Twitter Keyword Search

Shiza Ali and Mohammad Hammas Saeed

Overview

- Extract tweets using web crawler
- Word segmentation
- Build inverted index to search the word
- Store the inverted list using relevant data structure such as Hash Tables
- Input query
- Sort the similar tweets, and select the top 10 most similar

Data Structures and Algorithms Used

Inverted Index

- To gain the speed benefits of indexing at retrieval time, we have to build the index in advance. The major steps in this are:
  - Collect the tweets to be indexed
  - Tokenize the text
  - Turn each document into a list of tokens
  - Index the documents that each term occurs in by creating an inverted index, consisting of a dictionary and postings.

Inverted Index

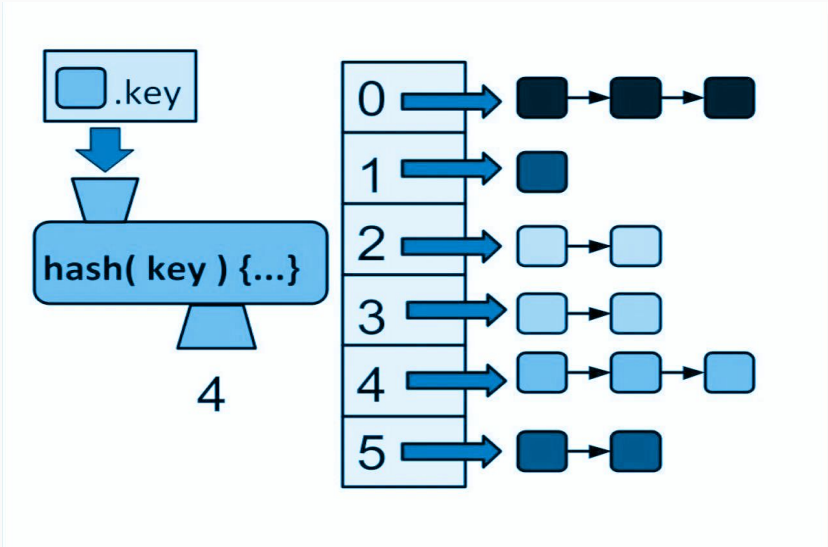
ID	Tweet
tw1	I am Amazing!
tw2	Amazing food.
tw3	My cat ate food.

Preprocessing

Keywords	IDs
I	tw1
am	tw1
amazing	tw1, tw2
food	tw2, tw3
My	tw3
cat	tw3
ate	tw3

Hash Map

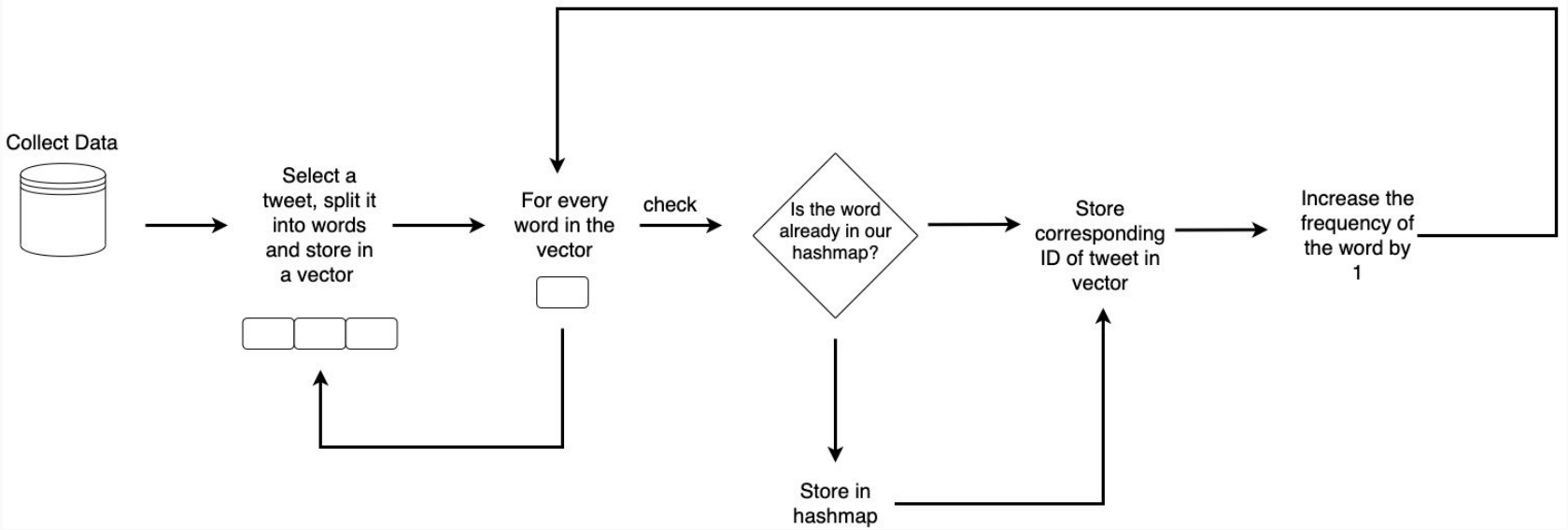
- Data structure that maps keys to values.
- The advantage of this searching method is its efficiency to handle vast amount of data items in a given collection.



map vs unordered_map in C++

Differences between map and unordered_map in C++:

	map	**unordered_map**
Ordering	Ascending Order	No order
Implementation	Self balancing BST	Hash Table
Searching	O(log(n))	O(1) Average, O(n) Worst Case
Insertion	O(log(n)) + rebalance	O(1) Average, O(n) Worst Case
Deletion	O(log(n)) + rebalance	O(1) Average, O(n) Worst Case



Creating Index

Data structure for Ranking the Queries

- Priority Queues/ Heaps
  - Every item has a priority associated with it.
  - high priority is dequeued before an element with low priority.
    - same priority are served according to order in the queue

Ranking

Keywords	Vector<tweet, occurrence>
I	<tw1, 1>
am	<tw1, 1>
amazing	<(tw1, 1),(tw2, 1)>
food	<(tw2, 1),(tw3, 1)>
My	<tw3, 1>
cat	<tw3, 1>
ate	<tw3, 1>

Continued

- Let’s pick the keyword “Amazing”
- The map returns a vector:
    - <(tw1, 1), (tw2, 1)>
- Heap Sort is performed on this vector using the number of occurrences
- The Top-10 from the sorted list is shown to the user

Demo

Performance Analysis

Time Analysis

Data structure	Set Up Index	Search	Sort Results
map + vector	~20s	~0.6s	~0.3s
unordered_map + vector	~5s	~0.3s	~0.3s

Note: Analysis performed on a huge dataset ~ 1 million tweets

Questions?