Complete Search and Analytic Solution over Twitter Data

Team Mavericks- Project IV, CSE-535, Fall 2018

Group members:

Darshan Nevgi, 50291068
Sai Krishna, 50288219
Siddharth Pandey, 50288608
Sravan, 50291064
Venugopal Shah, 50291126

Contents

Introduction

For our final project, we have designed, built and implemented a complete search and analytic solution for a corpus of twitter data that has been crawled over a period of 4-5 weeks. This data has been indexed using the okapi BM25 model which is offered by Solr. The report will discuss this more in detail in the following sections. Finally, we have built a user interface that enables users to search and analyze the corpus. The UI has a simple and elegant feel to it, and allows the use of all the features in an uncomplicated way.

Highlights of our Solution

- User can do extensive filtering on tweets based on facets Date, hashtag, topic city
- The user can query not only on one such filter but across different filters. For example, a user wants to see tweets related to environment, and in their choice of city.
- Users can see in-depth analytics of their queries through various graphs. This helps in visualizing the trends of queries that users have entered.
- Users can check overall sentiment of their queries.

The major challenge of this project was to have a design that incorporates the basic features without congestion. A new user should have no problems in figuring out the workflow and should be able to make use of all the features offered.

Data Set

The data set consists of twitter data, specifically tweets, that have been crawled using APIs that twitter offers. These tweets are not just random data, but instead are specific to 5 cities, 5 different topics and 5 languages. These are as following:

Topics

- Environment
- Crime
- Politics
- Social Unrest
- Infrastructure

Cities

- New York City
- Delhi
- Bangkok
- Paris
- Mexico City

Languages

- English
- Hindi
- Thai
- French
- Spanish

Over a period of 4-5 weeks, we have been able to crawl about 150,000 unique tweets that are stored as .json files. The tweets were crawled using the twitter4J library with the Search API that twitter offers to developers.

Technologies Used

★ Solr

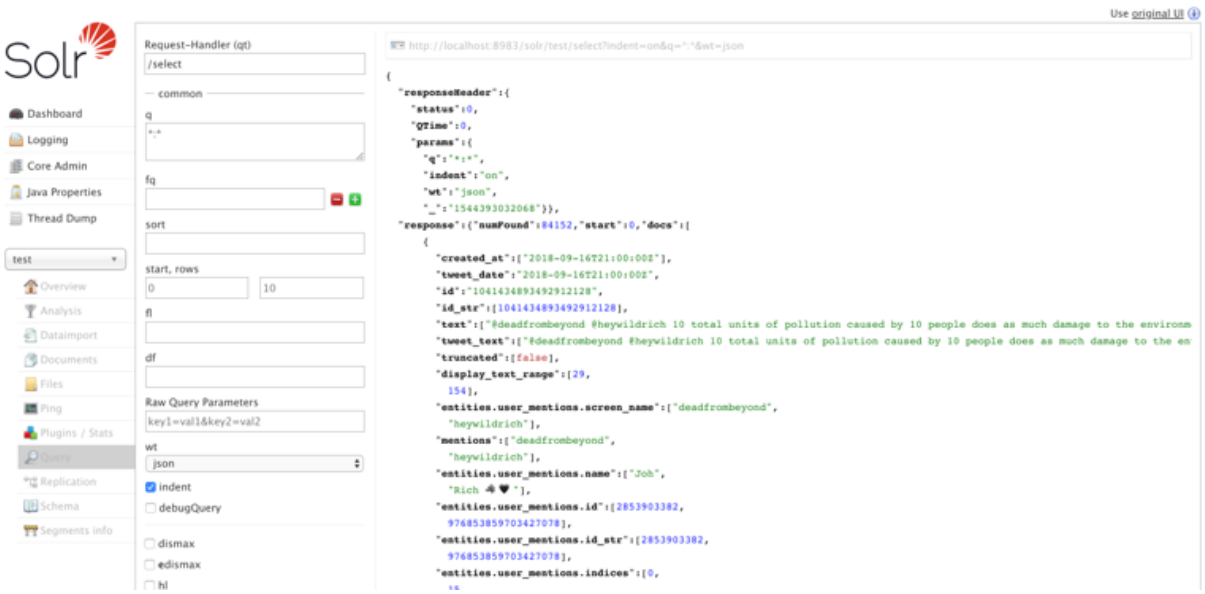


Figure: Screenshot of Solr host after indexing tweets

Solr is an open source enterprise search platform, written in Java, from the Apache Lucene project. Its major features include full-text search, hit highlighting, faceted search, real-time indexing, dynamic clustering and more. It is highly reliable, scalable and fault tolerant. It provides distributed indexing, replication and load-balanced querying. It is also capable of automated failover and recovery, centralized configuration and more. Solr powers the search and navigation features of many of the world's largest internet sites.

We use Solr's okapi BM25 model to index the tweets. Okapi BM25 (BM stands for Best Matching) is a ranking function used by search engines to rank matching documents according to their relevance to a given search query. It is based on the probabilistic retrieval framework developed in the 1970s and 1980s by Stephen E. Robertson, Karen Spärck Jones, and others.

★ JSP and JSTL

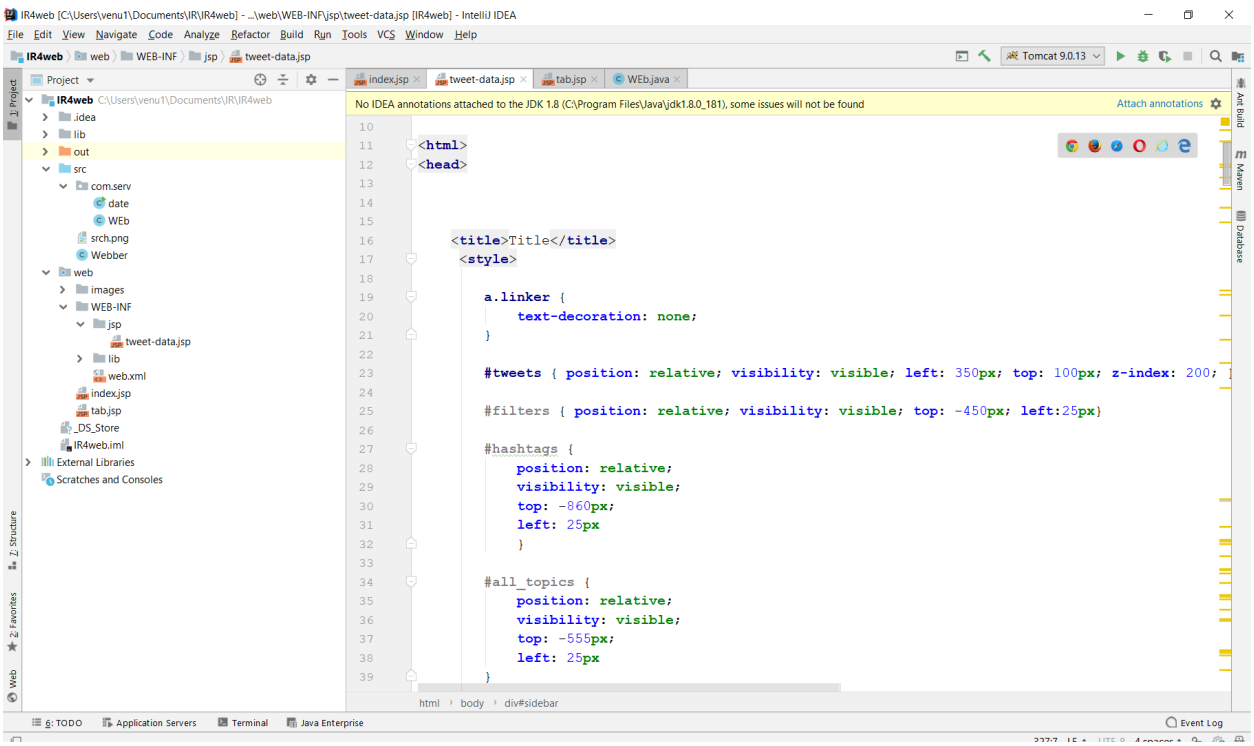


Figure: Screenshot to show implementation of HTML UI in a .jsp page.

Our complete UI is based and programmed on JSP and JSTL.

JavaServer Pages (JSP) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types.

The JavaServer Pages Standard Tag Library (JSTL) is a component of the Java EE Web application development platform. It extends the JSP specification by adding a tag library of JSP tags for common tasks, such as XML data processing, conditional execution, database access, loops and internationalization.

★ Google Charts

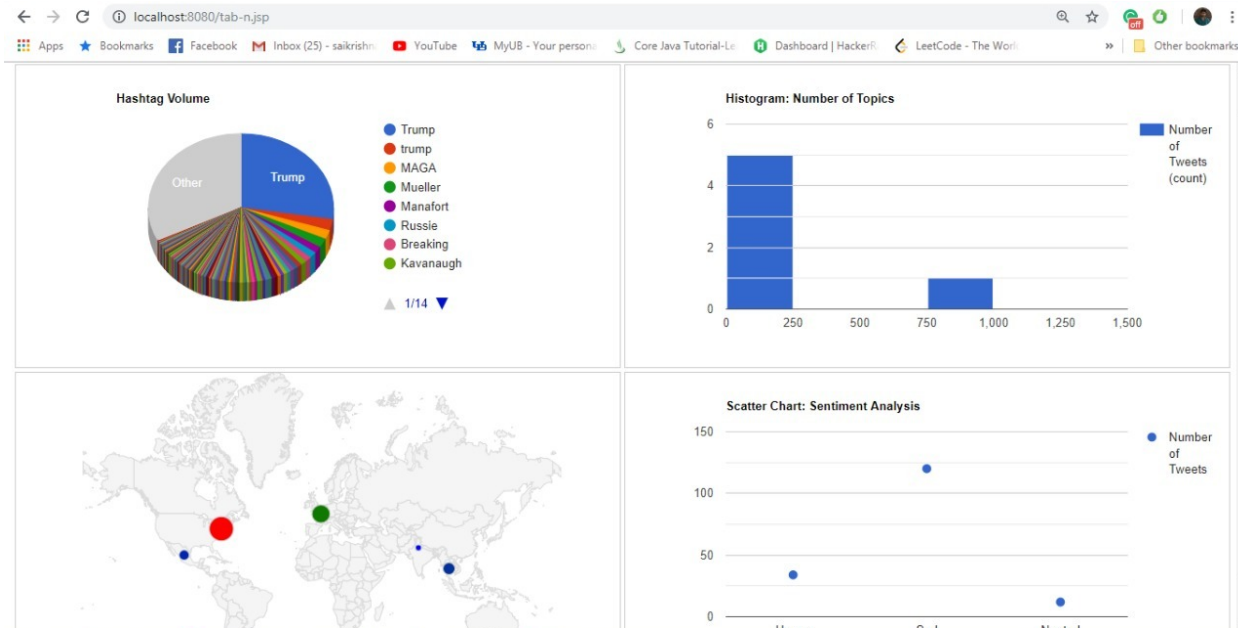


Figure: A screenshot of Google charts displayed on our web page with analytics. Here, we have the world map and the histogram as a form of representation.

Google Charts is an interactive Web service that creates graphical charts from user-supplied information. The user supplies data and a formatting specification expressed in JavaScript embedded in a Web page; in response the service sends an image of the chart.

★ Textblob

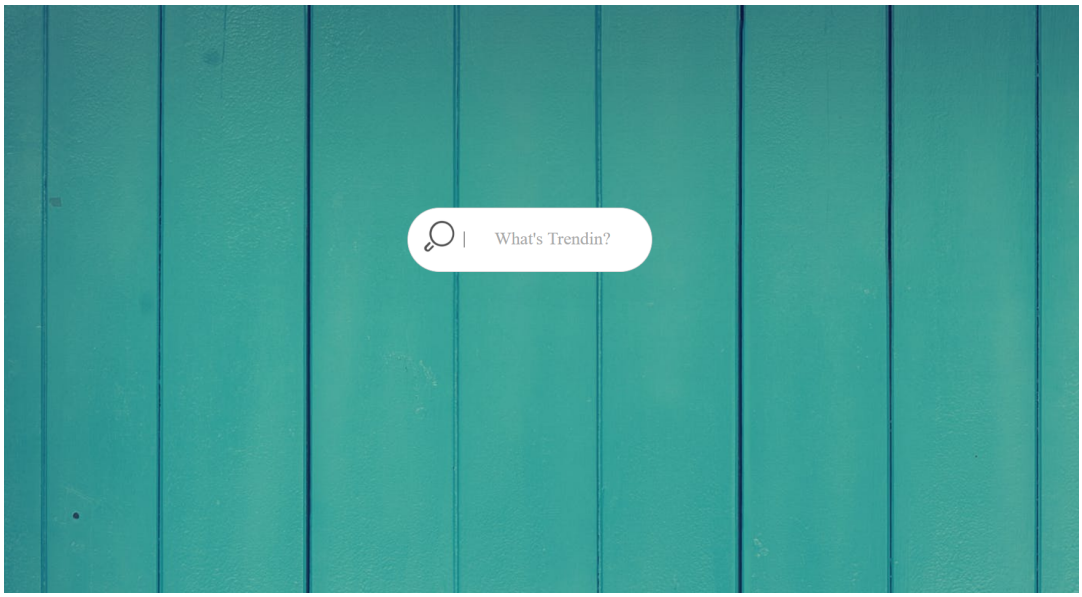
TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more. We have used this library for running a simple sentiment analysis on the user queries

★ AWS Elastic beanstalk

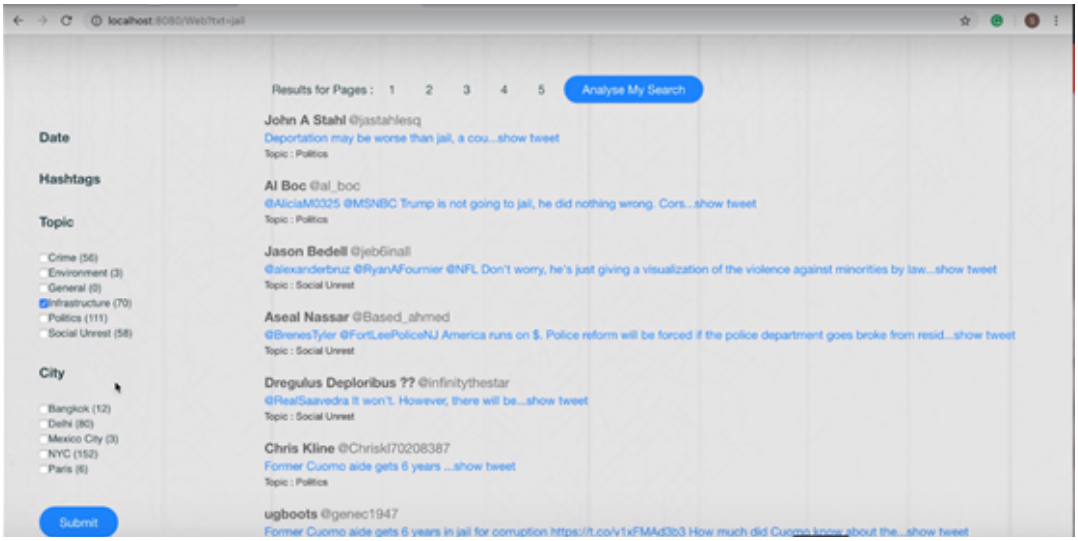
AWS Elastic Beanstalk is an orchestration service offered from Amazon Web Services for deploying infrastructure which orchestrates various AWS services, including EC2, S3, Simple Notification Service (SNS), CloudWatch, autoscaling, and Elastic Load Balancers. Elastic Beanstalk provides an additional layer of abstraction over the bare server and OS; users instead see a pre-built combination of OS and platform. We used this to deploy our project.

Features

This is the front page of our solution. A user can enter the query here and then will be redirected to the results page.



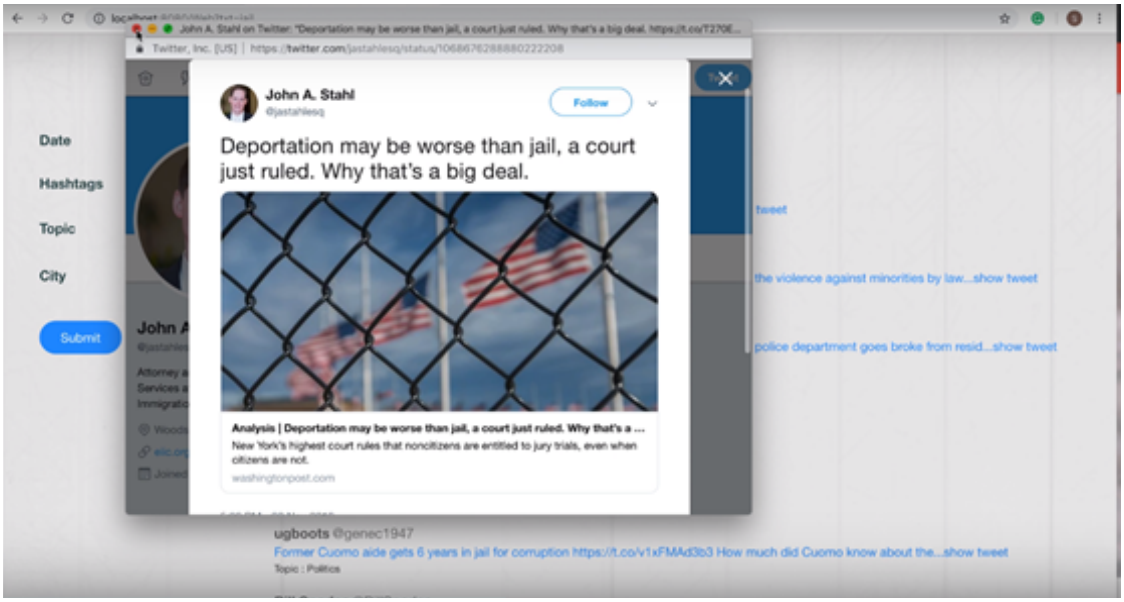
Here is look at our basic result page once the user enters a query:



In the above case, a user has typed the term ‘jail’ and can see the above results. Here are some observations and key features:

- On the left, the user can filter queries. He can select a specific topic, a specific city or can select from all the hashtags that represent a tweet containing the term ‘jail’.

- The top panel has pagination which decongests the result page. Each page displays the top 10 queries.
- Each result has a tweet summary and the topic. There is also a link which redirects the user to the original tweet (As shown in the image below).



- Through the ‘Analyse My Search’ button, a user is redirected to the analysis page where they can visualise various graphs and charts.
    - There is a pie chart which shows the hashtag volume.
    - There is also a world map visualisation to compare tweet volume per the various cities.
    - A histogram helps to compare the topics under which the queried term appeared the most.
    - A scatter chart shows the sentiment analysis, namely ‘Happy’, ‘Neutral’ or ‘Sad’.

Member Contribution

Darshan: I did a requirement analysis and designed an initial UI. I also worked on the Java backend. I used Map and HashMap which helped us to show filters in sorted manner. I hosted the project on AWS elastic beanstalk.

Sai: I worked on analytic and graphical visualization. I helped to design the World map and Pie-chart for Analytic page. I used Google charts for this. I successfully retrieved tweet data for the Google charts and was able to show the respective visualizations.

Siddharth: I worked on the back-end data processing, JSON data handling, integrating it with the User interface and other subtle changes. My work majorly involved working on stack java,

Java Standard Tag Library, Servlets, HTML and significant amount of CSS and Bootstrap. The communication from java backend to all JSPs was governed under the team supervision and the bugs in UI and code were fixed by me in a pure agile manner.

Sravan: I made contributions with tweet extraction and did indexing using Solr by creating cores. I also helped with filter analysis and attempted openNLP by linking it with Solr. I also worked on pushing the filter from JSP to servlet.

Venugopal: I studied the Google chart documentation implemented the Histogram and Scatter-chart. I helped to come up with the parameters for the various graphs. I also worked on demo video and compiled the final report.