Information Retrieval

Assignment 2: Inverted Index

Tapan Sahni `CS13B1030`
Hrishikesh Vaidya `CS13B1035`
Akilesh B `CS13B1042`

September 10, 2016

1 Implementation Details

The crawled data from Assignment 1 consists of 1000 documents each containing URL, Title, Date, Meta-Keywords, Content fields. Our code followed Google Python Style Guide and was checked by running *pylint* over your code.

1.1 Section-1

- *Handling date* - The month attribute in our corpus can be one of the following : 'jan', 'feb', 'mar', 'apr', 'may', 'june', 'july', 'aug', 'sep', 'oct', 'nov', 'dec', 'january', 'february', 'march', 'april', 'may', 'june', 'july', 'august', 'september', 'october', 'november', 'december'. We created a list with the above mentioned possibilities.
  Helper functions $expressionfordateb()$ and $expressionfordatea()$ are used for checking if previous and next tokens are digits or not.The previous token has to have starting two character as digits.This would take care of the dates like 3rd July 2016. If they return $true$, we pop the already inserted word from the inverted index using $popelement()$ and construct a single token for date using $constructdate\_expression()$.

- Appropriate regular expression matching functions are used to get title, meta-keywords and content which is used for index creation. The functions $getatitle()$, $getmetakeywords()$ and $getcontent()$ are used for performing these tasks respectively.

- Stopwords defined here are removed. The function $getstopwords()$, performs this task. Index $i2$ is then constructed.

- For stemming, we use $PorterStemmer()$, Index $i3$ is then constructed. The function $readcorpus()$ constructs indexes $i1$ to $i3$.

- We construct Index $i4$, from $i3$ after removing the least frequent terms (those occurring in less than 2% of the documents).

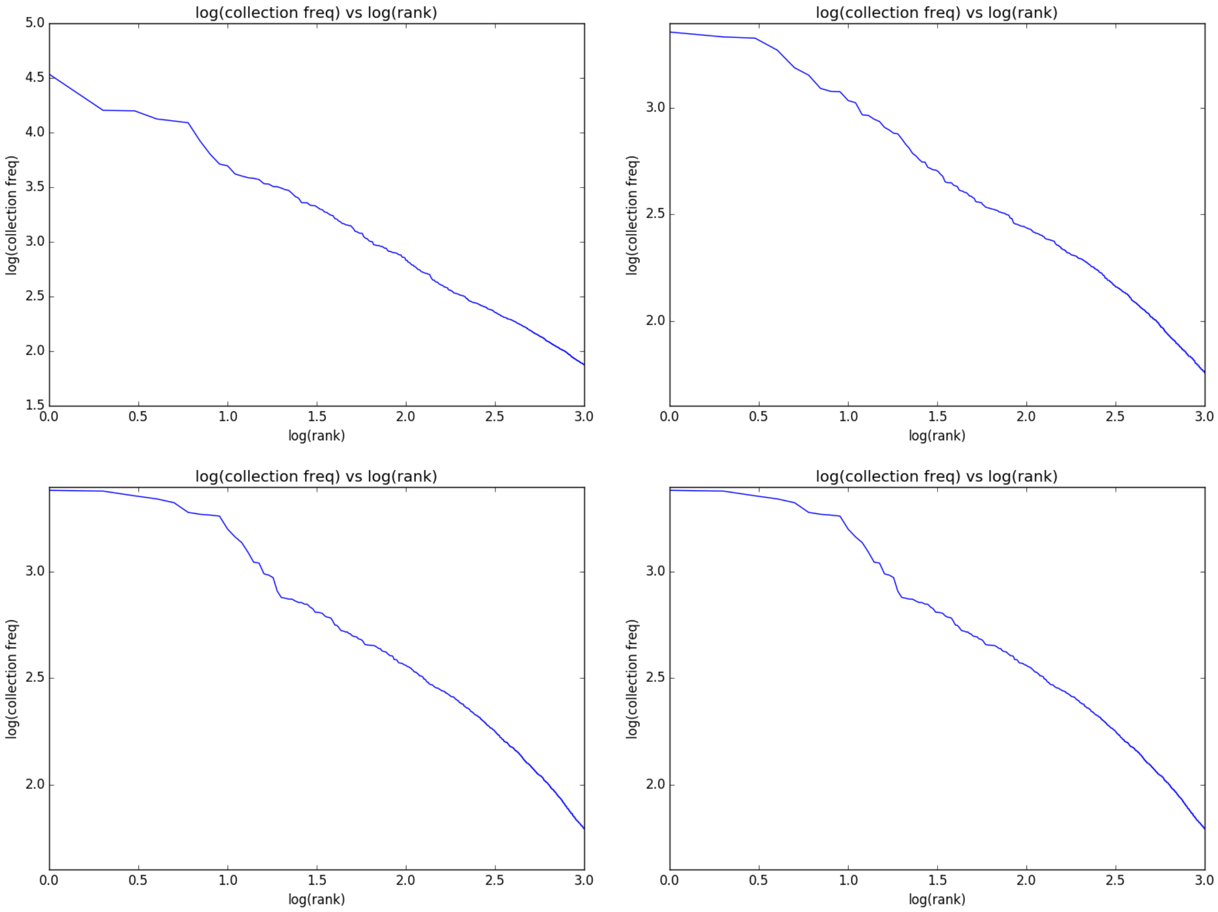
- The function $calculate()$ prints the number of terms, maximum length of postings list, minimum length of postings list, average length of postings list, size of the file that stores the inverted index, for each of the indexes $i1$ to $i4$.

1.2 Section-2

- The functions $get\_topk()$, $get\_medk()$ and $get\_leastk()$ gets the most frequent K words, median K words and least frequent K words respectively. For each of these, we print postings list size for each of the above K words and also find average gap between documents in the postings list, for each of these words.

1.3 Section-3

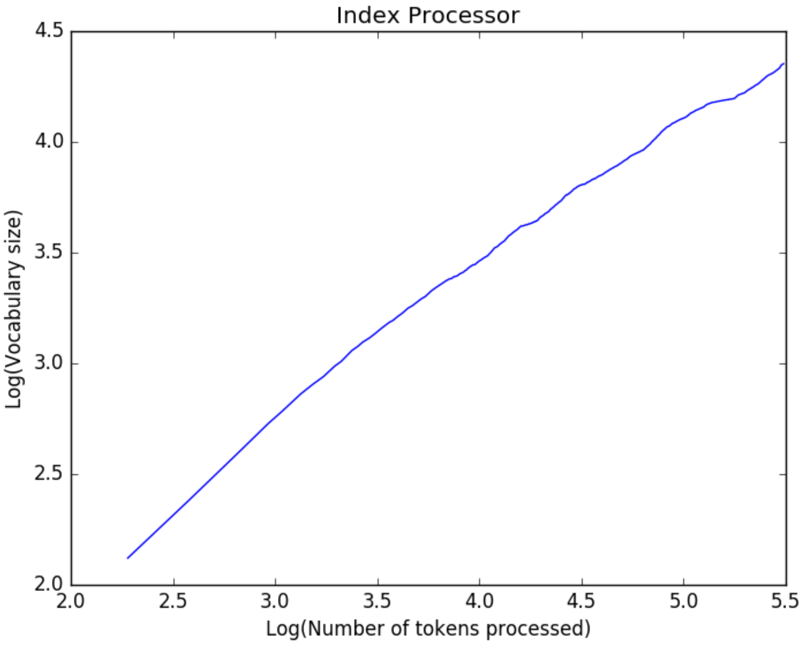
- The function $get\_highestcf()$ takes the top 1000 terms with highest frequency in the collection and plots a graph for each index $i1$ to $i4$ with x-axis: $log(i)$ for ith most frequent term, and y-axis: log of collection frequency of the term.



- Observations: It's a decreasing graph in all four cases. The value of log(collection freq) is highest in $i1$ (around 4.5 when rank is 1), its around 3.5 in other cases. The graphs for $i3$ and $i4$ are the same because we are just removing the least frequent terms (those occurring in less than 2% of the documents).

1.4 Section-4

- The function $indexprocessor()$ constructs two list which maintain the total tokens processed and the vocabulary size after every document. The graph of log(`vocabulary size`) is plotted against log(`number of tokens processed`)



- By looking at the graph we can say that the function is almost linear.The slope of the function decreases with x. From this we conclude that as the number of tokens processed increases the rate of increase in vocabulary size decreases.