Query:** A compound query is first split into words and then index is searched for each word, then the intersection is calculated for the resultset of each word in the query. The intersection is the final answer of the query.

   Example: “programming language” is first split into [“programming”, “language”] and then resultset of each of these words is intersected to get the final resultset.

4) **Dynamic Index:** Index is automatically updated whenever a new document is added in the “corpus/” directory. A list is maintained of the already processed documents and is used to check for any new documents.

Work remaining:

1) Time spent to resolve a single query. (benchmarks)
2) Better interface for user to search.
3) Document scraper/ Github Crawler
4) Additional indexes for important compound statements. (eg: “machine learning”, “web development” etc.)

Project is available on github: https://github.com/ujjwal-raizada/open-source-search