Information Retrieval and Web Search Project (Winter 2023) - Group 5

Topic : AthleticAlyze: A Twitter Sports Search Engine

Team Members:

1.Gayatri Bhosale (*gbhos002*)
2.Manoj Nagarajappa (*mnaga024*)
3.Puneet Singhania (*psing088*)
4.Sanjana Senthilkumar (*ssent013*)
5.Susmita Purushothaman (*spuru001*)

Data Source: Twitter

Collaboration details:
All team members took part in deciding the project topic and the general project workflow to implement. Puneet Singhania, Sanjana Senthilkumar and Susmita Purushothaman worked on developing the code that was used to crawl twitter data using Tweepy. We converted the responses to csv format, filtering out the required and relevant fields. Gayatri Bhosale and Manoj Nagarajappa worked on developing the code for indexing the data using Lucene. We all then created the sh files for crawler and indexer and then tested the code end-to-end. All team members worked on preprocessing, filtering strategies, crawling huge amounts of Twitter data and preparation of the reports.

Overview of the crawling system:
For our search engine, we decided to focus on tweets related to sports. For this purpose, we requested Elevated Access for our Twitter developer account to crawl large amounts of tweets. We collected Tweets that were geotagged as this information will be necessary for the second part of the project where we will use them to visualize our results. We chose python as our scripting language and Jupyter Notebook for our execution. We used Tweepy to crawl the tweets and collected JSON responses for a variety of hashtags and keywords related to sports and popular sporting figures. We extracted the necessary information from these responses and stored them in csv files. **We collected 520MB (more than 500MB) of data**, from an eclectic set of hashtags. We combined all of our csv files into a single file, removed duplicates again from our dataset and filtered tweets without geotags.

Crawling architecture:
Initially the tweets data is scraped from Twitter, then we take out the relevant records and fields from the response. We removed duplicate tweets from our dataset and filtered the trending tweets without geotags. We used a set for the tweetIds which helped in removing the duplicate data while crawling. Then we executed multiple threads to crawl the data so that tweets can be collected efficiently and quickly by using the multi-threaded crawler. We used 5 threads here which generated 5 different files containing the unique data for each hashtag being used. This

was done by each member of the team and for multiple attempts. The overall crawling architecture is well explained from the below diagram.

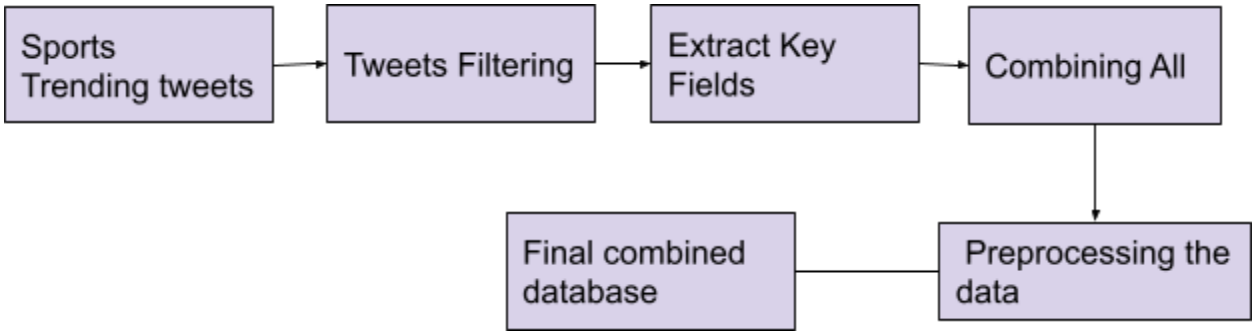


Fig.1: Crawling architecture

Crawling Strategy:

The API that we use returns recent tweets which are within a timeframe of 14 days. In order to maximize the amount of tweets we could collect, we chose trending sports hashtags and crawled tweets for them. We used OAuth (access keys and consumer keys) for our respective twitter accounts and crawled tweets for different multiple hashtags parallelly.

We then parsed the JSON responses and filtered out the following fields : 'id','tweet','time','lat','long','hashtag'. We combined the csv files for various hashtags. We removed duplicate tweets while crawling and again after collecting them into a single csv file. We collected tweets for the first two weeks of February 2023. In order to ensure that our search engine would have a wide range of tweets relating to sports we used the following keywords :

SerieA, LaLiga, NFLDraft, USOpen, RugbyWorldCup, NFLHonors, INDvAUS, Damar Hamlin, Brett Favre,Gates, soccer, AEW, cricket, INDVSAUS, man united, nhl, the nba, messi, D'Angelo, Sancho, Garnacho, Arribas, Rashford, Beasley, Westbrook, INDvAUS, Martinez, Man United, Laker, OMPSG, Utah, Weghorst, BGT2023, Old Trafford, Angelo Russell, Varane, Elon, Conley, Zelda, Ghost Trick, Switch, Fantasy Life, BOTW, Russ, LakeShow, AEWDynamite, Sixers, Tobias, Pikmin 4, The Acclaimed, TestCricket, 1st Test, James O'Keefe, Woody, KL Rahul, Kuldeep, Gill, Villanova, Blake Griffin, 76ers, Ashwin, Portland, Jazz, Vanderbilt, Jae Crowder, INDvsAUS, Pant, KS Bharat, Jadeja, Brooklyn, The NBA, Khawaja, KD to Phoenix, KD and CP3, fifa, football, soccer, futbol, messi, championsleague, premierleague, ronaldo, worldcup, pes, fifaultimateteam, realmadrid, futebol, uefa, laliga, neymar, cr, fifamobile, easportsfifa, barcelona, ucl, goal, RoyalRumble, ManCity, argentina, barcelona, billsmafia, bulls, championsleague, chess, clippers, cr, cristianoronaldo, dabears, eagles, easportsfifa, FCBLive, federer, fifa, fifa2022, fifa2023, fifaultimateteam, football, futbol,

futebol, golf, hockey, ipl, johnwall, laliga, lebronjames, lionelmessi, magnus, manchesterunited, mcfc, messi, nadal, NBA, neymar, nfl, olympic, patbev, patriots, pes, premierleague, rafaelnadal, raidernations, realmadrid, rockets, rogerfederer, ronaldo, royalrumble, skol, soccer, sports, sports2023, swimmingpool, tennis, tmltalks, ucl, uefa, worldcup

Cleaning and Handling Duplicate Tweets:

We remove duplicate tweets while crawling by putting them in a set having only unique tweetIDs. We then again handle duplicate tweets by removing them from each csv file after combining the data all together from different crawled files. We also remove the phrase 'RT' which is used to indicate when a tweet has been retweeted as this information is not needed. We remove links from the tweet as well as we are concerned with only the content. We removed hex characters from the tweets. We handle tweets that have multiple lines by storing them in one cell in the csv.

The list of fields extracted:

'id','tweet','time','lat','long','hashtag'

id : Taken from tweet.id. This is the unique id given to each twitter post.
tweet: Taken from tweet.full_text. This has the content of the individual tweet.
time: Taken from tweet.created_at. This returns the timestamp for when the tweet was posted.
lat: Taken from tweet.coordinates. This is the latitude coordinate for the user.
long: Taken from tweet.coordinates. This is the longitude coordinate for the user.
hashtag: Taken from the tweet body. This has all the hashtags that each user included in their tweets.

By storing the incoming tweets in a csv using the above process, we get the maximum throughput. To emphasize again, each tweet on Twitter has a special ID that can be used to distinguish it from other tweets. Whether a tweet's ID has previously been received, we check to see if it hasn't been stored yet and put it accordingly.

Overview of Lucene indexing strategy:

After gathering the complete data using the crawler, we used pylucene to index the dataset based on a few fields, like hashtags, and tweet. Standard Analyzer for tweets and Keyword Analyzer for hashtags were employed here. Additionally, we looked at the performance of pylucene as the number of documents increases. We tested the performance using a number of documents. Overall, for more than **1900000 documents(tweets)**, the indexing takes around 113 seconds which is very good.

Fields used in the Pylucene index and their justification:

Hashtag: It is possible to efficiently and quickly retrieve information on trending sports topics by indexing on hashtags. They hold a special meaning on twitter and its easier to find all the relevant tweets using the hashtag so we indexed using them. Because hashtags and keywords have different properties and use cases and necessitate various indexing techniques, it is usual practice to index hashtags independently from keywords.

Hashtags are frequently used to group tweets with similar content together or to categorize tweets into topics. For purposes like trend analysis or subject tracking, we can rapidly get tweets that contain a given hashtag by indexing hashtags independently.

On the other hand, keywords are frequently used to search for and retrieve tweets based on their content. For purposes like sentiment analysis, indexing keywords individually enables effective searching and retrieval of tweets that contain particular keywords.

We can optimize the data storage and retrieval systems for the unique use cases of each type of information by indexing hashtags and keywords independently, which can result in better performance, and less storage overhead.

Tweet: Text-based tweets are indexed in order to facilitate high-speed and easier search functionality.

The fields that we haven't indexed are as follows:

ID: As we will need to know this ID beforehand and the search queries usually employ natural language to retrieve the data, indexing on this field will not be of any assistance to us when searching. It is very tough to search using this field owing to the fact that we need to know the specific ID before executing the search operation. A low Lucene scoring would result from indexing on unique fields. Owing to the above reasons, we chose not to index this field.

Time: The time of the creation of the tweet is stored in this field. This field was removed from our list of indexing because it is unique. A low Lucene scoring would result from indexing on unique fields. Owing to the above reasons, we chose not to index this field.

Text analyzer choices:

1. Standard Analyzer- It is used to process text content and turn it into tokens that can be effectively searched and indexed.

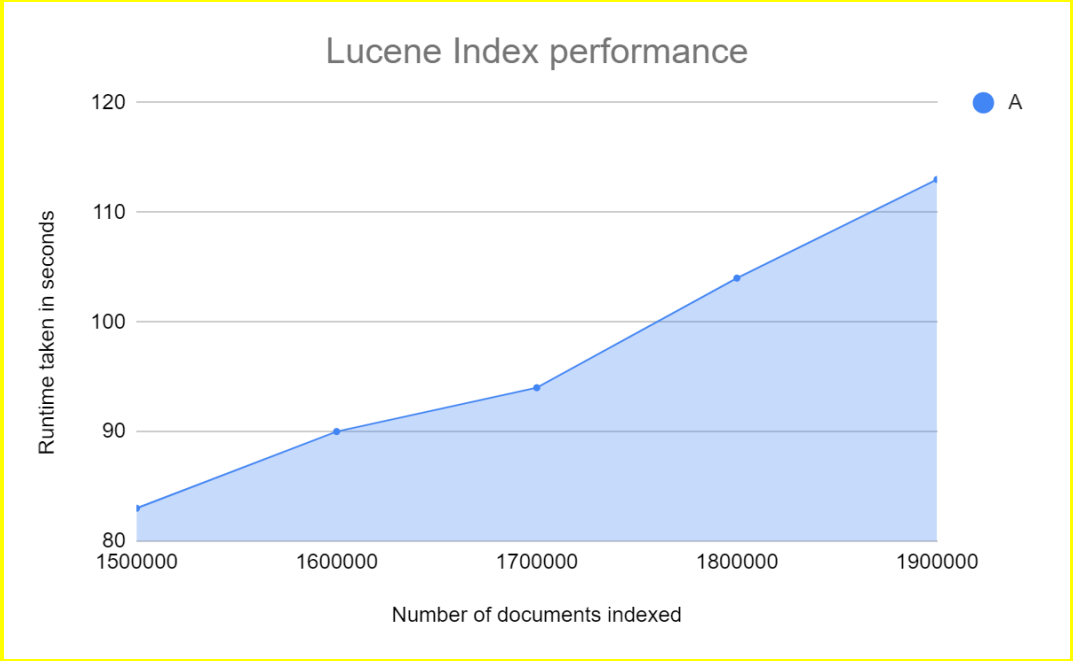
The Standard Analyzer goes through the following steps in its processing pipeline:

- Converts all characters to lowercase.
- By separating text into tokens (also known as terms) based on whitespace and punctuation letters.
- Stop words are terms that are frequently used in texts and are probably not distinctive/unique enough to distinguish between different documents, therefore they are removed from searches.
- In order to match word variations (like "running" and "runner"), stemming reduces words to their base forms.
- To lessen index noise, any special characters that are not letters or numbers are eliminated.

2. Keyword Analyzer - The Keyword Analyzer just treats the entire input as a single token rather than parsing the text and breaking it down into tokens.
It is helpful in circumstances where you want to perform a straightforward, exact-text search. It is more effective for searching exact strings than other analyzers that carry out more extensive processing because it doesn't analyze or process the text.

Lucene index performance:

Below is a performance analysis of Lucene indexing, that is, the *run time of the Lucene index* creation process. A graph with *run time on the y axis* and *number of documents on the x axis*.



Limitations of the system:

1. Data Crawling is not a rapid process and it takes a lot of time to crawl over 500 MB of data.
2. Complete crawling of tweets, not just those containing the search term is performed. Therefore, the information that is not specifically related to the topic may also be displayed as part of the search term.
3. Due to the retweets and quotes referring to the same search term, there may be duplicate data.
4. The entire material must be crawled over multiple hours, so to quickly fetch the data, we used numerous accounts to make sure that during the crawling, no new data duplication was added. Multiple csv files were obtained which were then combined into a single csv file after removing the duplicates again.
5. The rate limiting of Twitter and obtaining elevated access for crawling the tweets poses quite a challenge.
6. Finding words that are synonyms of one another won't produce the same outcomes.
7. Currently our indexer does not handle hex characters so we removed them during the preprocessing stage. In the future, we would like our indexer to handle those characters too and display the relevant values.

Obstacles and solutions:

Crawling usually results in an enormous amount of output, the most of which was not relevant. So, only six fields were used while the other fields were removed in the actual crawling process. We also encountered difficulties when attempting to collect data from Twitter due to the API endpoints' rate constraints, which place some restrictions on the total number of tweets that can be gathered in a given period of time. As a solution to this issue, to collect data from many hashtags concurrently, we employed numerous accounts and credentials.

Instructions to Deploy the Crawler:

We use Jupyter Notebook to execute our python scripts. We utilize Tweepy to run the code to crawl tweets. We provide our access and consumer credentials for authentication.

./crawler.sh "tweetsDataset" "10"

Arguments required:

1) arg1 : outputFilename - CSV output filename.

2) arg2 : maxTweets - Maximum tweets that need to be crawled per API call.

Instructions on how to build Lucene index:

Extract the contents of the submitted zip file and run the below command in the terminal with appropriate arguments.

./AthleticAlyze.sh "final_onemb.csv" "fifa" "tweet"

./AthleticAlyze.sh "final_onemb.csv" "fifa" "hashtags"

Arguments required:

1) arg1 : csvFilename - CSV dataset file to be parsed.

We have stored the sample data in 'final_onemb.csv'

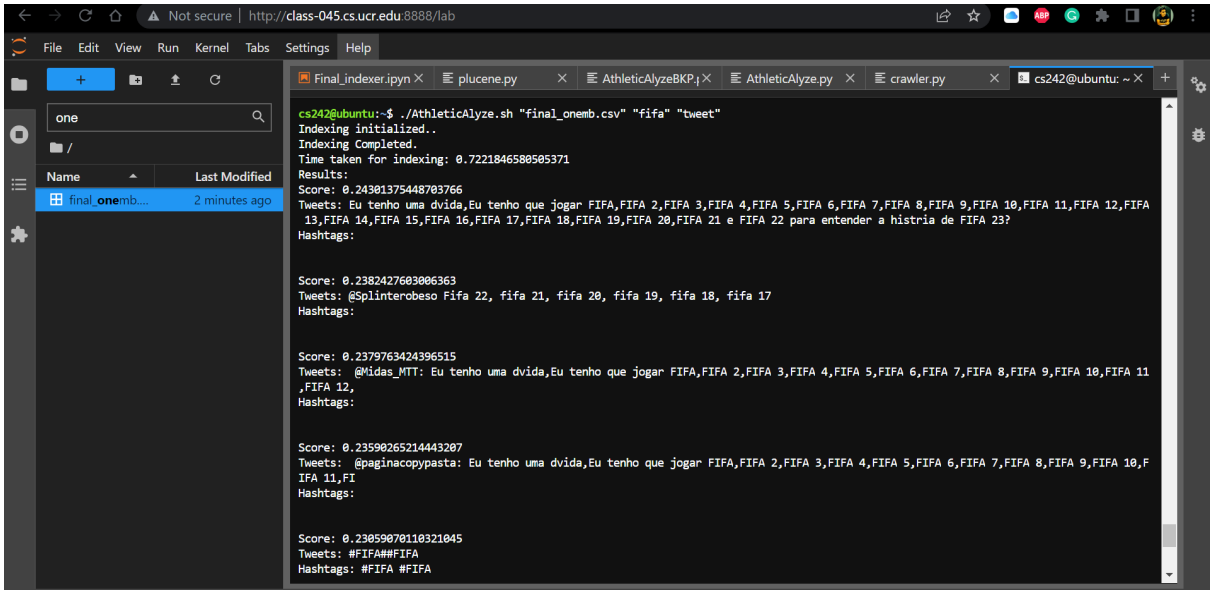
2) arg2 : searchKey. In the above example, the word we are searching for is "fifa". Multiple words can also be provided for the search operation. For example: "fifa 2023".

3) arg3 : searchCol. In the above examples, the columns we are using are "tweet" and "hashtags". To search using the tweet or hashtags column, "tweet" or "hashtags" should be given as an argument.

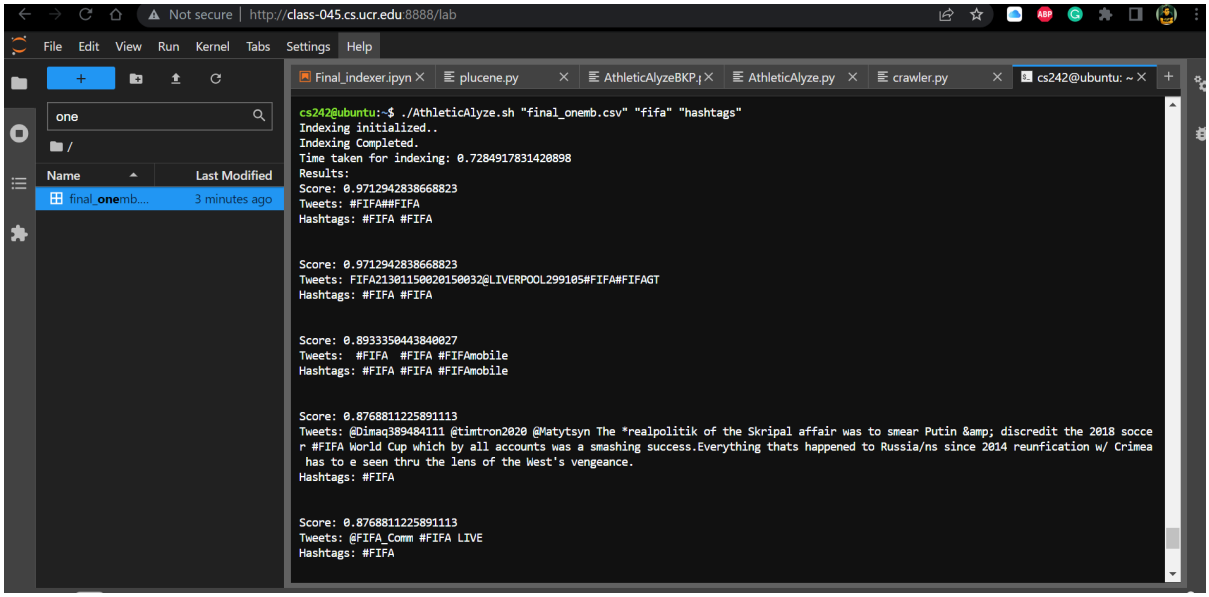
Sample Query Screenshot:

Please find the terminal screen snapshot below, which displays a sample query, the top 5 results that were returned on indexbuilder's execution, and Lucene scores:

./AthleticAlyze.sh "final_onemb.csv" "fifa" "tweet"

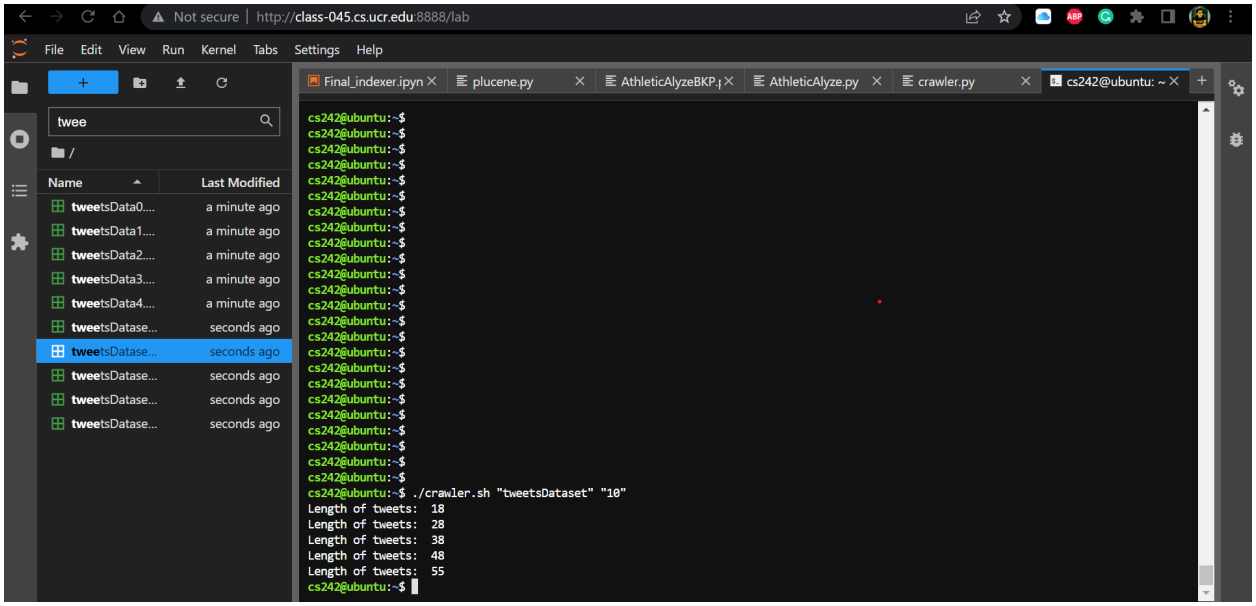


./AthleticAlyze.sh "final_onemb.csv" "fifa" "hashtags"



Please find the terminal screen snapshot below which is obtained on the crawler's execution:

./crawler.sh "tweetsDataset" "10"


Not secure | http://class-045.cs.ucr.edu:8888/lab
File Edit View Run Kernel Tabs Settings Help
twee
/
Name
Last Modified
tweetsData0.... a minute ago
tweetsData1.... a minute ago
tweetsData2.... a minute ago
tweetsData3.... a minute ago
tweetsData4.... a minute ago
tweetsDatase... seconds ago
tweetsDatase... seconds ago
tweetsDatase... seconds ago
tweetsDatase... seconds ago
tweetsDatase... seconds ago
Final_indexer.ipyn
plucene.py
AthleticAlyzeBKP.j
AthleticAlyze.py
crawler.py
cs242@ubuntu: ~
cs242@ubuntu:~$ ./crawler.sh "tweetsDataset" "10"
Length of tweets: 18
Length of tweets: 28
Length of tweets: 38
Length of tweets: 48
Length of tweets: 55