Model Search Engine

CS244: System Programming Lab

Nitesh Jindal (160101084),
Ameya Daigavane (160101082),
Abhishek Suryavanshi (160101009)

18th March, 2018

1 The Search Engine

- Rapid querying through pre-processing of files (takes approximately 3 seconds to process 10 MB of text with around 150,000 lines).
- Ranks files in order of calculated Term-Frequency and Inverse-Document-Frequency scores for each search.
- Extended line context of word match shown, along with line number.

2 Algorithm and Data Structures

- Indexing algorithm generates a list of line numbers for each word - takes $O(n)$ time where n is the total number of words in all of the files.
- For each word, we create an inverse-index record, mapping this word to a set of file names with line indexes.
- Hashtables (as Python dictionaries) allow us to access individual word records in $O(1)$ time.
- For the ranking procedure, we find the following quantities:
  - Term Frequency in each document, TF

$$TF(\text{word}) = \frac{\text{frequency(word)}}{\sum\limits_{w} \text{frequency}(w)}$$

  - Inverse Document Frequency, IDF

$$IDF(\text{word}) = \log \frac{\text{number of documents}}{|\{\text{document} \mid \text{word} \in \text{document}\}|}$$

- For single word queries, we rank the files according to term frequencies. When given a phrase with multiple words, we calculate a score for each file containing all the words by finding the dot-product of the TF and IDF vectors (made from combining the scores for each word), and sort them in decreasing value of this quantity.

- To summarize, we take $O(|\text{query}| * D)$ worst-case time for each query, where D is the total number of documents.