**UNIVERSITY AT BUFFALO**

**THE STATE UNIVERSITY OF NEW YORK**

**INTRODUCTION TO INFORMATION RETRIEVAL**

**CSE 535**

**FALL 2019**

**PROJECT 4**

**ANALYZING THE IMPACT OF POLITICAL RHETORIC IN TRADITIONAL AND SOCIAL MEDIA**

**SUBMITTED BY:**

**NAME: AMAN GARG, SOURABH MAJUMDAR, SWAPNIL KISHORE**

**UBITNames: amangarg, smajumda, kishore2**

**1. INTRODUCTION**

In this project a search application is implemented which runs on its own custom database of tweets taken from famous political and media personalities from three countries namely India, United States of America and Brazil. This search application is used to return tweets based on a custom user query and rank them in order of decreasing with respect to the query. Furthermore, a visualization page is also created where, through various graphs and diagrams, the nature of these tweets and their impact is represented which helps in fulfilling the end goal of the project which is to represent the impact of political rhetoric on social media.

**2. PROPOSED METHOD**

The tweets collected are indexed in an Elasticsearch instance. The visualizations are created using the Kibana tool and the search application is created using the flask framework for python. Once a user inputs a query in the search bar of the application the top ten most relevant tweets for that query are displayed and upon clicking the visualization button, various analytical graphs of the tweets with respect to the query can be seen.

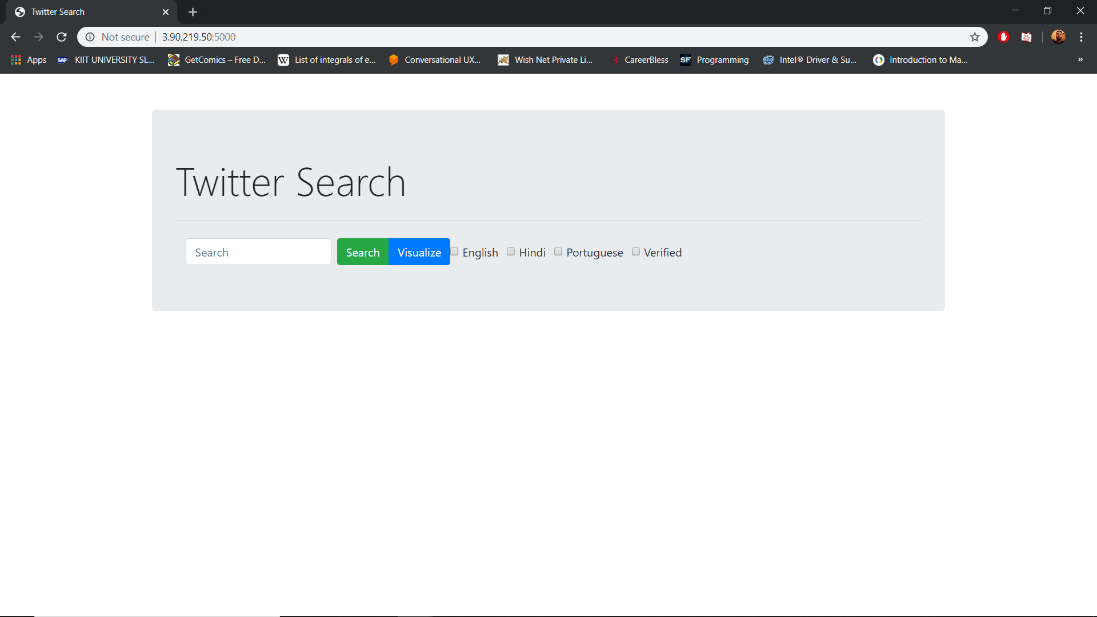


Fig 1: UI of the Twitter Search Application.

**3. TOOLS USED**

**3.1 ELASTICSEARCH:**

Elasticsearch is a free, open-source, RESTful, distributed search and analytics engine built on Apache Lucene. It is different in some ways and similar in some to Solr which has already been used in a previous project for this course. It was released in 2010. Similar to Solr, data can be indexed to elasticsearch in the form of JSON documents. A key feature of elasticsearch is its distributed nature which allows it to process large volumes of data in parallel, quickly finding the best matches for the user’s queries. Its support for various languages such as Java, Python etc. make it convenient for application development. Amazon Web Services offers Amazon Elasticsearch Service delivers elastisearch as a service which allows users to easily understand, access and utilize it.

**3.2 KIBANA:**

One of the major advantages of using Elasticsearch is that it comes integrated with Kibana, a popular visualization tool. Kibana helps in easily visualizing elasticsearch data and also allows tracking query load. The amount of visualization tools available for use in Kibana are several such as histograms, line graphs, pie charts, time-series, sunbursts and many more. Just like elasticsearch, kibana is also open-source. It uses ‘dashboards’ for quick and easy creation of representations of data. Once created, these visualizations reflect the changes to elasticsearch queries in real time.

**3.3 FLASK:**

Flask is a Python web framework built to enable easy creation of web applications using Python. It is classified as a microframework because it does not require a particular set of tools and libraries. It was created by Armin Ronacher nearly ten years after the creation of Django. Using the various tools and libraries that flask provides, it is fairly simple to create a basic flask application with the ability to scale it to complex applications as well. In this project, flask is used to create a web search application that runs queries on an elasticsearch database and returns results for the query.

**3.3 TEXTBLOB:**

Textblob is a python library for processing textual data. It is built upon NLTK and provides and easy to use interface to the NLTK library. Usage of textblob is pretty simple as the document is first passed as a parameter to the TextBlob method. After tokenization, a polarity score is assigned to each token of the document. The overall polarity score of the document is the sum of all the polarity scores of the tokens of the document. In this project, a polarity score is generated for each tweet and if the score is greater than 0, the tweets is considered positive, if the score is less than 0 the tweet is considered negative and if the score is equal to 0 the tweet is considered neutral.

**4. ELEMENTS OF SEARCH APPLICATION**

**4.1 FACETED SEARCH:**

Faceted search, also sometimes referred to as guided navigation, is a technique which involves augmenting traditional search techniques with a faceted navigation system, allowing users to narrow down search results by applying multiple filters based on classification of the items. It broadly entails analyzing a large set of content and excluding any objects that don’t meet certain criteria. Faceted navigation is different from filters in one key aspect that it includes multiple filters for many different attributes of the objects in a set. The web application created here enables the user to perform faceted querying and search to some extent.

1. **English:** This filter is applied on the language attribute of the tweets and the criteria applied is that the tweets be of English language
2. **Hindi:** This filter applies the criteria that the tweets be of Hindi language.
3. **Portuguese:** This filter applies the criteria that the tweets be of Portuguese language.
4. **Verified:** This filter applies the criteria that the tweets returned be from a verified twitter user.

**4.2 SEARCH BAR AND BUTTON:**

The search bar is where the user enters the query and clicking the search button retrieves the top 20 results relevant to the query.

**4.2 VISUALIZE:**

Clicking the visualize button redirects the user to a Kibana dashboard which contains all the visualizations for the tweets that are relevant to the query entered by the user.

**5. SCORING IN ELASTICSEARCH**

By default Elasticsearch uses the Okapi BM25 model to rank results. For multi-term queries, the standard scoring algorithm used by elasticsearch is called the Practical Scoring Function. This combines the Boolean model, TF/IDF and the vector space model to score documents. When a document matches a query, the score for that document for that query is calculated by combining the scores of matching term. The formula for practical scoring function is given as:

score(q,d) =

queryNorm(q)

· coord(q,d)

· ∑ (

tf(t in d)

· idf(t)²

· t.getBoost()

· norm(t,d)

) (t in q)

Here,

1. score(q,d) is the relevance score for document d for query q
2. queryNorm(q) is the query normalization factor
3. coord(q,d) is the coordination factor
4. tf(t in d) is the term frequency for term t in document d
5. idf(t) is the inverse document frequency of term t
6. t.getBoost() is the boost (if any) that has been applied to the query
7. norm(t,d) is the field-length normalization combined with any index-time field level boosting.

**6. CONTRIBUTIONS**

This project was made by a team of 3 members: Aman Garg, Sourabh Majumdar and Swapnil Kishore. The UI of the application was designed by Aman using Flask framework in Python. The sentiment analysis was performed by Swapnil and the gathering of news articles using Bing API was done by Sourabh. All team members worked together to index the data and create the visualizations on Kibana.

**7. REFERENCES**

1. <https://www.elastic.co/guide/en/kibana/current/introduction.html>
2. <https://www.fullstackpython.com/flask.html>
3. <https://textblob.readthedocs.io/en/dev/>
4. <https://docs.aws.amazon.com/elasticsearch-service/latest/developerguide/es-gsg-create-domain.html>
5. <https://qbox.io/blog/optimizing-search-results-in-elasticsearch-with-scoring-and-boosting>