

#GIVEAWAY TWITTER SEARCH ENGINE

Introduction to the Problem Statement

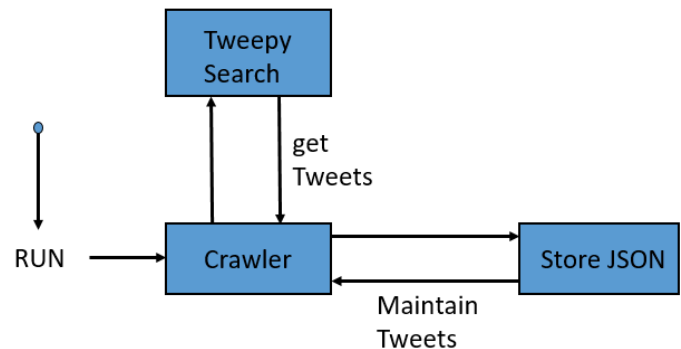
Giveaways are a popular method for expanding a social media audience on websites like Twitter, Twitch, YouTube, etc. On social media, a lot of influencers and celebrities launch giveaways and publish requirements. The followers are supposed to adhere to those directives. Then, a select number of fortunate people receive the gifts. Twitter is used by many famous personalities and powerful people. These giveaway tweets typically go viral and receive a lot of attention from followers. On the other hand, if you look at it from the perspective of a regular user, how many influencer accounts can one person follow? Also, because they happen frequently, tracking them is difficult. To develop a search engine that is dedicated to Giveaway tweets so that users may easily find them through keyword searches.

The project will use the Twitter API to gather tweets about giveaways, use PyLucence and BERT to index the tweets and create a user-friendly interface for searching and filtering through the indexed tweets. The tweets will be gathered by filtering for particular hashtags and keywords associated with giveaways. The user interface makes it simple for users to look for giveaways by keywords, filter results by date range, and enjoyably watch tweets.

I intend to build a corpus of tweets containing hashtags: #giveaway, #giftcard, #raffles, #win, #cosmetics, #food, #concert, #amazon, #amazongiftcard, #holiday, #nike, #nikegiftcard, #valentine, #freebie, #freebiefriday, #competition. I plan to index this corpus with PyLucene and BERT. A Web interface will be built where the user can search for a given keyword.

Overview of the Crawling System

Twitter is a social media and networking site that lets users send and receive quick messages, or "tweets," in real-time. Twitter maintains data that is available to developers. After setting up a developer account, asked for advanced developer access. To extract tweets from Twitter, used the free, open-source Tweepy package. Tweepy offers a tweet-pulling cursor for use. This is vastly superior to repeatedly accessing the Twitter REST APIs since the Tweepy cursor cleverly retrieves the tweets until the Twitter Rate constraint is fulfilled and it has very effective exception handling. Started the crawler to run for a very long time using the Tweepy cursor. A hashtag is chosen at random from # at the start of each batch. To ensure that only tweets about giveaways are pulled, this is appended to the hashtag #giveaway. Upon the creation of the connection using the API key, API secret key, Access Token, and Access Token Secret, the Tweepy cursor receives this connection object. The cursor uses api.search tweets with the "recent" filter, pulling tweets using the search query specified within the last seven days.



1. Twitter Crawling Architecture:

The architecture consists of three files: config.py, crawler.py, crawler.sh

- a) **config.py**: The API keys, consumer key, API/Consumer secret, access key, and access key secret generated by the Twitter app are stored in configuration files. The first thing performed for this project was to create the app in the Twitter developer account.
- b) **crawler.py**: The maximum number of requests that can be submitted during a given time interval, usually 15 minutes, is constrained. A maximum of 2000 requests can be sent to an endpoint within 15 minutes, for instance, if its rate restriction is 1000 requests per OAuth2 window or 2000 requests per OAuth1 window. Submit more than 2000 requests in under 15 minutes since over 550 MB of data was crawled. So, we've introduced a wait-on rate cap. It enables automatic data retrieval for us. When the cap of 2000 requests per 15 minutes is reached, the crawler will more thoroughly slumber. The algorithm will automatically relaunch the data crawling process after 15 minutes based on the geolocation of the data, and if the geolocation is there, then crawl and store the data. The crawler scans and stores tweets in a single thread. It loops through all of the tweets that the tweepy has returned. and saves them in batches to files using the cursor. The crawler takes a brief break between batches to save the data and prevent exceeding Twitter's rate constraints. Each batch is saved in a separate JSON file.

c) **crawler.sh**: An executable file is included that takes as input all necessary parameters.

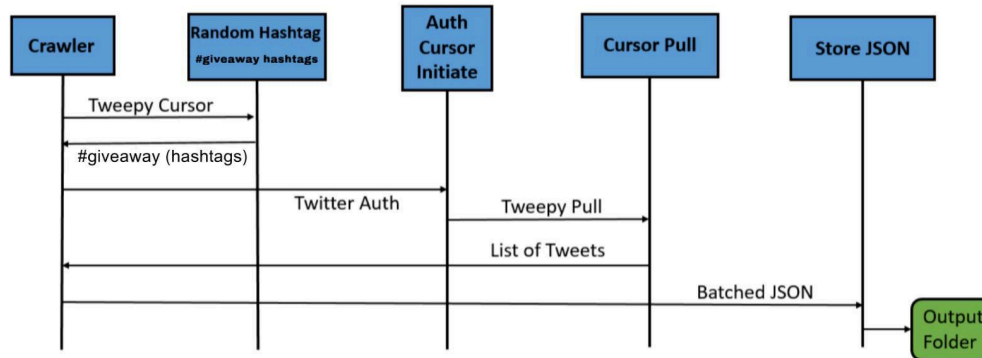


Fig: Crawler Activity Diagram

2. Twitter Crawling Strategy

Handling Duplicate Tweets: -

```

import json
import csv

with open('/content/data.json') as json_file:
    jsontdata = json.load(json_file)

data_file = open('duplicatesrem.csv', 'w', newline='')
csv_writer = csv.writer(data_file)

count = 0
for data in jsontdata:
    if count == 0:
        header = data.keys()
        csv_writer.writerow(header)
        count += 1
    csv_writer.writerow(data.values())

data_file.close()

import pandas as pd
file_name = "/content/duplicatesrem.csv"
file_name_output = "file_without_dupes.csv"

df = pd.read_csv(file_name, sep="\t or ,")

df.drop_duplicates(subset=None, inplace=True)

# Write the results to a different file
df.to_csv(file_name_output, index=False)

[5]: import pandas as pd
df = pd.read_csv ('/content/file_without_dupes.csv')
df.to_json ('duplicates_removed.json')
    
```

Data Storage:

Write the tweet data in a JSON writer and the text file in the JSON file.

```

json_filename = (f"data.json")
with open(json_filename, "w") as f:
    json.dump(tweets_to_save, f)
    logger.info(f"Saved {json_filename}")
    
```

Tweepy Cursor:

The cursor method is the latest release of Tweepy. It handles the pagination work for us behind the screen so the code can now just focus on processing the results. I have passed the parameters like query, the language into a cursor, and

```

tweets = tweepy.Cursor(
    api.search_tweets,
    q=search_words,
    lang="en",
    result_type="recent",
).items(MAX_TWEETS)
    
```

the result type and will automatically pass the parameters into a request whenever a request is made.

```
MacBook-Pro:TwitterDataCrawler saistruthikovur$ sh crawler.sh 200000 'output/'
2023-02-12 02:51:37.692 | INFO | __main__:<module>:136 -

Crawler is Starting!
Config loaded from: config.json.

2023-02-12 02:51:37.692 | INFO | __main__:<module>:153 - Iteration 0 starting... | Crawler is gonna run with search
2023-02-12 02:51:37.692 | INFO | __main__:authenticate:28 - Authenticated! Crawler ready to start!

Rate limit reached. Sleeping for: 821
2023-02-12 03:21:02.248 | INFO | __main__:crawl:91 - Crawled 2000 tweets, pausing to save batch number: 1
2023-02-12 03:21:02.249 | INFO | __main__:crawl:94 - Took 917s to crawl 2000 tweets
2023-02-12 03:21:02.323 | INFO | __main__:crawl:99 - Saved output/giveawayWins/batch-1-2023-02-12T03:21:02.json
2023-02-12 03:21:02.447 | INFO | __main__:crawl:105 - Saved data.json

Rate limit reached. Sleeping for: 816
2023-02-12 03:36:28.204 | INFO | __main__:crawl:91 - Crawled 4000 tweets, pausing to save batch number: 2
2023-02-12 03:36:28.206 | INFO | __main__:crawl:94 - Took 1843s to crawl 2000 tweets
2023-02-12 03:36:28.301 | INFO | __main__:crawl:99 - Saved output/giveawayWins/batch-2-2023-02-12T03:36:28.json
2023-02-12 03:36:28.304 | INFO | __main__:crawl:105 - Saved data.json

Rate limit reached. Sleeping for: 813
2023-02-12 03:51:43.398 | INFO | __main__:crawl:91 - Crawled 6000 tweets, pausing to save batch number: 3
2023-02-12 03:51:43.399 | INFO | __main__:crawl:94 - Took 2758s to crawl 2000 tweets
2023-02-12 03:51:43.482 | INFO | __main__:crawl:99 - Saved output/giveawayWins/batch-3-2023-02-12T03:51:43.json
2023-02-12 03:51:43.601 | INFO | __main__:crawl:105 - Saved data.json

Rate limit reached. Sleeping for: 809
2023-02-12 04:07:01.000 | INFO | __main__:crawl:91 - Crawled 8000 tweets, pausing to save batch number: 4
2023-02-12 04:07:01.001 | INFO | __main__:crawl:94 - Took 3676s to crawl 2000 tweets
2023-02-12 04:07:01.057 | INFO | __main__:crawl:99 - Saved output/giveawayWins/batch-4-2023-02-12T04:07:01.json
2023-02-12 04:07:01.062 | INFO | __main__:crawl:105 - Saved data.json

Rate limit reached. Sleeping for: 815
Rate limit reached. Sleeping for: 813
2023-02-12 04:35:52.793 | INFO | __main__:crawl:91 - Crawled 10000 tweets, pausing to save batch number: 5
2023-02-12 04:35:52.796 | INFO | __main__:crawl:94 - Took 5408s to crawl 2000 tweets
2023-02-12 04:35:53.008 | INFO | __main__:crawl:99 - Saved output/giveawayWins/batch-5-2023-02-12T04:35:52.json
2023-02-12 04:35:53.192 | INFO | __main__:crawl:105 - Saved data.json

Rate limit reached. Sleeping for: 812
2023-02-12 04:51:23.402 | INFO | __main__:crawl:91 - Crawled 12000 tweets, pausing to save batch number: 6
2023-02-12 04:51:23.403 | INFO | __main__:crawl:94 - Took 6338s to crawl 2000 tweets
2023-02-12 04:51:23.541 | INFO | __main__:crawl:99 - Saved output/giveawayWins/batch-6-2023-02-12T04:51:23.json
2023-02-12 04:51:23.549 | INFO | __main__:crawl:105 - Saved data.json

Rate limit reached. Sleeping for: 806
2023-02-12 05:06:54.703 | INFO | __main__:crawl:91 - Crawled 14000 tweets, pausing to save batch number: 7
2023-02-12 05:06:54.705 | INFO | __main__:crawl:94 - Took 7270s to crawl 2000 tweets
2023-02-12 05:06:54.882 | INFO | __main__:crawl:99 - Saved output/giveawayWins/batch-7-2023-02-12T05:06:54.json
```

Fig: Logs during the execution of the crawler

The list of fields in the JSON file is listed below:

- tweet_id - The integer representation of the unique identifier for this Tweet.
- user_id - The string representation of the unique identifier for this Tweet.
- Tweet content (tweet) - The actual UTF-8 text of the status update.
- Location(lat, lang & geo) - Represents the geographic location of this Tweet as reported by the user or client application.
- user_name - Name of the user who posted this Tweet.
- posted_at - UTC when this Tweet was created.
- followers_count - The number of followers the user has.
- retweet_count - Number of times this Tweet has been retweeted.
- tweet_count - Number of times this Tweet has been tweeted.

- listed_count - Number of times this Tweet has been listed.
- like_count - Indicates approximately the count of a tweet that has been liked by Twitter users.
- hashtags - Hashtags that are mentioned in the tweet
- user_mentions: Number of mentioned added by the user
- verified - Verified status of the user account.

```

tweet_id:      1624260567195353000
user_id:       6354392
tweet:        "WIN $3,700 Giveaway from Teeming River Cruises #giveaway #win
lng:          null
lat:          null
geo:          "Sydney, Australia"
user_name:     "SM"
posted_at:    "2023-02-11 04:14:31"
followers_count: 35
retweet_count: 0
tweets_count: 91
listed_count:  2
like_count:    0
hashtags:
  0:           "giveaway"
  1:           "win"
user_mentions: null
verified:      0

```

Fig: List of fields in the JSON file

Overview of the PyLucene indexing strategy

i) Fields in the PyLucene index

tweet_id : Used to easily look up the handle of the user who posted the giveaway tweet.

user_name : The username is unique and helps to create the index in PyLucene.

tweet : The content of the giveaway hashtag is the tweet which is a vital field for indexing.

geo : Traces the precise location from the tweet that was posted

posted_at : Date and time when the tweet containing the giveaway hashtag is posted

followers_count : Traces back the number of followers who posted the tweet to ease the process of indexing.

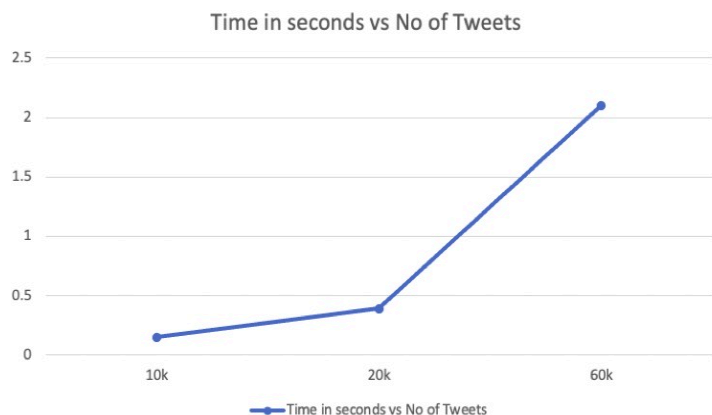
Two files namely index.py and the shell script is used in the process of indexing.

ii) Text Analyzer

Here, a standard analyzer is utilized. It is available in the PyLucene package and is primarily implemented to delete punctuations and unnecessary words that occurred in the JSON file after crawling. Before indexing, stop words like "a," "the," and "an" are also eliminated. The text is likewise changed to lowercase by Standard Analyzer.

iii) Run time of the PyLucene index creation process

The graph obtained after analyzing and indexing is as follows:



In the above graph, the y-axis corresponds to the time taken for the index creation process in seconds and the x-axis represents the number of tweets with hashtags relevant to the giveaway. For 66,000 tweets the run time is less than 1.5 seconds, for 20,000 tweets it is 0.4 seconds and a large run time is obtained for 60,000 tweets (i.e.) 2.1 seconds. Thus, can conclude that the larger the number of tweets considered, the run time increases.

PyLucene index results

The retrieve function creates a QueryParser and uses it to parse the query string to search the index. The user's search query is supplied as an argument to the retrieve function as the query string. The search method of the IndexSearcher instance is used to perform the query and retrieve the top 10 results when the query string has been properly parsed. The topDocs variable contains the top 10 outcomes.

Run time and output of the PyLucene index(from report A):

```
Loaded 264000 JSON documents.
Lucene took 1758ms to index 66000 JSON documents.
Lucene took 5301ms to index 132000 JSON documents.
Lucene took 6097ms to index 198000 JSON documents.
Lucene took 6475ms to index 264000 JSON documents.
Indexing is done!
```

```
cs242@ubuntu:~/Group12$ sh index.sh
[{"score": 2.1115267276763916, "user_name": "imperfect replica", "tweet": "RT @GiveawayBase: Grab your #free entries for a chance to win a brand new #microphone! #worldwide #giveaway #sweepstakes #free #stream #g.", "follower": "558", "geo": "Twilight Town", "Date": "2023-02-10 08:29:30"}, {"score": 2.1115267276763916, "user_name": "Cassies reviews", "tweet": "Knock, knock! Free items are waiting for you! Accept my invitation and earn free items!\nhttps://t.co/bVEmEfUhgZ", "follower": "1765", "geo": "Ohio, USA", "Date": "2023-02-08 12:24:15"}, {"score": 1.950984001159668, "user_name": "Cynthia Sizemore", "tweet": "FREE Daily Green Smoothie Kit #giveaway #win https://t.co/o03w0M5479", "follower": "571", "geo": "Manchester, Ohio", "Date": "2023-02-07 21:53:00"}, {"score": 1.950984001159668, "user_name": "Cynthia Sizemore", "tweet": "FREE Daily Green Smoothie Kit #giveaway #win https://t.co/o03w0M5479", "follower": "572", "geo": "Manchester, Ohio", "Date": "2023-02-07 21:53:00"}, {"score": 1.950984001159668, "user_name": "Jeanette Gillett", "tweet": "FREE Daily Green Smoothie Kit #giveaway #win https://t.co/4U1Y1GoPO", "follower": "545", "geo": "Morristown, TN", "Date": "2023-02-07 19:12:28"}, {"score": 1.950984001159668, "user_name": "Veracity Tech. LLC", "tweet": "Free Website Analysis https://t.co/Xh59UDzIdF #giveaway #win via @RealKingSumo", "follower": "359", "geo": "New Albany, IN", "Date": "2023-02-08 19:29:45"}, {"score": 1.950984001159668, "user_name": "Cynthia Sizemore", "tweet": "FREE Daily Green Smoothie Kit #giveaway #win https://t.co/o03w0M5479", "follower": "571", "geo": "Manchester, Ohio", "Date": "2023-02-07 21:53:00"}, {"score": 1.950984001159668, "user_name": "Veracity Tech. LLC", "tweet": "Free Website Analysis https://t.co/Xh59UDzIdF #giveaway #win via @RealKingSumo", "follower": "359", "geo": "New Albany, IN", "Date": "2023-02-08 19:29:45"}, {"score": 1.86235511302948, "user_name": "val swanson", "tweet": "FREE Daily Green Smoothie Kit #giveaway #win https://t.co/5MBW18wpLH @RanaDurham @oh_thats_mookie", "follower": "663", "geo": "East Coast", "Date": "2023-02-10 03:33:40"}]
```


Overview of the BERT

BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained language model developed by Google that can be fine-tuned for a variety of natural language processing (NLP) tasks, such as text classification, question answering, and named entity recognition.

The initial step when utilizing BERT for a particular NLP task is to optimize the pre-trained model on a task-specific dataset. To optimize the model for the given job, extra layers are learned on top of the pre-trained BERT model. The pre-trained BERT model's weights are changed as part of the fine-tuning procedure to improve its performance on the task-specific dataset.

BERT is a model option that comes in several different forms, including BERT-base, BERT-large, and others. The size of the dataset, the difficulty of the task, and the available computer resources all influence which model is selected.

BERT is often used in conjunction with an indexing scheme that makes it possible for quick retrieval of pertinent documents or passages. The sentence embedding strategy, which entails using BERT to encode each sentence in a corpus and storing the resulting embeddings in an index, is a typical indexing schema for BERT.

To retrieve pertinent texts or passages, a query sentence is additionally encoded using BERT and compared to the sentence embeddings in the index. BERT cosine similarity along with the ranking for each tweet is also obtained as a result and stored in the database. The latitude and longitude results stored in MySQL are then fetched to plot the map using the folium library (Extra Credit section shows the results)

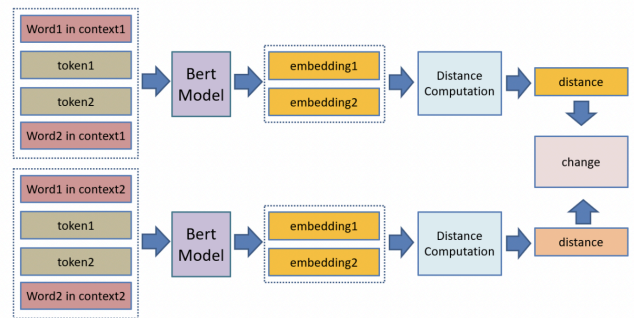


Fig: BERT architecture

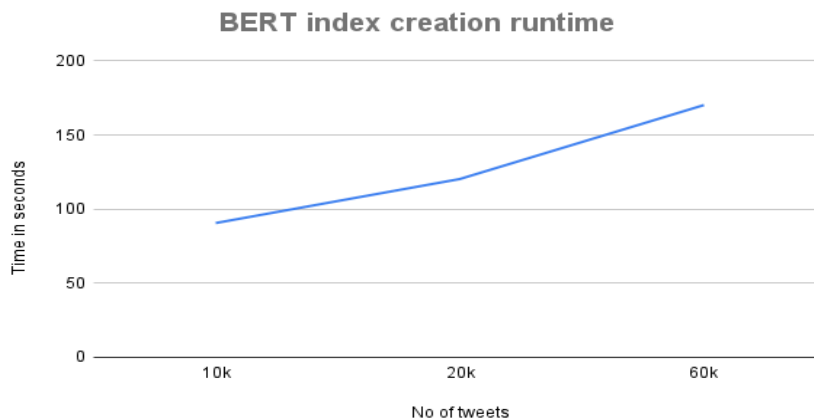


Fig. Runtime of BERT index creation

The first step when utilizing the BERT index to perform ranking is to encode the query using BERT to produce a query embedding. In a high-dimensional vector space, this embedding represents the semantic meaning of the query. The next step is to get a set of potential documents or passages from the index using the query embedding. Using a similarity metric like cosine similarity or dot product, the returned candidates are ordered according to how similar they are to the query embedding.

A ranking algorithm is used to determine the relevance score for each candidate document or passage after the candidate set has been retrieved. The ranking system takes into account several factors, including the query phrases' placement within the page, their frequency, and other relevant signals.

The relevance ratings given to each contender are then used to determine the final ranking. The search returns the top-ranked texts or passages as its results.

Run time and output of the BERT index()

```
1 |Time taken to create BERT embeddings: 95.99 seconds
2 |Total tweets indexed: 2000
3 |Time taken to create BERT embeddings: 95.99 seconds
4 |Number of tweets with the word 'giveaway': 2000
5 |Top 50 tweets based on cosine similarity and ranking:
6 |1. RT @LyndaCheckley: 🎉🎉GIVEAWAY🎉🎉
7 |
8 |The Delegate by Ali Carter
9 |
10 |As part of the @RandomTTours book tour you can win a copy thanks to @matadorb... (distance: 0.1112, ranking: 1)
11 |2. RT @LyndaCheckley: 🎉🎉GIVEAWAY🎉🎉
12 |
13 |The Delegate by Ali Carter
14 |
15 |As part of the @RandomTTours book tour you can win a copy thanks to @matadorb... (distance: 0.1112, ranking: 2)
16 |3. RT @LyndaCheckley: 🎉🎉GIVEAWAY🎉🎉
17 |
18 |The Delegate by Ali Carter
19 |
20 |As part of the @RandomTTours book tour you can win a copy thanks to @matadorb... (distance: 0.1112, ranking: 3)
21 |4. RT @LyndaCheckley: 🎉🎉GIVEAWAY🎉🎉
22 |
23 |The Delegate by Ali Carter
24 |
25 |As part of the @RandomTTours book tour you can win a copy thanks to @matadorb... (distance: 0.1112, ranking: 4)
26 |5. RT @LyndaCheckley: 🎉🎉GIVEAWAY🎉🎉
27 |
28 |The Delegate by Ali Carter
29 |
30 |As part of the @RandomTTours book tour you can win a copy thanks to @matadorb... (distance: 0.1112, ranking: 5)
31 |6. RT @LyndaCheckley: 🎉🎉GIVEAWAY🎉🎉
32 |
33 |The Delegate by Ali Carter
34 |
35 |As part of the @RandomTTours book tour you can win a copy thanks to @matadorb... (distance: 0.1112, ranking: 6)
36 |7. RT @LyndaCheckley: 🎉🎉GIVEAWAY🎉🎉
37 |
38 |The Delegate by Ali Carter
39 |
40 |As part of the @RandomTTours book tour you can win a copy thanks to @matadorb... (distance: 0.1112, ranking: 7)
41 |8. RT @LyndaCheckley: 🎉🎉GIVEAWAY🎉🎉
42 |
43 |The Delegate by Ali Carter
44 |
45 |As part of the @RandomTTours book tour you can win a copy thanks to @matadorb... (distance: 0.1112, ranking: 8)
46 |9. RT @LyndaCheckley: 🎉🎉GIVEAWAY🎉🎉
47 |
48 |The Delegate by Ali Carter
49 |
```

Fig: BERT Indexing Result 3 with a tweet, cosine distance, and ranking score


```

50 As part of the @RandomTTours book tour you can win a copy thanks to @matadorb... (distance: 0.1112, ranking: 9)
51 10. RT @LyndaCheckley: 🎉🎉GIVEAWAY🎉🎉
52
53 The Delegate by Ali Carter
54
55 As part of the @RandomTTours book tour you can win a copy thanks to @matadorb... (distance: 0.1112, ranking: 10)
56 11. RT @LyndaCheckley: 🎉🎉GIVEAWAY🎉🎉
57
58 The Delegate by Ali Carter
59
60 As part of the @RandomTTours book tour you can win a copy thanks to @matadorb... (distance: 0.1112, ranking: 11)
61 12. RT @LyndaCheckley: 🎉🎉GIVEAWAY🎉🎉
62
63 The Delegate by Ali Carter
64
65 As part of the @RandomTTours book tour you can win a copy thanks to @matadorb... (distance: 0.1112, ranking: 12)
66 13. RT @LyndaCheckley: 🎉🎉GIVEAWAY🎉🎉
67
68 The Delegate by Ali Carter
69
70 As part of the @RandomTTours book tour you can win a copy thanks to @matadorb... (distance: 0.1112, ranking: 13)
71 14. RT @LyndaCheckley: 🎉🎉GIVEAWAY🎉🎉
72
73 The Delegate by Ali Carter
74
75 As part of the @RandomTTours book tour you can win a copy thanks to @matadorb... (distance: 0.1112, ranking: 14)
76 15. RT @LyndaCheckley: 🎉🎉GIVEAWAY🎉🎉
77
78 The Delegate by Ali Carter
79
80 As part of the @RandomTTours book tour you can win a copy thanks to @matadorb... (distance: 0.1112, ranking: 15)
81 16. RT @LyndaCheckley: 🎉🎉GIVEAWAY🎉🎉
82
83 The Delegate by Ali Carter
84
85 As part of the @RandomTTours book tour you can win a copy thanks to @matadorb... (distance: 0.1112, ranking: 16)
86 17. RT @LyndaCheckley: 🎉🎉GIVEAWAY🎉🎉
87
88 The Delegate by Ali Carter
89
90 As part of the @RandomTTours book tour you can win a copy thanks to @matadorb... (distance: 0.1112, ranking: 17)
91 18. RT @collinsjacob115: #Giveaway time!
92
93 I have one paperback copy of the brilliant new novel by @FionaAnnCummins to #win
94
95 To enter:
96
97 Follow @... (distance: 0.1113, ranking: 18)
98 19. RT @collinsjacob115: #Giveaway time!

```

Fig: BERT Indexing Result 3 with a tweet, cosine distance, and ranking score

```

98 19. RT @collinsjacob115: #Giveaway time!
99
100 I have one paperback copy of the brilliant new novel by @FionaAnnCummins to #win
101
102 To enter:
103
104 Follow @... (distance: 0.1113, ranking: 19)
105 20. RT @collinsjacob115: #Giveaway time!
106
107 I have one paperback copy of the brilliant new novel by @FionaAnnCummins to #win
108
109 To enter:
110
111 Follow @... (distance: 0.1113, ranking: 20)

```

Fig: BERT Indexing Result 3 with the tweet, cosine distance, and ranking score

BERT results stored in MySQL

tweet \					
0	•दो दोस्तों के साथ वीडियो बनाएं\n•गीत डालें \n...				
1	•दो दोस्तों के साथ वीडियो बनाएं\n•गीत डालें \n...				
2	•दो दोस्तों के साथ वीडियो बनाएं\n•गीत डालें \n...				
3	•दो दोस्तों के साथ वीडियो बनाएं\n•गीत डालें \n...				
4	@PassPass_Maze You "A-Maze" me!!!!\n\n#MazeOfL...				
...	...				
12958	RT @cryptomarsdo: \$100 #Giveaway in 1Day\n\nRu...				
12959	RT @cryptomarsdo: \$100 #Giveaway in 1Day\n\nRu...				
12960	RT @cryptomarsdo: \$100 #Giveaway in 1Day\n\nRu...				
12961	👉 Birmingham v West Brom is the fixture select...				
12962	👉 Birmingham v West Brom is the fixture select...				
		embedding	distance	ranking	\
0	[-0.4450036585330963, -0.06633590161800385, -0...		0.0	6	
1	[-0.4450036585330963, -0.06633590161800385, -0...		0.0	6	
2	[-0.4450036585330963, -0.06633590161800385, -0...		0.0	6	
3	[-0.4450036585330963, -0.06633590161800385, -0...		0.0	6	
4	[0.16050933301448822, 0.04243739694356918, 0.2...		0.0	46	
...
12958	[0.019889768213033676, 0.22881314158439636, -0...		0.0	1931	
12959	[0.019889768213033676, 0.22881314158439636, -0...		0.0	1931	
12960	[0.019889768213033676, 0.22881314158439636, -0...		0.0	1931	
12961	[-0.1608889102935791, 0.00030578672885894775, ...		0.0	2	
12962	[-0.06598464399576187, -0.020626073703169823, ...		0.0	1	
...					
12961	0.628710	London (UK)			
12962	0.619978	🇬🇧 London, England			
[12963 rows x 6 columns]					

Fig: BERT results stored in MySQL

Comparison of the ranking quality of PyLucene and BERT:

- PyLucene and BERT are both powerful information retrieval tools, but they differ in their underlying architectures and approaches to ranking.
- PyLucene is a popular open-source information retrieval library, while BERT is a state-of-the-art deep learning-based language model for natural language processing.

PyLucene:

Lucene uses a vector space model, where documents are represented as vectors in a high-dimensional space, and queries are also represented as vectors. Documents are ranked based on their similarity to the query vector, which is calculated using various techniques such as cosine similarity.

BERT:

BERT uses a transformer-based architecture that is pre-trained on a large corpus of text data to generate contextualized word embeddings. These embeddings capture the relationships between words and enable BERT to understand the meaning and context of language better.

Scenario-based on the project:

To compare the rankings, focus on giveaway-related keywords like "giveaway", "win", and "prize". With PyLucene, utilize its default settings for indexing and searching tweets. Meanwhile, with BERT, convert the keywords to

embeddings and seek the nearest matches. Then manually review the top 10 results for both tools to assess which one retrieves the most relevant tweets for the giveaway campaign.

Case 1: where BERT outperforms PyLucene: For instance, consider the following two tweets:

"I just won the giveaway! So excited to receive my prize!"

"I didn't win the giveaway, but congrats to the lucky winner!"

Both tweets contain the keyword "giveaway," but have different sentiments and meanings. Lucene may struggle to distinguish between these two tweets based on keyword matching alone. On the other hand, BERT's contextual understanding of language could help it recognize the differences and return more relevant results. BERT may be able to identify the first tweet as a positive and relevant result for the giveaway campaign, while the second tweet may not be as relevant to the campaign's objective.

Case 2: where PyLucene outperforms BERT:

In retrieving tweets that contain exact or partial matches for the specific campaign keywords.

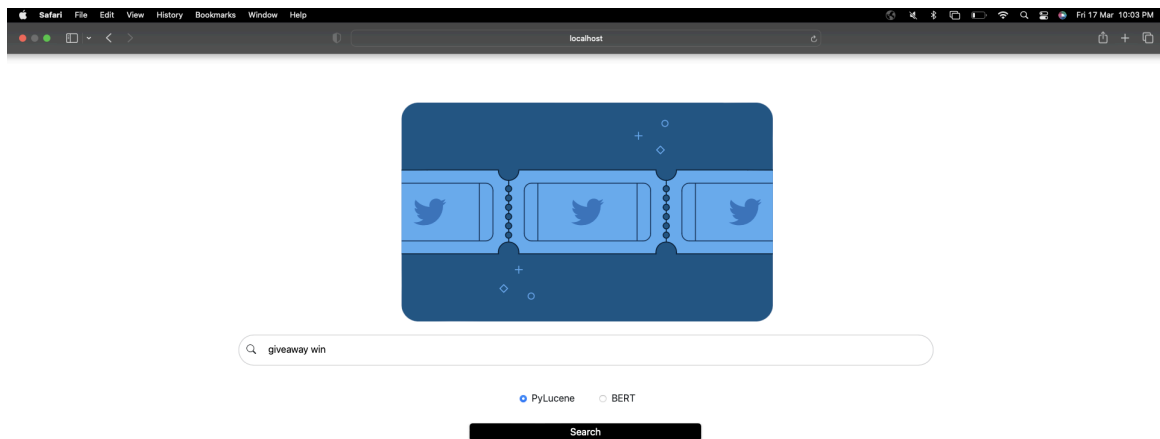
For instance, if the company wants to retrieve tweets that contain the exact phrase "Win a trip to Hawaii" in the tweet, PyLucene may be better equipped to retrieve tweets that contain the exact phrase, while BERT may retrieve tweets that are semantically similar but do not contain the exact phrase. The Runtime of PyLucene for 10k tweets for example runs faster in ms compared to BERT which takes 90 seconds and more as the number of tweets increases.

Web Application Implementation

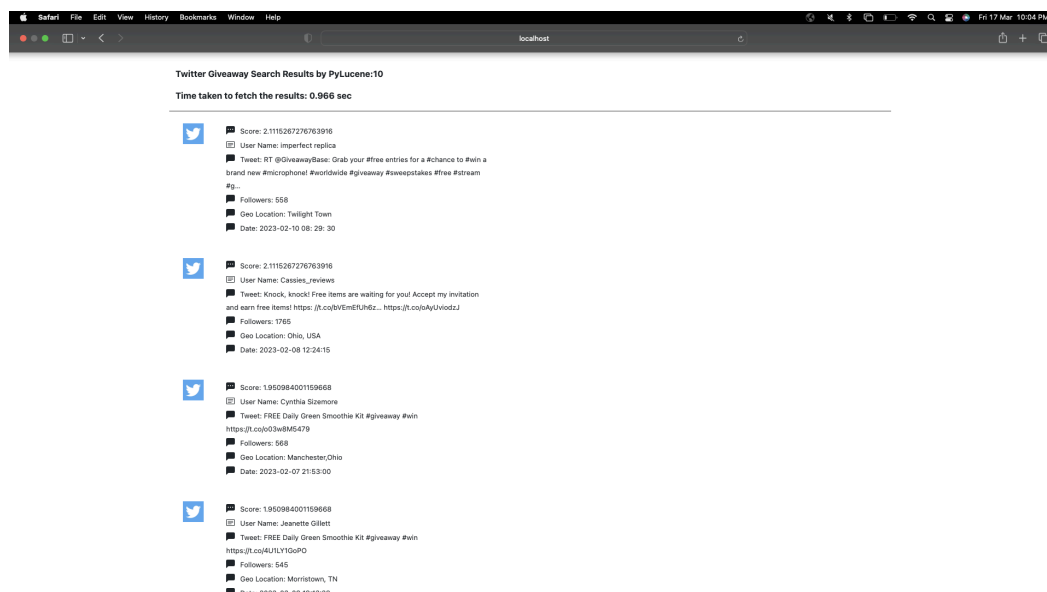
Frontend:

For front-end implementation, Angular 13 is used. Angular was chosen because of its component-based architecture and extremely quick rendering method. To make the program scalable, modular, and extendable, and for its improved user interface, I have developed several components.

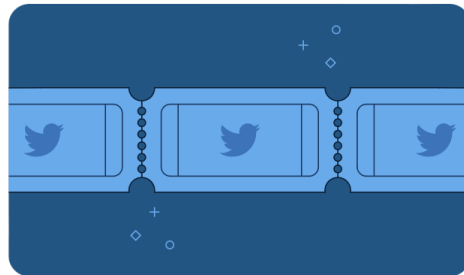
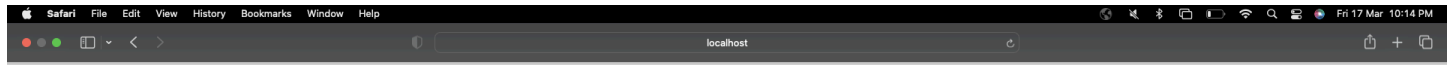
PyLucene:



The landing page has been maintained as straightforwardly as Google's, making it possible for users to easily search events.



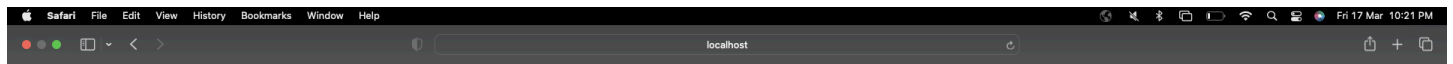
BERT:



Q giveaway win







☐ PyLucene ☒ BERT

Search



Twitter Giveaway Search Results by BERT:20

Time taken to fetch the results: 39.66 sec

-  Distance: 0.1112
Ranking: 1
Tweet: RT @LyndaCheckley: 🎉🎉 GIVEAWAY🎉🎉 The Delegate by Ali Carter As part of the @RandomTours book tour you can win a copy thanks to @matadorb...
-  Distance: 0.1456
Ranking: 2
Tweet: Bookish Besties Galentine's Giveaway! #giveaway #win
<https://t.co/FtbKAzzKFV>
-  Distance: 0.1509
Ranking: 3
Tweet: Looking to try your hand at #investing? Stop by my blog to read an #Excerpt from the #nonfiction AVOIDING SWINDLERS... <https://t.co/AUyb3acuHl>
-  Distance: 0.1542
Ranking: 4
Tweet: Have you entered this #Competition to #Win yet? If you refer friends you get more chances to win :)... <https://t.co/2vOJueJDz>
-  Distance: 0.158
Ranking: 5
Tweet: RT @ScotlandYardCSI: It's #GIVEAWAY Time! 🎉 !! Open 🌐🌐🌐
WW #Win a "signed" copy of Avenue The Dead by Jackie Baldwin @JackieMBaldwin1...
-  Distance: 0.158
Ranking: 6
Tweet: RT @ScotlandYardCSI: It's #GIVEAWAY Time! 🎉 !! Open 🌐🌐🌐
WW #Win a "signed" copy of Avenue The Dead by Jackie Baldwin

Users must enter the event's keywords, such as "giveaway win" , choose PyLucene or BERT as their preferred search engine by checking the appropriate box, and then click the Search button.

For PyLucene:

Point 1: Display the number of search results and the retrieval time.

Point 2: The information emphasizes the Score, Username, Tweet (original/RT), Followers, Geo-location, and Date.

For BERT:

Point 1: Display the number of search results and the retrieval time.

Point 2: The information emphasizes the Distance, Ranking, and Tweet (original/RT)

Limitations of the system

There are limitations to the project such as:

1. Only limited hashtags were crawled. There are a lot of other giveaways that happen, these hashtags could be used to pull tweets from Twitter too.
2. The project is only limited to Twitter, other Social Media platforms don't have developer-friendly APIs such as Twitter, but if present, they should also be considered.
3. The system can also be limited by privacy settings, which may prevent specific tweets from being searched even if they exist.

Instructions on how to deploy the system

- **Instruction on how to deploy the crawler:**

A crawler.sh executable file is included that takes as input all necessary parameters

```
MacBook-Pro:TwitterDataCrawler sairuthikovur$ sh crawler.sh 200000 'output/'
```

- **Instruction on how to build the PyLucene index:**

An index.sh executable file is included that takes as input all necessary parameters.

- **Instruction on how to build the BERT index:**

Install the necessary modules like transformers, folium, etc.

A bert.sh executable file is included that takes as input all necessary parameters.

- **Instruction on how to deploy the User Interface:**

- Install all the required packages by executing "npm install" for running Angular JS (Version 13) application.
- Run the webapp.sh file to execute web applications.

- Implemented a web application that inputs a query and an index choice (PyLucene or BERT) and outputs the top-k results.

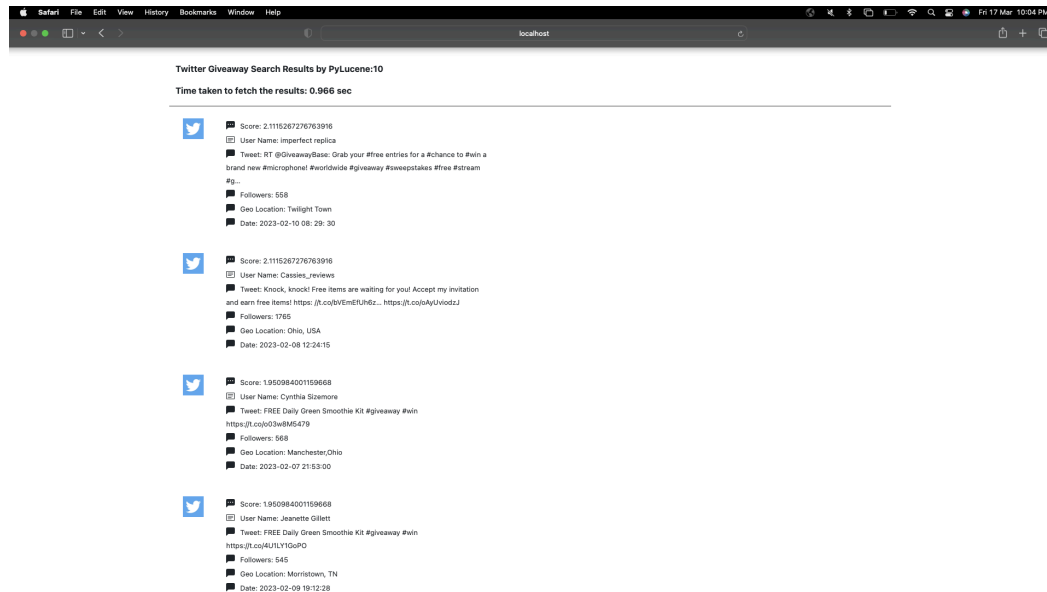


Fig: PyLucene Web App Results

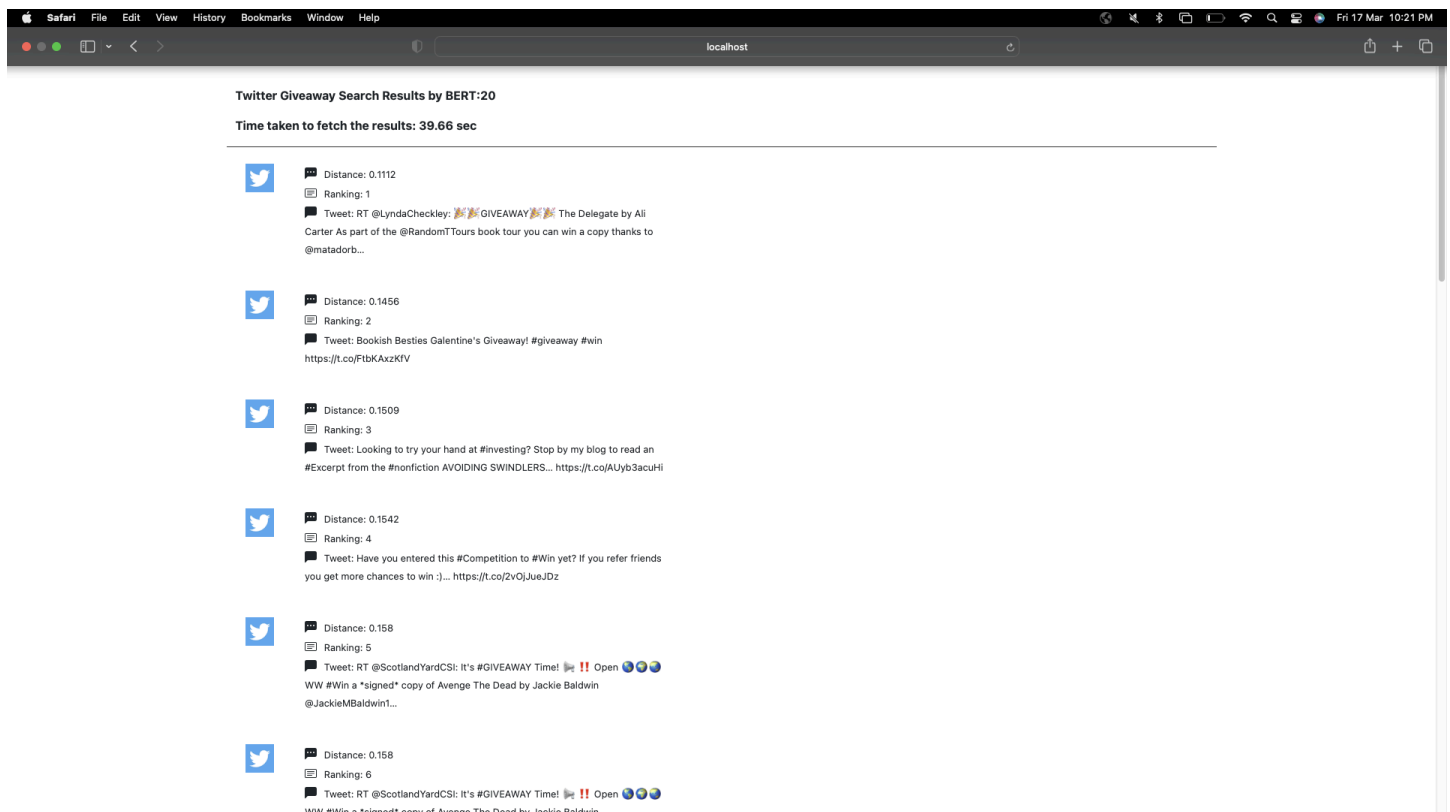


Fig: BERT Web App Results

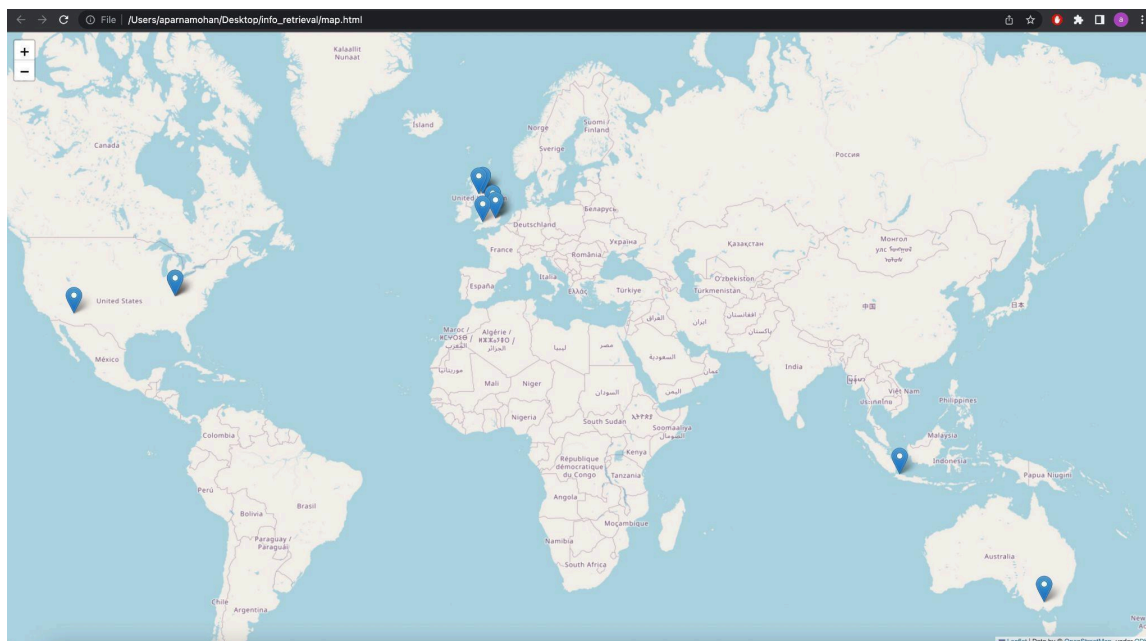


Fig: Map Visualization of tweets (BERT indexing)