THE SOLR'S : A COMPREHENSIVE TWITTER SEACH ENGINE
CSE 535 FALL 2021 PROJECT

Department of Computer Science and Engineering
University at Buffalo, The State University of New York
NY - 14214

ABSTRACT

The goal of this project is to build and implement a full text based retrieval of relevant documents like Tweets in the case of this project. We provide our search engine name Solr's indexed with more than sixty thousand relevant documents across multiple person of interests and the views from the general population in order to understand the view of the above mentioned demographics about the covid pandemic, vaccine and vaccine hesitancy in general.

1 INTRODUCTION

The Solr's present a full text based retrieval system for tweets related to the COVID pandemic and the vaccination drives further in three major economies namely United States of America, India and Mexico. Given the vast scale of the efforts to stop the spread of COVID and the vaccination drives, these three countries present a very effective data-set of vaccine and pandemic related sentiments both for and against to build analytics around what the people in power and people in general think and have done for their countries respectively. With more than 100 thousand documents that include tweets and replies across two cores we have gathered a rather extensive data of most of the topics surrounding the aforementioned subjects we try to address here.

2 METHODOLOGY

We begin by first collecting raw data. Our data-set of documents contains tweets from around 19 person of interests spread across 3 countries with thousands of tweets from the general population or citizens of those countries. The dataset also contains tweets from a curated list of keywords related to COVID and vaccinations. The collection of tweets was done adhering to Twitter privacy and developer account guidelines through their API v1 access documented by the tweepy API. The tweet collection process is continuous and the index gets updated every 5 minutes(also to avoid rate limit errors) till the probable submission/evaluation day.

The tweets as retrieved are pre-processed and cleaned using a set of functions that remove all the trailing white-spaces, they are then separated into fields relevant to our search engine and also as shown in a image below. With more than 10 fields we were able to segregate the data quite well to facilitate our search. Tweets were also cleaned off of the emoticons that are widely used nowadays to associate a tweet with the tweeters emotion and stored in a different field. Once done, the tweets were then indexed to Solr using the pySolr API with a defined schema indexing. The fields included different classes of filters for all the languages that we used for this search engine so that the data is well tokenized in it's respective language field for better search results. With experiments we found out that BM25 model performed well for us and decided to go ahead with it.

Field	Type	Remarks
id	string	mandatory
country	string	mandatory
tweet_lang	string	mandatory
tweet_text	text_general	mandatory
text_xx (where xx=hi, en, es)	text_xx (where xx=hi, en, es)	mandatory
tweet_date	pdate	mandatory
verified	boolean	mandatory
poi_id	plong	mandatory in POI
poi_name	string	mandatory in POI
replied_to_tweet_id	plong	mandatory in replies
replied_to_user_id	plong	mandatory in replies
reply_text	text_general	mandatory in replies
hashtags	strings, multivalued	required when present
mentions	strings, multivalued	required when present
tweet_urls	strings, multivalued	required when present
tweet_emoticons	strings, multivalued	required when present
geolocation	strings, multivalued	required when present

Figure 1: Fields indexed in Solr for any particular tweet

The relevant documents have not been displayed in the order they were retrieved. Instead, our ranking system will display relevant retrieved documents based on the the user's query. This is done using the Solr's inbuilt relevancy scoring of documents based on your search criteria and query parsers like Lucene which is the default one followed by dismax and edismax. After running multiple queries through different parsers we happen to choose between Lucene or edismax. The User Interface also plays a key role in building a search system. Our UI has been designed to provide user some statistics such as relevant tweets on the top with sentiment per tweet, news related to the topic the user has searched and the general statistics around the search terms such as a word-cloud of the words most frequently used in the tweets and a pie chart showing the tweet distribution. With advanced filters we give the power to the user to refine their search query by helping them filter through a list of important person of interests, country and language making it a fully reliable experience.

A minimalist UI as shown below helps user focus on the search more rather than figure out how to.

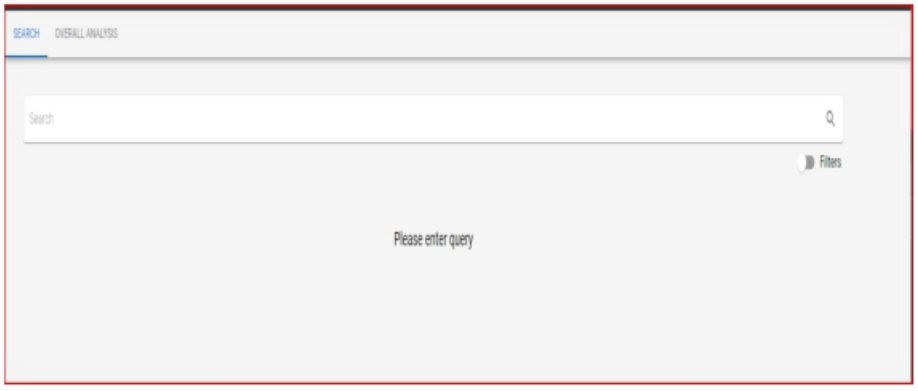

Figure 2: Fields indexed in Solr for any particular tweet

2.1 TECHNOLOGY STACK

The technology stack used for this project was decided taking into consideration speed, ease of deployment and familiarity of the languages and frameworks among the teammates.

Frontend : ReactJs, Google React charts, deployed on AWS Am
Frameworks and Services : Flask, JSONify, Tweepy, PySolr, TextBlob, NLTK, Matplotlib, Numpy, SSLify.
BackEnd : Solr, EC2, Flask

2.2 FUNCTIONALITY AND USER EXPERIENCE

A single text box as shown in Figure 2 is all you need to begin your search into a world of tweets. TO enhance this experience we provide the user with filters like those shown in the figure below.

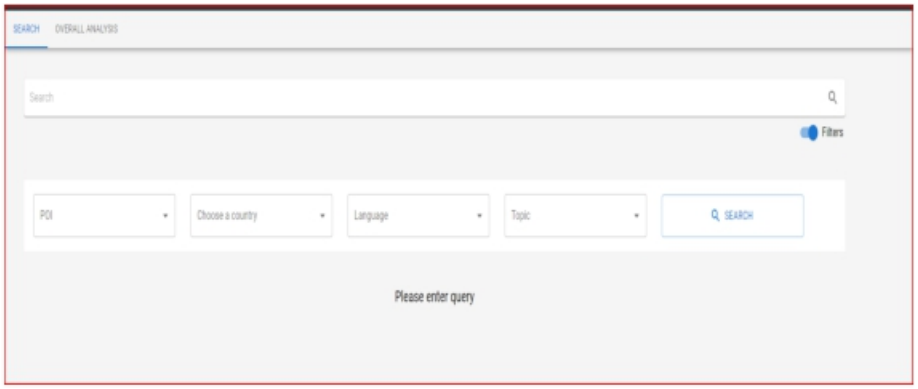

Figure 3: Fields indexed in Solr for any particular tweet

After the search term is performed the user is presented with a view as shown below.

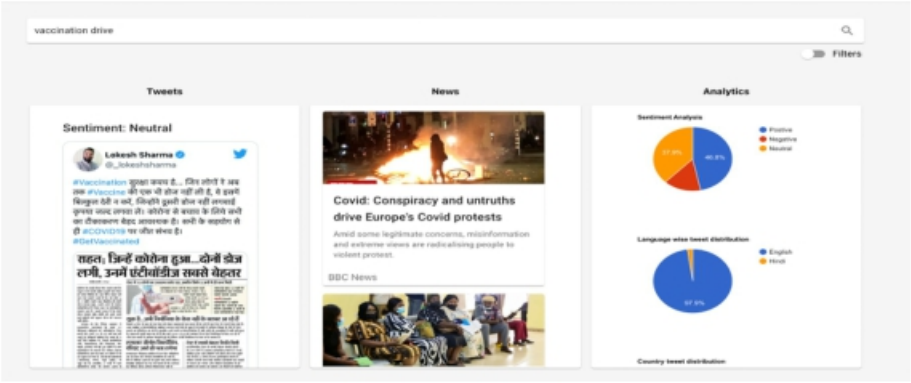

Figure 4: Fields indexed in Solr for any particular tweet

Whenever the user inputs a search query and adds/removes filters of his choice all the data collected as part of the query is sent to a Flask server running on a separate EC2 instance in JSON format. We then extract relevant query terms from the request and begin processing it. Once we get the query term and the POI filter the user is specifically trying to search tweets of we construct the final URL that is sent out to the Solr back-end served on a different EC2 instance. The URL retrieves all the relevant documents from the Solr index and we return the results in a JSON format to the front-end. The user sees a graphically enriching result of his search query term with tweets displayed as they are in the native twitter app alongside the latest news fetched form the internet regarding that particular topic and analytics like a wordcloud of most frequently used words, tweet distribution and many more.

Moreover as a part of transparency over collected tweet data the user is also shown a analytical view of distribution of tweets across various pie charts like tweet distribution over COVID or VACCINE or general rhetoric alongwith a pie chart displaying POI vs general population tweets and language

wise distribution of overall tweets. A glimpse of this display of data can be seen as in the figure below:

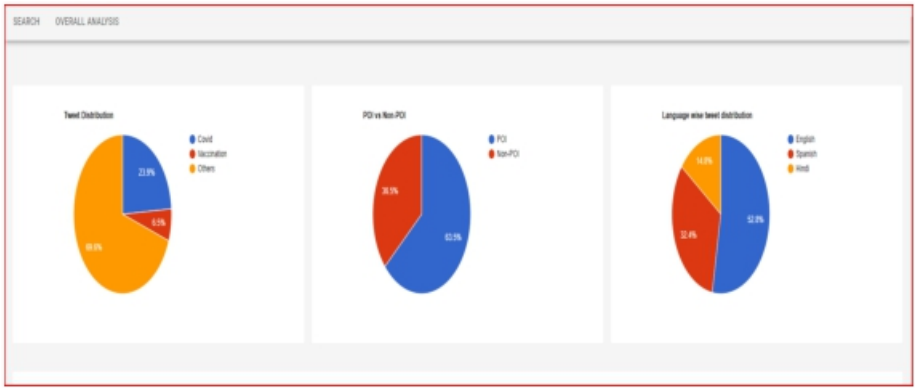

Figure 5: Fields indexed in Solr for any particular tweet

Furthermore, two charts below that depict the tweet distribution per person of interest and a map displaying a heatmap of the tweet collected country wise to give a better view and understanding of collected data.

Figure 6: Fields indexed in Solr for any particular tweet

You can also fetch the tweet sentiment per POI and see the language wise distribution of the particular POI's tweets to get an idea of how much a particular person of interest has talked about covid and vaccine and also other topics in general. For the overall analysis a wordcloud generated gives you a brief look into the most frequently used words by that particular POI.

3 Team Contributions

Rugved Jaysing Thorve: Developed the UI and work on request sending from frontend to the Flask endpoint, sentiment analysis, overall analysis, report.

Ashwin Vinay Phadke: Worked on the Flask endpoint, deploying applications, sentiment analysis, overall analysis, Solr schema, report.

Shashank Desai: Worked on indexing all the documents for the tweets and maintaining the index with a considerable corpus full of tweets and replies, keeping the Solr backend running error-free.

Swapnil Sanjay Satpute: Worked on indexing tweets and replies and managing Solr cores, keeping the Solr backend running error-free.

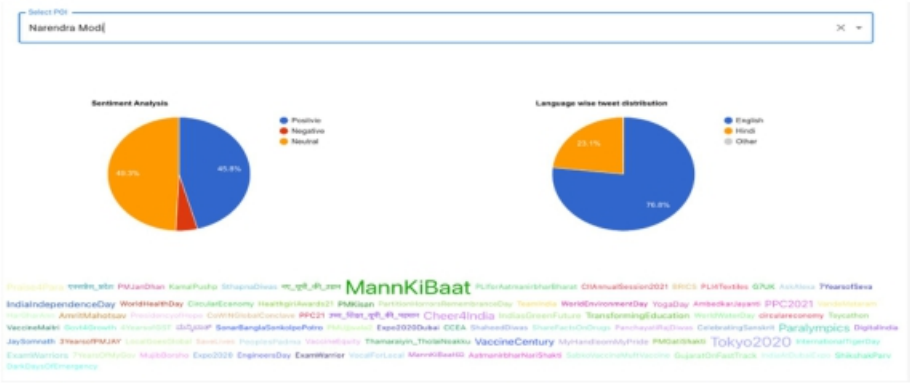

Figure 7: Fields indexed in Solr for any particular tweet

4 Remarks

Basic Requirements:

- Completed and successfully analyzed and displayed results for the attitude of general population vs POI towards COVID vaccines.See dashboard the part that we could not complete was analyze the exact attitude towards only covid due to lesss number of covid tweets over general population but considerable vaccine/covid vaccine tweets.
- Analyzed the impact of COVID related political rhetoric on masses through news and political tweets sentiment analysis.
- Learned and understood the basics of building a search engine UI through an end-to-end IR system.
- Built a comprehensive web UI to facilitate document search.

Bonus Requirements:

- Analyzed excerpts of vaccine hesitance through tweet sentiment analysis and display the proper label over tweets that show hesitancy over vaccine as negative.

5 Results, Analysis and Conclusion

Analysis:

- Top results are boosted and the probability of seeing the most relevant tweets is higher as the user uses more advanced search filters.
- The fields are tokenized well to handle multi-lingual queries however it has been observed that some times multi word queries don't seem to work but it does most of the times.
- Analysis shown is dyanamic to the content being indexed constantly to the core and always shows the most recent results.

Limitations:

- Only retrieve a maximum of results before the retrieval starts to slow down and takes time to load.
- Rely on news api with limited number of queries per day due to trial version.
- Spell check and multilingual and multi word queries aren't handled properly some times due to rely on some 3rd party API.
- Limited number(only a hundred thousand) of tweets per section/part of tweet collection framework due to constant rate limitations from the twitter API especially while collecting replies.

- HTTPS vs HTTP request processing conflict however handled may render a query on load sometimes but returns the results most of the times.

Conclusion: As a part of the project throught the CSE 535 course the team was well able to understand how a search engine works end-to-end and appreciate the work of thousands working across these domains. The course TA's throughout all the projects provided excellent boilerplate codes for all the projects that helped us quickly get started with all of the projects. Course material by the professor proved extrememly useful in implementing the concepts required for a full blown search engine.

6 REFERENCES

- https://solr.apache.org/
- https://github.com/django-haystack/pysolr
- https://aws.amazon.com/console
- https://reactjs.org/
- https://flask.palletsprojects.com/en/2.0.x/
- https://textblob.readthedocs.io/en/dev/
- https://matplotlib.org/
- https://numpy.org/