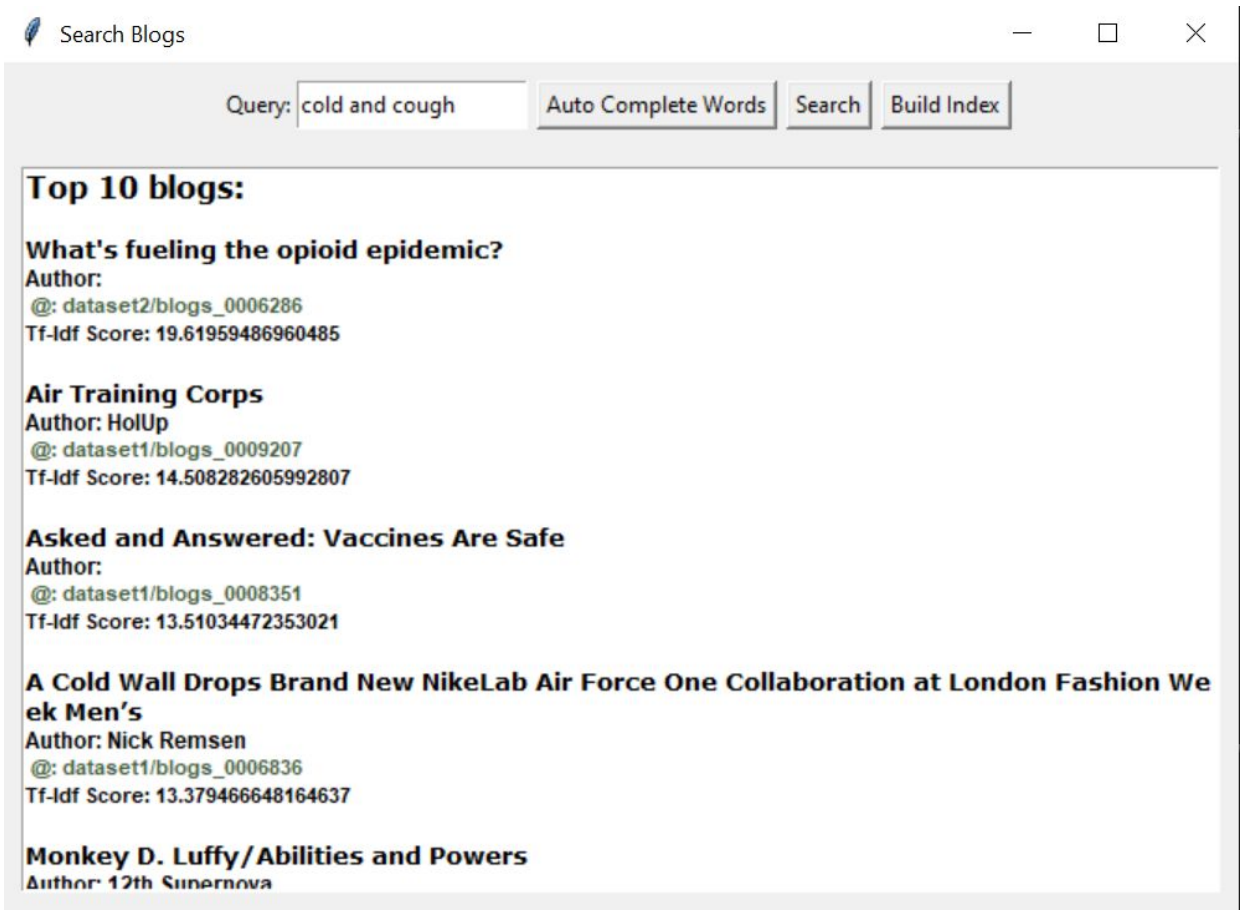


Search Engine

Information Retrieval

S20180010191

Manasa Pakala

Overview

A search engine for querying blogs about all topics from health and lifestyle to entertainment.

Dataset:

I used the popular blogs dataset from webhose.io.

Link: [Dataset Link]

The blogs are structured documents in json format. It has the attributes text, author, title, entities etc, amongst which I indexed and the blog content in text, and retrieved the title and author for the search results.

Tasks

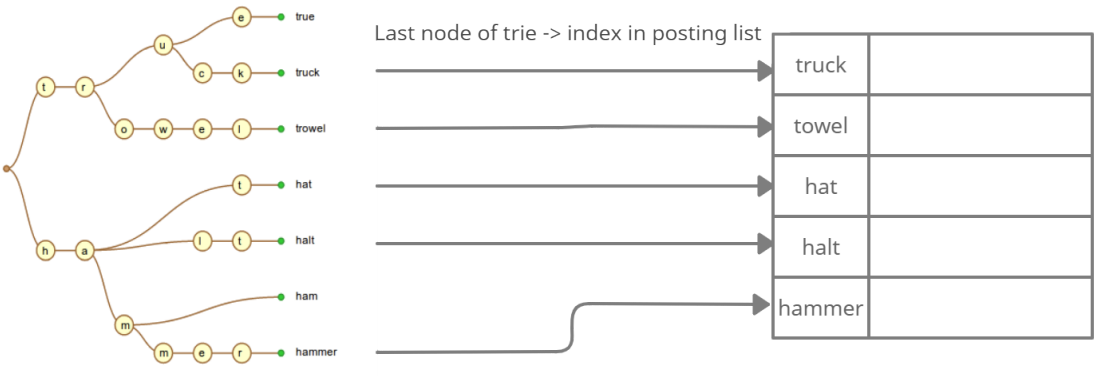
I. Preprocessing:

The words in each blog read were first tokenized, stemmed and filtered before proceeding with building the index and trie.

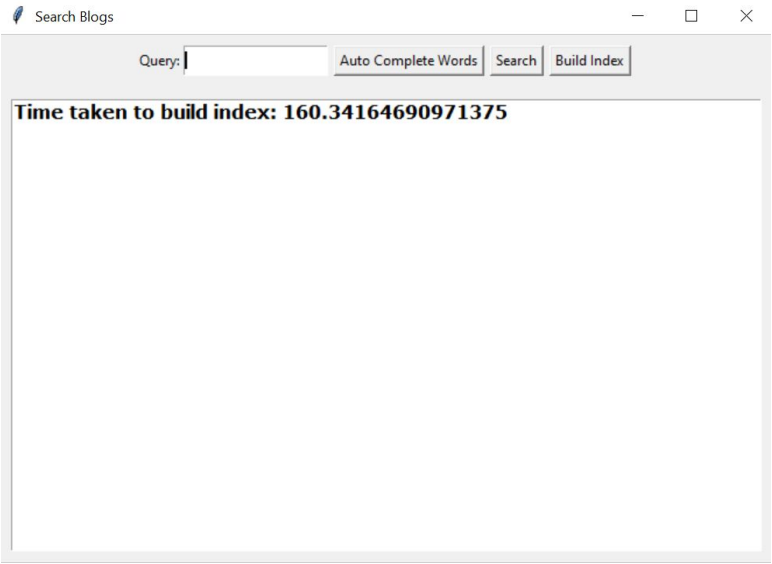
II. Building and Index:

I used a block based algorithm to build the inverted indexing.

- About 20k docs are retrieved first, these blogs on being preprocessed were added to a trie.
- At the last node of each word in the trie, the indices to the posting list were stored. This was done for faster search O(length of word) in the future and for lesser space consumption.



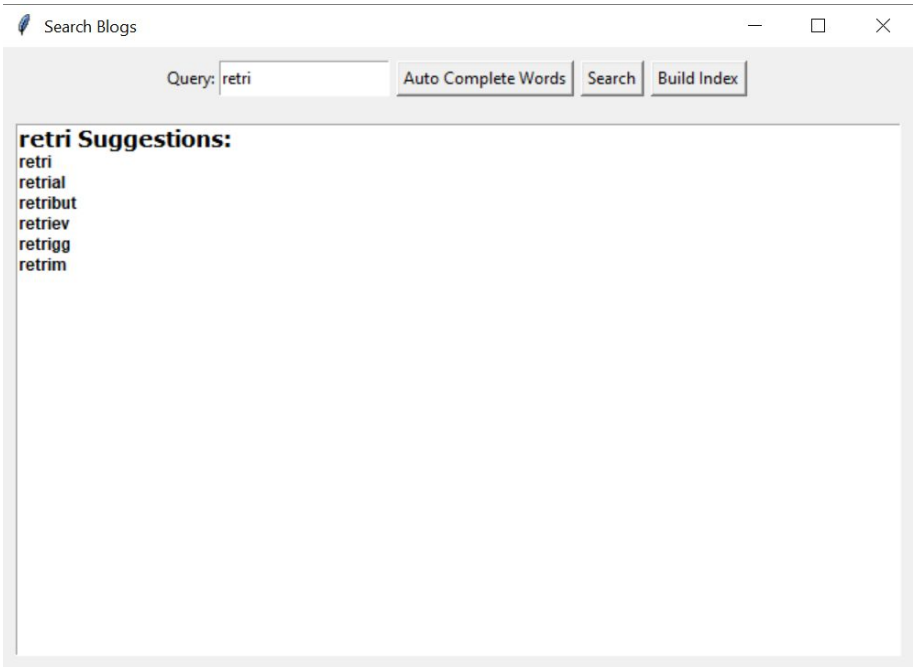
- Once the words in these 20k documents were added to the trie, the posting list for that collection was cleared from the memory and written to a disk output file : output.txt
- The next set of 20k documents were then preprocessed and added to the trie in a similar fashion.
- Once this was done the previous posting list was read from the memory and both of the posting lists were merged.
- This process went on till all the docs were read and their words were indexed and written to a file.



- Further to avoid reading that big a file, outputs.txt over and over again for every implementation of the algorithm, the trie with the indices of posting lists was also written to a disk file : trie.txt
- **For further searches, this trie was loaded from trie.txt and only the line associated with the index in outputs.txt was read and parsed when required.**

III. Autocomplete Query:

- Using a recursive algorithm, I implemented a basic auto-complete suggestion for finding tokens that are in the blogs.



IV. Search:

- ➢ The query was first preprocessed, by splitting and stemming.
- ➢ The preprocessed tokens of the query were then used for search.
- ➢ Nodes of each of the tokens in the query were traversed in the trie data structure(loaded from trie.txt) to see if that word exists in the blogs.
- ➢ If the word does exist, the index of the posting associated with the word was first retrieved.
- ➢ This index was used to get the postings list from outputs.txt

```
No. of tokens in trie:  107572
['cold', 'cough']
IDF for the term  cold  :  3.608
IDF for the term  cough  :  6.438
No. of blogs found:  567
```

V. Ranked Retrieval:

- ➢ The idf score for each token of the query was first calculated.

```
['cold', 'remedi']
IDF for the term  cold  :  3.608
IDF for the term  remedi  :  5.573
No. of blogs found:  604
```

- Then the posting lists returned from the search algorithm were parsed and used to find the term frequency for each term.
- This term frequency was used to find the tf-idf score for each query token and document pair.
- The top 10 ranked documents with their titles, authors and references were returned and displayed.