CSE 535 Fall 2018

Project 4 – Report

Complete Search and Analytics Solution

Based on Dissecting Twitter Data

Team Name: Team KenSpring

Project Name: TwiFi

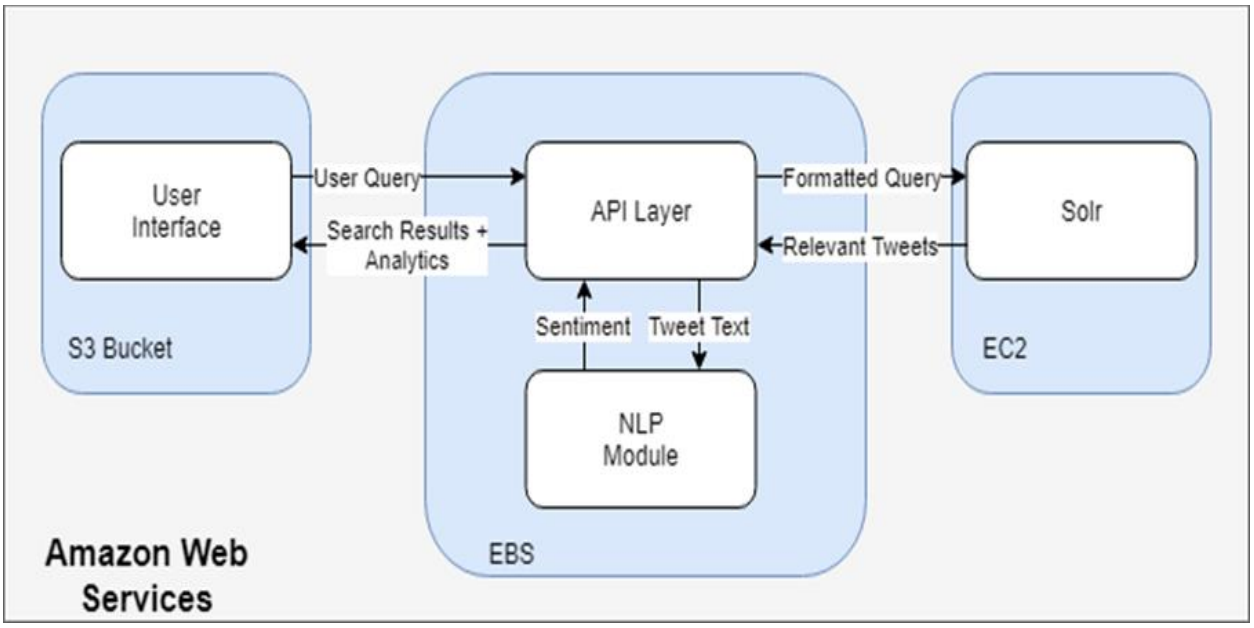
Project URL: https://goo.gl/PmkesU

Name	UBIT Name
Linus Castelino	linuscas
Anand Gangar	agangar
Ram Singh	ramvalla
Harsh Gandhi	harshnar
Yudhister Singh	yudhiste

Abstract

In this project, we have built a web application called TwiFi as a product that provides insights about various topics and other statistical information related to it by analyzing a corpus of 400,000 plus tweets. The corpus contains tweets pertaining to 5 distinct topics viz. Politics, Environment, Crime, Social Unrest and Infrastructure from 5 distinct cities viz. New York, Mexico, Delhi, Bangkok and Paris in 5 languages viz. English, Spanish, Hindi, Thai and French. TwiFi captures the user query, retrieves the relevant search results, performs sentiment analysis on the retrieved results and displays it to the user along with a statistical view of the retrieved results. It also provides the user with facetted search options. TwiFi proves to be an end-to-end solution for searching tweets and performing analysis on the specific search query.

System Architecture



The application setup follows the microservices architecture. The complete application is deployed on Amazon Web Services.

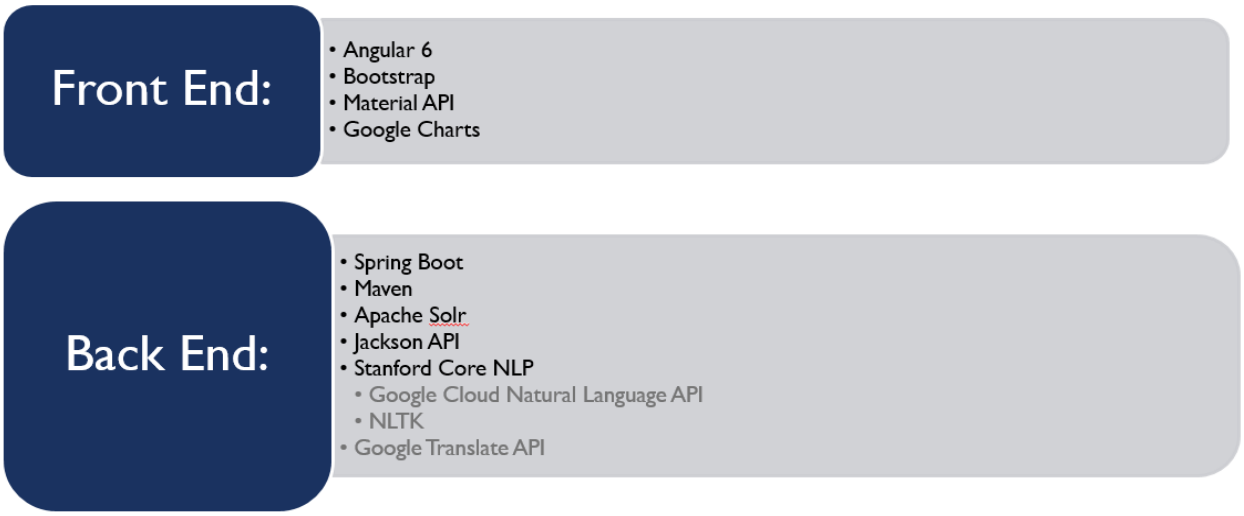
The User Interface layer is the user-facing component that captures the user query and passes it on to the API layer. The User Interface is deployed on an S3 Bucket as a static website. On receiving a response to the search query from the API layer, it displays all the retrieved content and also renders a graphical view of the statistical data.

The API layer is clubbed with a natural language processing module and is deployed using Elastic Beanstalk. It receives the user search query from the API layer and queries the Solr instance after adding the necessary filters and formatting options to the search query. On receiving a response from Solr, it performs sentiment analysis on each retrieved results and returns a sentiment score for the same. Jackson API is further used to convert the Solr response to a java object which is further pre-processed to a specific data format used for plotting statistics using Google charts.

Solr is configured to use Okapi BM25 model to retrieve top relevant results with respect to the query received from the API layer.

The application performs sentiment analysis on the every search result at runtime and returns the computed result asynchronously to the User Interface.

Technology Stack



- **Angular 6.0:** Google's Angular 6 was used as the framework for developing the user interface.

- **Bootstrap & Material API: F**ront-end frameworks used to enhance various UI components.

- **Google Charts:** We used Google Charts API to render the statistical information returned by the API layer in a visually appealing manner.

- **Spring boot**: Spring framework module which provides features for quick API development.

- **Apache Solr:** An open source enterprise search platform, written in Java used for full-text search, hit highlighting, faceted search, real-time indexing, dynamic clustering, database integration, NoSQL features, and rich document handling.

- **Jackson API:** Used for as an object mapper to convert the JSON responses from Solr into Java object.

- **Stanford Core NLP:** Used to determine the sentiment of the tweet text.
  Note – We began our application development with Google Cloud Natural Language API but later realized that it does not support Hindi and Thai languages. We then worked upon NLTK for sentiment analysis but did not continue with it since the tool was compatible only with Python and R.

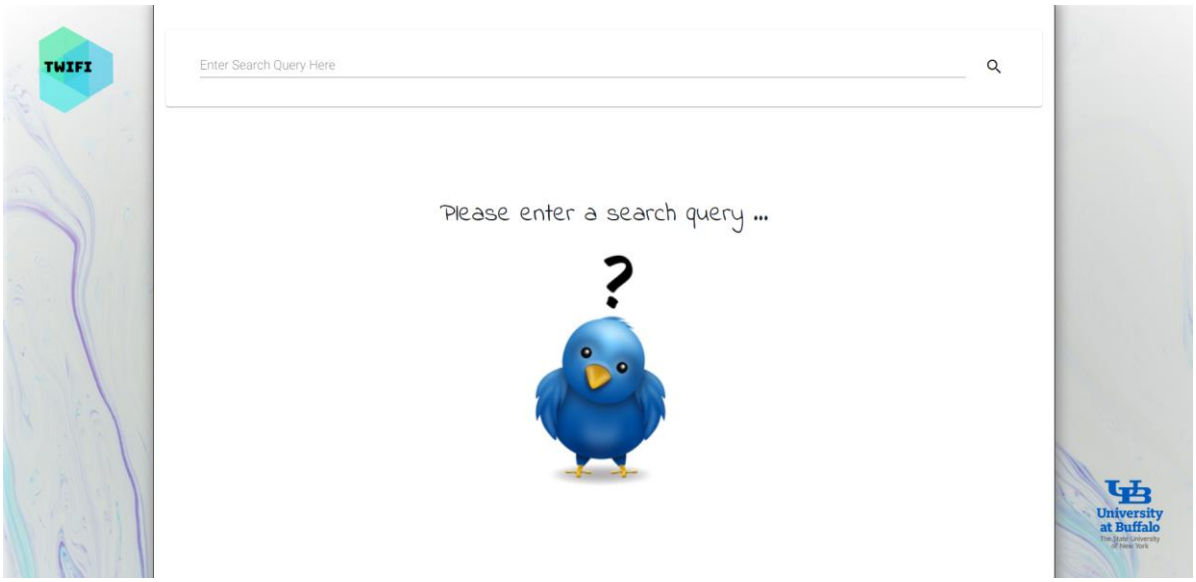
- **Google Translate API:** Translation API that was used to translate tweets into different languages before querying Solr. It was observed that when Solr was queried with translated search queries, the number of relevant results reduced drastically due to which the translation API was later commented out from the deployed version of the application.

Member Contributions

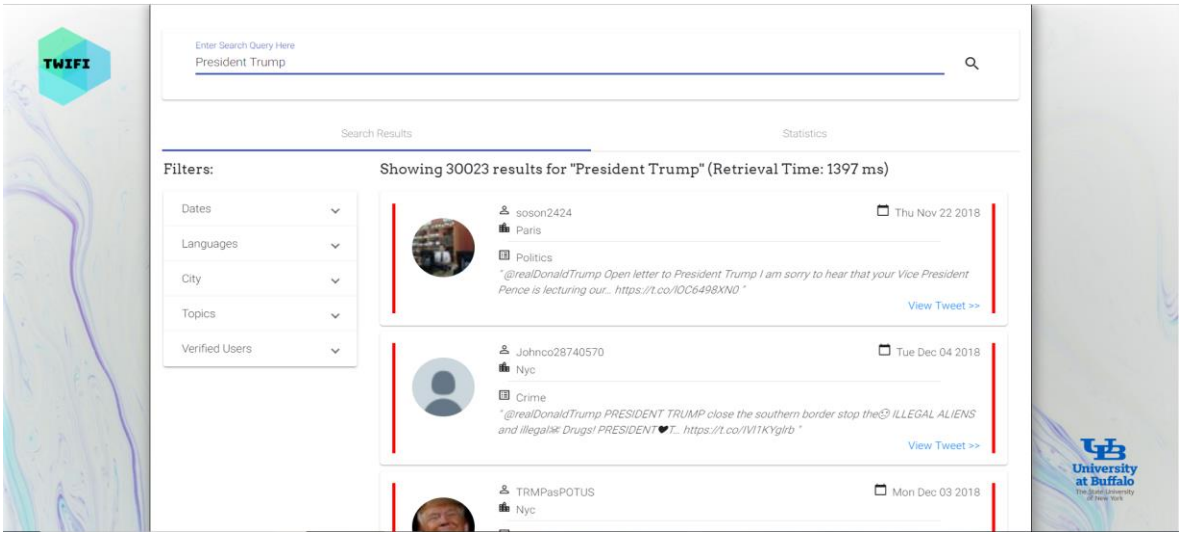
Linus Castelino – Application Architect, UI Development (60%)
Anand Gangar – UI Development (40%), Solr Administration (80%)
Ram Singh - API development (50%), Solr Administration (20%)
Harsh Gandhi – API development(50%), Sentiment analysis using Stanford Core NLP (100%)
Yudhister Singh – Sentiment Analysis using NLTK (not included in the application)

Application Screenshots

Homepage:



Search Results page:



Statistics page: