University of California, Riverside

CS 242 : Information Retrieval & Web Search

Project Report- Part A

Lokesh Koppaka, Abhilash Sunkam, Vishal Lella
{lkopp001, asunk001, vlell001}@ucr.edu

Project Overview:

The Internet plays a major role in providing a substantial amount of information about all sectors. A huge amount of data is available on the Internet which can be used to get the required information if deployed properly. For every other aspect in day to day to life people rely on this information. Web information can be of various sort like videos, pictures or textual info like audits, remarks and so forth. Learning and gaining knowledge from the web, can be utilized to get profitable productivity thus prospering the exchange.

We see that we have enough information about technologies, tech conferences and various other things related to tech. We use the twitter data to make a search engine, where the user can search for any tech term and get to know what's currently happening around him. In the figure given below (Figure 1) you can see the workflow we have followed in doing this project.

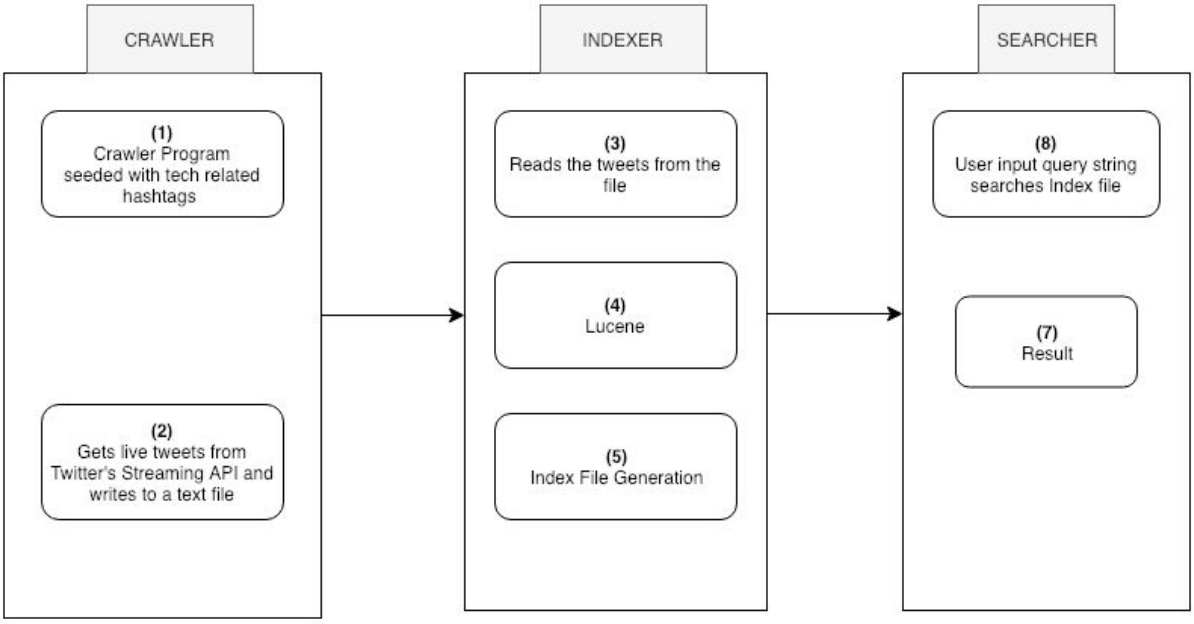


Fig 1: Flowchart showing the workflow of the project

Crawling:

We wanted to build a tech-based search for this project. To build this tech search, we needed lots of tweets related to the latest technologies, tech conferences etc. The first question which arises was how to extract tweets from twitter. We had two options to choose i.e we can either use Twitter's REST API or Twitter's Streaming API.

Since the search engine, we are building requires the latest information regarding tech conferences or events happening in near future, we needed to get the latest tweets. Therefore, we have decided to go forward with Twitter's Streaming API. Twitter's REST API is more useful if we wanted to do some analysis or search on historical data. Twitter's Streaming API gives us high volumes of live tweets data as per our request until we want to stop receiving.

Our choice of language for crawling the tweets is python. In order to fetch the live tweets, we have used tweepy library. Tweepy makes it easier to use the streaming API as it handles connection, authentication, sessions etc.

We seeded a lot of hashtags related to tech companies, new technologies, conferences and ran the crawler. Some of the hashtags included are "#ArtificalIntelligence", "#MachineLearning", "#BigData", "#Google", "#Microsoft", "#IBM", "#Saleforce". The challenging thing we faced was trying to get those tweets faster. We then implemented multiprocessing in our program, as it utilizes all the processors in the given machine and tries to get the data much faster than earlier. It runs on both Windows and Unix. It gradually improved performance.
We store the incoming tweets in a batch of files having a maximum size of 100MB.

How to run the crawler:

- python tweets_crawl.py doc-path N
- Doc-path is the argument given to tell where the tweets should be stored
- N is the number of tweets you want to get.

Indexing:

Once the tweets are crawled it is essential for performing indexing for quick search of tweets. Across multiple indexing architectures, today's information retrieval/Search Engines uses the concept of inverted indexes.In such architectures, the indexes are maintained as pairs with the key being the word/term and value being the document ID along with some ranking parameters like term frequency. Inverted index architecture with Lucene is used for indexing in the project.

Lucene is a high-performance text search engine which provided essential APIs to perform indexing and search over huge text data.

Design Choices:

- The following are the list of tweet attributes considered

TWEET ATTRIBUTE	DESCRIPTION
HashTag	Tweet HashTag
Tweet	The text of the Tweet
Title	URL Title
createdAt	Created Timestamp
Coordinates	Tweet Location
URL	Expanded URL of the Tweet

- Tweet text and Hashtags are the only attributes indexed as they hold the keywords and logical relevance. It is also important to consider Hashtags for indexing since most of the hashtag may or may not resemble keywords and can contain abbreviation which is essential for searching tweets. For Example #imcConf2019,#lol, #instamood etc..

- Apart from the above the rest of the attributes are stored. These attributes are later used in project part B for the purpose of geotagging.

- Stopwords such as [but, be, with, such, then, for, no, will, not, are, and, their, if, this, on, into, a, or, there, in, that, they, was, is, it, an, the, as, at, these, by, to, of] are not considered as such stopwords might affect the overall ranking of the documents. The standard analyzer of Lucene is used to prevent stopwords while index generation.

- Search on the tweets is performed with case insensitivity. This is essential as the indexer might consider the same word with different Case as different words.

Implementation Details:

Indexing and Searching are implemented in Java using Lucene. The implementation holds the following Java classes

1. **TweetIndexer** - holds the functionality to read the crawled tweets and construct the index
2. **TweetSeacher** - holds the functionality to read the query and output top 100 search results along with the rank and score
3. **TwitterSearchEngine** - Main java class which holds users preference such as the prompting the user for the query string, initial indexing and initiating tweet search.

How to run Indexer and Searcher?

1. Make sure Java is installed in the machine.
2. Unzip the File "Project1_PartA.zip" provided.
3. Use the following execution command format to run TweetSearchEngine.jar :

Tweet Indexer

```
Java -jar TweetSearchEngine.jar [indexFilePath] [dataFilePath] [option]
  Example : Java -jar TweetSearchEngine.jar "c:/InformationRetrieval/indexedFiles" "c:/InformationRetrieval/dataFiles" "1"
```

Tweet Searcher

```
Java -jar TweetSearchEngine.jar [indexFilePath] [option]
  Example : Java -jar TweetSearchEngine.jar "c:/InformationRetrieval/indexedFiles"  "2"
```

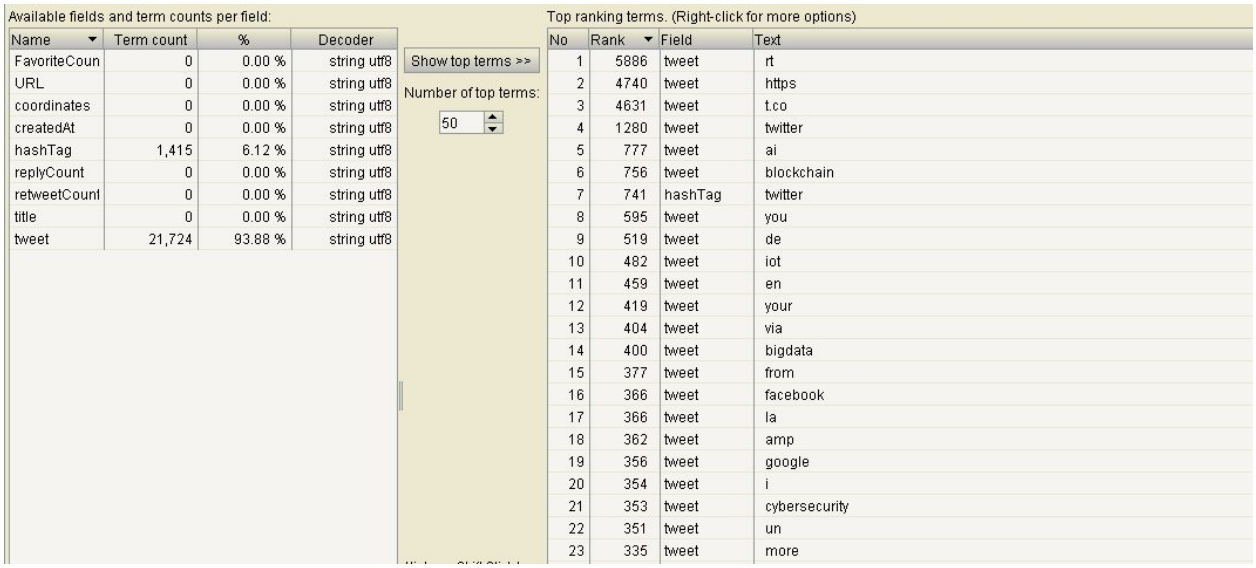
Command Line arguments description

ARGUMENT	DESCRIPTION
[indexFilePath]	For Option 1 - The full directory path where the constructed indexing file need to be stored For Option 2 - The full directory path of the generated indexFile, this is used by TwitterSearchEngine to search tweets
[dataFilePath]	The full directory path of the data files location
[option]	1 - Construct Indexing 2 - Search for Tweet

Output:

1. Tweet Indexing

```
D:\UCR\UCR_SecondQuater_Winter2019\InformationRetrieval>Java -jar TweetSearchEngine.jar "D:\UCR\UCR_SecondQuater_Winter2019\InformationRetrieval\Project\IndexedFiles" "D:\U
CR\UCR_SecondQuater_Winter2019\InformationRetrieval\Project\InputData" "1"
Started Indexing!!!
Done with Indexing!!!
Time Taken = 597millSec
```



Available fields and term counts per field:

Name	Term count	%	Decoder
FavoriteCoun	0	0.00 %	string utf8
URL	0	0.00 %	string utf8
coordinates	0	0.00 %	string utf8
createdAt	0	0.00 %	string utf8
hashTag	1,415	6.12 %	string utf8
replyCount	0	0.00 %	string utf8
retweetCount	0	0.00 %	string utf8
title	0	0.00 %	string utf8
tweet	21,724	93.88 %	string utf8

Show top terms >>

Number of top terms: 50

Top ranking terms. (Right-click for more options)

No	Rank	Field	Text
1	5886	tweet	rt
2	4740	tweet	https
3	4631	tweet	t.co
4	1280	tweet	twitter
5	777	tweet	ai
6	756	tweet	blockchain
7	741	hashTag	twitter
8	595	tweet	you
9	519	tweet	de
10	482	tweet	iot
11	459	tweet	en
12	419	tweet	your
13	404	tweet	via
14	400	tweet	bigdata
15	377	tweet	from
16	366	tweet	facebook
17	366	tweet	la
18	362	tweet	amp
19	356	tweet	google
20	354	tweet	i
21	353	tweet	cybersecurity
22	351	tweet	un
23	335	tweet	more

2. Tweet Searching

```
D:\UCR\UCR_SecondQuater_Winter2019\InformationRetrieval>Java -jar TweetSearchEngine.jar "D:\UCR\UCR_SecondQuater_Winter2019\InformationRetrieval\Project\IndexedFiles" "2"
******** Welcome Tech Tweet Seacher**********
Enter the search query =
MachineLearning
------------------<== Query Top Results ==>--------------------
Title = None
HashTag = #MachineLearning
Tweet = Book about #MachineLearning
createdAt = un Feb 10 04:46:37 +0000 2019
URL = None
<== Score and Rank Info ==>
Rank = 1
Score = 2.2642565
**********************************************************************
Title = None
HashTag = #Robotic
Tweet = Influencers #ArtificialIntelligence #MachineLearning #Robotic
createdAt = un Feb 10 04:47:57 +0000 2019
URL = None
<== Score and Rank Info ==>
Rank = 2
Score = 2.2642565
**********************************************************************
Title = None
HashTag = #MachineLearning
Tweet = Artificial Intelligence Timeline #ArtificialInteligence #MachineLearning
createdAt = un Feb 10 04:58:53 +0000 2019
URL = None
<== Score and Rank Info ==>
Rank = 3
Score = 1.9812244
**********************************************************************
Title = La aplicacin de machine learning contra el fraude llega a Marruecos Finanzas CIO
HashTag = #fraude
Tweet = RT @GrupoAIS: #Machinelearning #fraude https://t.co/tlIAUPIaiT
createdAt = un Feb 10 02:31:14 +0000 2019
URL = https://www.ciospain.es/finanzas/la-aplicacion-de-machine-learning-contra-el-fraude-llega-a-marruecos
<== Score and Rank Info ==>
Rank = 4
Score = 1.6981924
**********************************************************************
```

INDIVIDUAL CONTRIBUTION:

Lokesh Koppaka

1. Primary Contributor for Tweet indexing using Lucene
2. Performed necessary analysis and design for tweet indexing
3. Implemented Indexing of Tweets using Lucene in Java
4. Implemented Query Search logic to retrieve top search results
5. Worked on the project report
6. Helped in designing crawler strategies

Abhilash Sunkam

1. Wrote code for crawling tweets from Twitter Streaming API
2. Checked out how to optimize the crawling code
3. Implemented multiprocessing
4. Learnt how Lucene works.
5. Helped in visualizing the indexed file using Luke
6. Researched the Basic layout of the search engine.
7. Worked on the project report.

Vishal Lella

1. Learnt how Twitter Streaming API works
2. Helped in deciding the hashtags to seed
3. Designed text analyzer choices
4. Helped in visualizing the indexed file using luke
5. Learnt how lucene works
6. Worked on the project report

References:

1. https://www.javacodegeeks.com/2015/09/building-a-search-index-with-lucene.html
2. http://lucene.apache.org/core/4_6_0/core/org/apache/lucene/document/StoredField.html
3. http://www.getopt.org/luke/
4. https://docs.python.org/2/library/multiprocessing.html
5. http://lucene.apache.org/
6. http://docs.tweepy.org/en/3.7.0/streaming_how_to.html