IR Assignment
CS F469

KAUSTUBH BHANJ	2019A7PS0009H
PRATHMESH SRIVASTAVA	2019A7PS1322H
SATHVIK BHASKARPANDIT	2019A7PS1200H

HITS Algorithm Implementation

Hyperlink-Induced Topic Search (HITS; also known as hubs and authorities) is a link analysis algorithm that rates Web pages, developed by Jon Kleinberg. The idea behind Hubs and Authorities stemmed from a particular insight into the creation of web pages when the Internet was originally forming; that is, certain web pages, known as hubs, served as large directories that were not actually authoritative in the information that they held, but were used as compilations of a broad catalog of information that led users direct to other authoritative pages. In other words, a good hub represents a page that pointed to many other pages, while a good authority represents a page that is linked by many different hubs.

Performance

PreProcessing:

For preprocessing we are removing stop words and performing stemming. Both the processes take linear time.

Indexing:

We are indexing the dataset by visiting each token and then firstly inserting each word in our PermutermIndex and then in our InvertedIndex.

- **Insertion in InvertedIndex:**
  We are using a hashmap where each value of the hashmap represents a posting list. Each insertion can at max take O(N) time where N is the size of the posting list for a token.

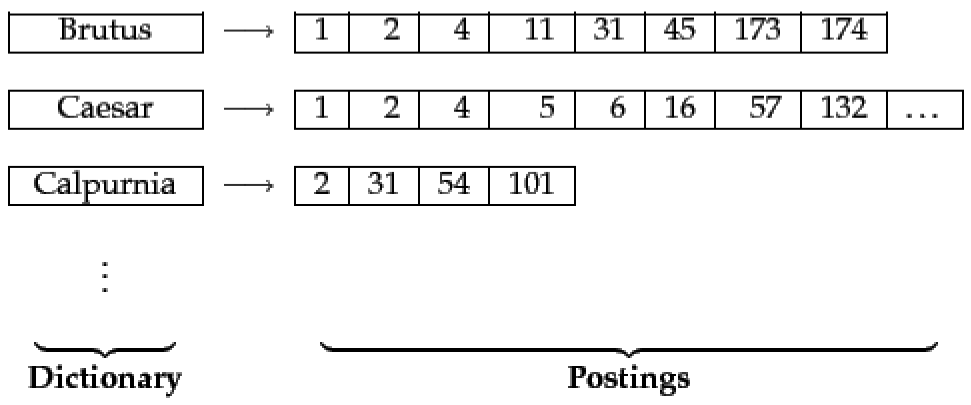
Searching/Retrieval:

In case of a hit, i.e not wildcard and correct spelling, the complexity to get a posting list is constant time. The matching documents are classified as **Root Set**. From the root set, all the documents which either have an outgoing on an incoming edge from the root set are classified as the **Base set**. The HITS algorithm is applied on the **Base Set**.

Index Structure

InvertedIndex:

The inverted index is implemented using python dicts and python lists. Each value in the dict represents a posting list which is in the form of a list.



▶ **Figure 1.2** The two parts of an inverted index. The dictionary is commonly kept in memory, with pointers to each postings list, which is stored on disk.

The posting list elements are in a sorted order.

Procedure to run

The project root contains main.py, requirements.txt and config.py. Main.py is the entry point of the application. Requirements.txt has the required modules to run the application. And config.py has some global variables.

Install the dependencies

```
pip install -r requirements.txt
```

Update the config.py file

Open the file in a text editor. Update the path of web graph gpickle file.

Run the algorithm

```
python main.py -query <search_term>
```

Example

```
python main.py -query pension
```

Sample output

```
Root Set:  [0]
Base Set:  [0, 3, 10, 61, 75, 87]
Auth Hub Scores:
  DocID    Auth Score    Hub Score
-------  ------------  -----------
      0      0.320012     0.204815
      3      0.21121      0
     10      0            0.320012
     61      0.21121      0
     75      0.128785     0.320012
     87      0.128785     0.155162

Top auth scores:
  DocID    Auth Score
-------  ------------
      0      0.320012
      3      0.21121
     61      0.21121

Top hub scores:
  DocID    Hub Score
-------  -----------
     10     0.320012
     75     0.320012
      0     0.204815
Runtime:  178.65228652954102 milli secs
```

Runtime analysis for a few search terms

Query Term	**Runtime in milli seconds**
pension	3
sports	15
ibm	4.9
tuesday	10