University at Buffalo, The State University of New York

CSE 535

Information Retrieval

5BITs - Multilingual Search System

Author:
Miki Padhiary (50286289)
Narendra Reddy Tippireddy (50290387)
Shreyas Rajguru (50289420)
Yash Narendra Saraf (50290453)
Yashankit Shikhar (50289472)

1 Introduction

The goal of this project is to implement a Cross Lingual search engine that allows users to query in any of the five different languages - English, Spanish, French, Thai, and Hindi and retrieve the data in any or all of the languages. The application searches for relevant content in a multilingual database of tweets and presents it to the user with reasonable precision. We have also worked to increase the browsing efficiency with the help of machine translation, result clustering and cross lingual auto complete feature. In the following sections, we briefly describe the data-set used as Index, Pre-processing techniques, back end implementation and finally the front end features available to the end user.

Video URL to the project is: `https://youtu.be/jV-hHOMgpe4`
GitHub URL for the project is: `https://github.com/mikip91/MultilingualSearchSystem`

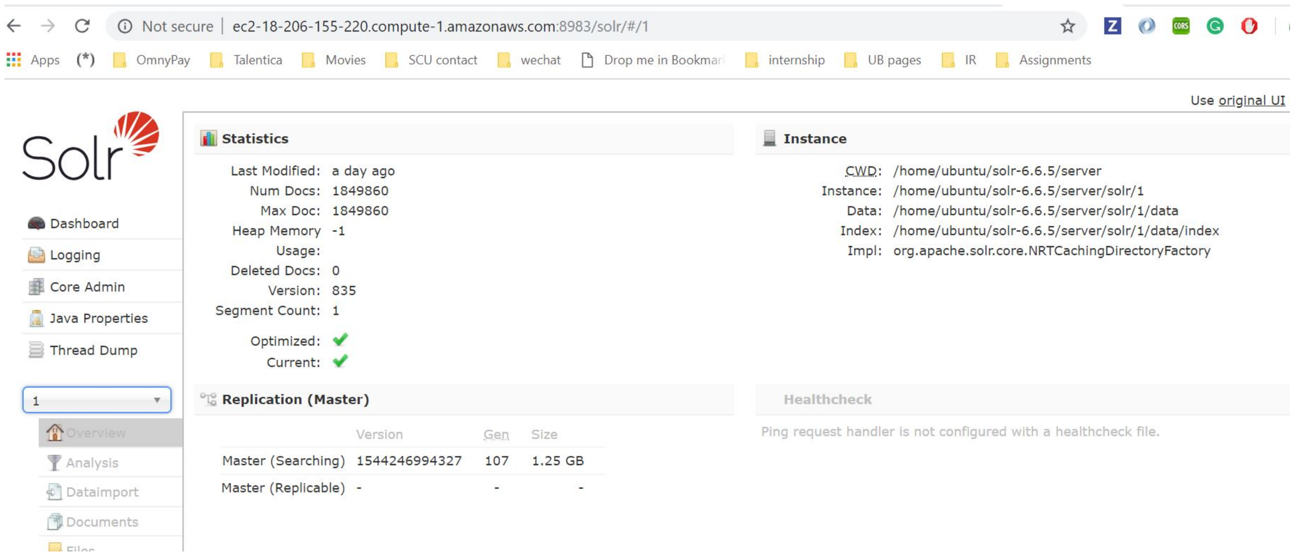
2 Dataset

Data for this project was collected from Twitter using the Twitter API. In addition a GEO coding API was used in order to derive the country of origin for some tweets.

3 Data Preprocessing

3.1 Data Collection

We have been collecting the tweets from Twitter over a span of 75 days. We were able to get around 18 lakh tweets for the given topics. Below is a figure which displays the total number of documents indexed:



Not secure | ec2-18-206-155-220.compute-1.amazonaws.com:8983/solr/#/1

Use original UI

Statistics

```
   Last Modified:  a day ago
        Num Docs:  1849860
         Max Doc:  1849860
     Heap Memory   -1
          Usage:
    Deleted Docs:  0
         Version:  835
   Segment Count:  1
       Optimized:  ✓
         Current:  ✓
```

Instance

```
      CWD:  /home/ubuntu/solr-6.6.5/server
 Instance:  /home/ubuntu/solr-6.6.5/server/solr/1
     Data:  /home/ubuntu/solr-6.6.5/server/solr/1/data
    Index:  /home/ubuntu/solr-6.6.5/server/solr/1/data/index
     Impl:  org.apache.solr.core.NRTCachingDirectoryFactory
```

Replication (Master)

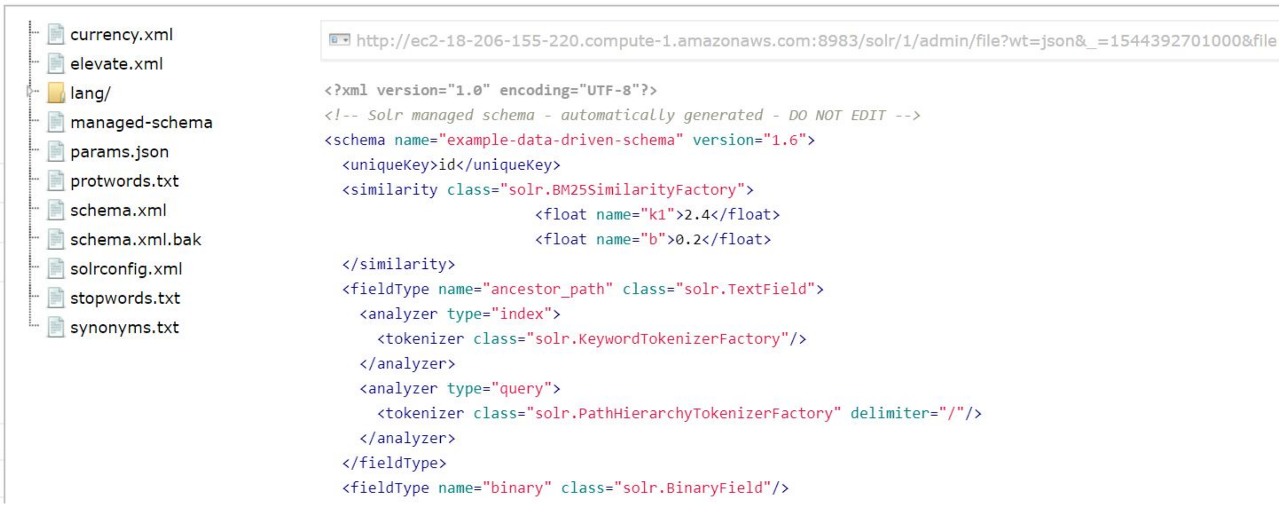
	Version	Gen	Size
Master (Searching)	1544246994327	107	1.25 GB
Master (Replicable)	-	-	-

Healthcheck

Ping request handler is not configured with a healthcheck file.

3.2 Solr functionality

We are using **BM25** Similarity Factory with k1 and b values as 2.4 and 0.2 respectively. The initial corpora had a significant number of duplicate documents. To eliminate the duplicate results, we implemented deduplication strategy once in the preprocessing stage and again in post processing stage. In preprocessing, we filtered the retweeted documents before indexing it to solr. In post processing, we eliminated the rest of the duplicate documents using solr SignatureUpdateProcessorFactory.



http://ec2-18-206-155-220.compute-1.amazonaws.com:8983/solr/1/admin/file?wt=json&_=1544392701000&file

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Solr managed schema - automatically generated - DO NOT EDIT -->
<schema name="example-data-driven-schema" version="1.6">
  <uniqueKey>id</uniqueKey>
  <similarity class="solr.BM25SimilarityFactory">
                      <float name="k1">2.4</float>
                      <float name="b">0.2</float>
  </similarity>
  <fieldType name="ancestor_path" class="solr.TextField">
    <analyzer type="index">
      <tokenizer class="solr.KeywordTokenizerFactory"/>
    </analyzer>
    <analyzer type="query">
      <tokenizer class="solr.PathHierarchyTokenizerFactory" delimiter="/"/>
    </analyzer>
  </fieldType>
  <fieldType name="binary" class="solr.BinaryField"/>
```

4 Back end features

4.1 Query logs and User Relevance Logs

For implementing Query logs we have used mongoDB atlas cloud database where we have two collections.

- The first contains all the logs i.e. the query along with the filters and all the returned tweets ID's. This can be used to check how well the system performed over period of time. Also these documents have a timestamp. So the results can be analyzed based on timestamps.
- The second one is User relevance logs, where if the user clicks on some particular tweet then we draw the conclusion that the user considered the particular tweet important for the respective query so that query along with the particular tweet ID has been stored in the database. As future improvements these relevance logs can be taken into consideration while returning the results for the same set of queries later on.

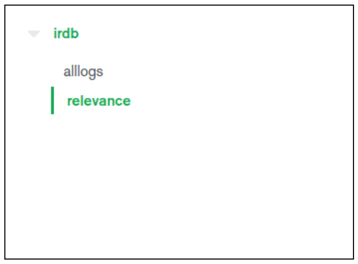
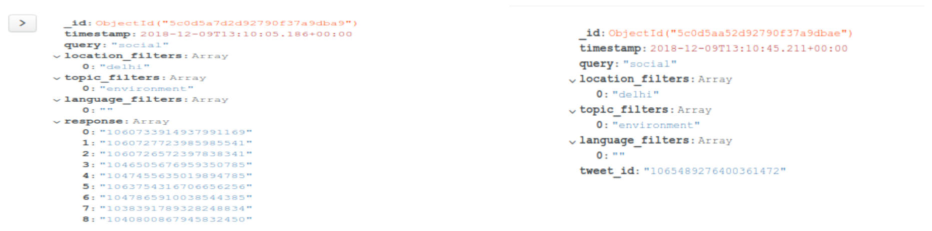


Figure 1: Database Structure

4.2 Pseudo-Relevance Feedback

When we search a specific query and if the results are less than required, Pseudo-Relevance feedback helps in getting more results. Suppose the query returned around 10 results, we take the first tweet's text and use this as the query and get more relevant results. This can be used to improve upon the recall values of the search engine.



```
_id: ObjectId("5c0d5a7d2d92790f37a9dba9")
timestamp: 2018-12-09T13:10:05.186+00:00
query: "social"
location_filters: Array
  0: "delhi"
topic_filters: Array
  0: "environment"
language_filters: Array
  0: ""
response: Array
  0: "1060733914937991169"
  1: "1060727723985985541"
  2: "1060726572397838341"
  3: "1046505676959350785"
  4: "1047455635019894785"
  5: "1063754316706656256"
  6: "1047865910038544385"
  7: "1038391789328248834"
  8: "1040800867945832450"
```

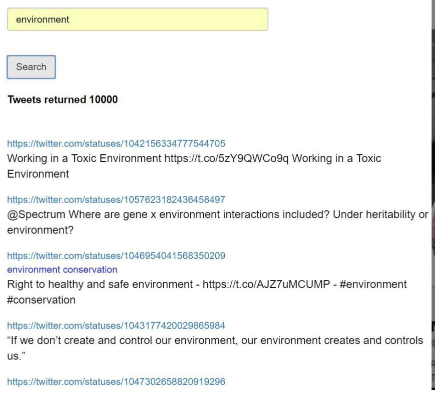
```
_id: ObjectId("5c0d5aa52d92790f37a9dbae")
timestamp: 2018-12-09T13:10:45.211+00:00
query: "social"
location_filters: Array
  0: "delhi"
topic_filters: Array
  0: "environment"
language_filters: Array
  0: ""
tweet_id: "1065489276400361472"
```

Figure 2: *a*.All logs, data structure in the database *b*. Relevance logs, data structure in the database

5 Front end features

5.1 Redirection to Twitter Handle

The user was redirected to the particular tweet on the twitter page so that the user can follow the particular user or check out what other tweets the twitter user has posted. This was also stored in the database as a relevance feedback. Below is a figure which illustrates this



environment

Tweets returned 10000

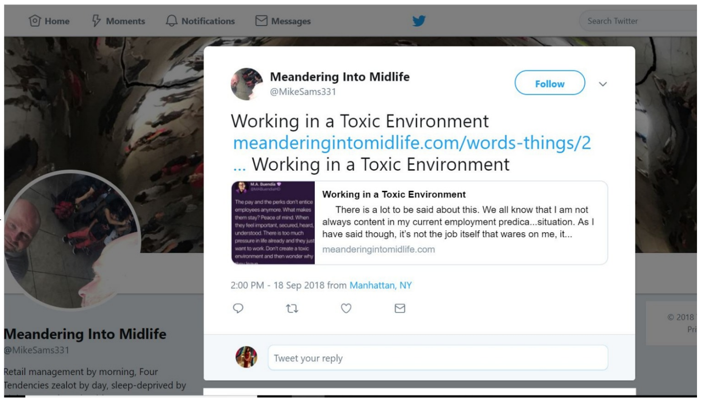
https://twitter.com/statuses/1042156334777544705
Working in a Toxic Environment https://t.co/5zY9QWCo9q Working in a Toxic Environment

https://twitter.com/statuses/1057623182436458497
@Spectrum Where are gene x environment interactions included? Under heritability or environment?

https://twitter.com/statuses/1046954041568350209
environment conservation
Right to healthy and safe environment - https://t.co/AJZ7uMCUMP - #environment #conservation

https://twitter.com/statuses/1043177420029865984
"If we don't create and control our environment, our environment creates and controls us."

https://twitter.com/statuses/1047302658820919296



Meandering Into Midlife
@MikeSams331

Working in a Toxic Environment meanderingintomidlife.com/words-things/2 ... Working in a Toxic Environment

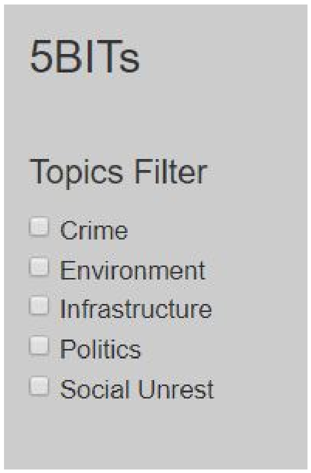
Working in a Toxic Environment
There is a lot to be said about this. We all know that I am not always content in my current employment predica...situation. As I have said though, it's not the job itself that wares on me, it...
meanderingintomidlife.com

2:00 PM - 18 Sep 2018 from Manhattan, NY

Meandering Into Midlife
@MikeSams331
Retail management by morning, Four Tendencies zealot by day, sleep-deprived by

5.2 Faceted query

Solr's faceting feature automatically determines the unique terms for a field and returns a count for each of those terms. We have used **facet.query**, and overridden the default solr behavior and select exactly which terms or expressions a user would like to see counted. The selection of checkboxes is a parameter for facteted eries. Below is a figure which illustrates how we have applied faceted search for our search engine

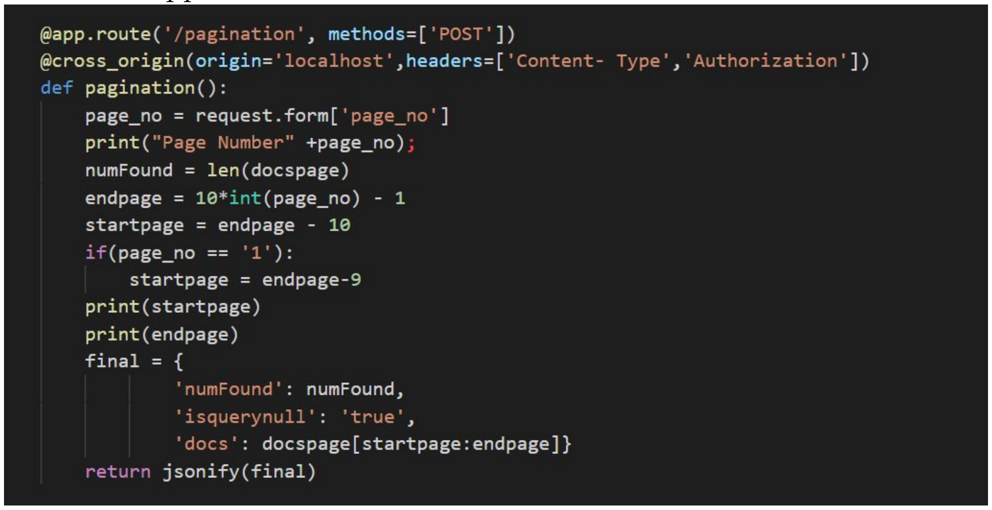

5BITs

Topics Filter

- Crime
- Environment
- Infrastructure
- Politics
- Social Unrest

5.3 Pagination

We have also implemented **Pagination**, which divides the total number of documents into discrete pages. For every query only top 10 results are returned when a query is made. As soon as the user clicks on another page a new ajax call is made and next 10 results are pushed back from the server and the same is displayed in the UI. The implementation is done in python. Below is a snippet for the same:

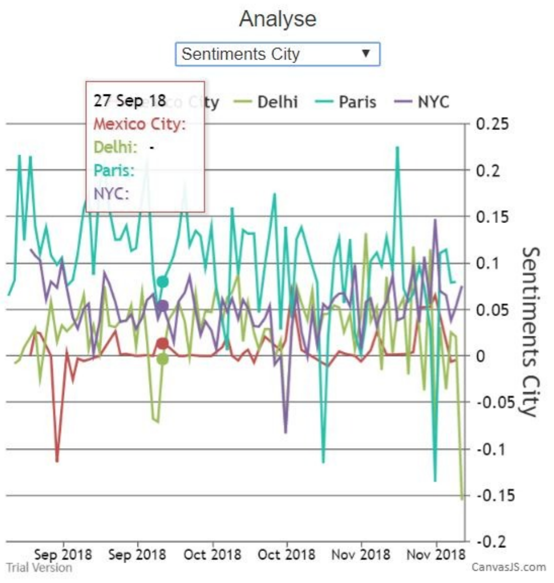


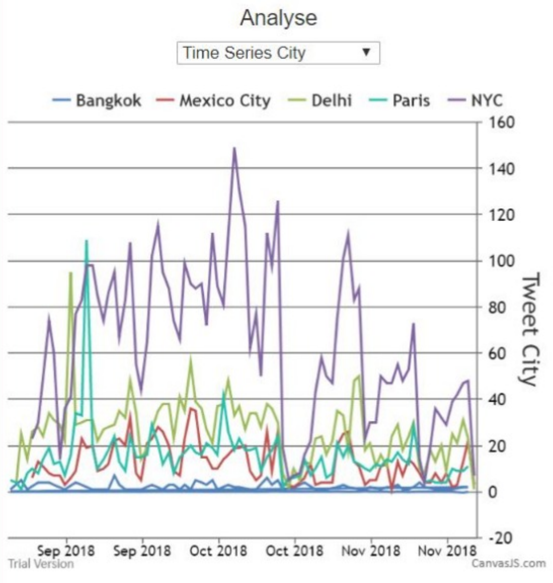
```
@app.route('/pagination', methods=['POST'])
@cross_origin(origin='localhost',headers=['Content- Type','Authorization'])
def pagination():
    page_no = request.form['page_no']
    print("Page Number" +page_no);
    numFound = len(docspage)
    endpage = 10*int(page_no) - 1
    startpage = endpage - 10
    if(page_no == '1'):
        startpage = endpage-9
    print(startpage)
    print(endpage)
    final = {
            'numFound': numFound,
            'isquerynull': 'true',
            'docs': docspage[startpage:endpage]}
    return jsonify(final)
```

6 Analytics

6.1 Time Series Analysis

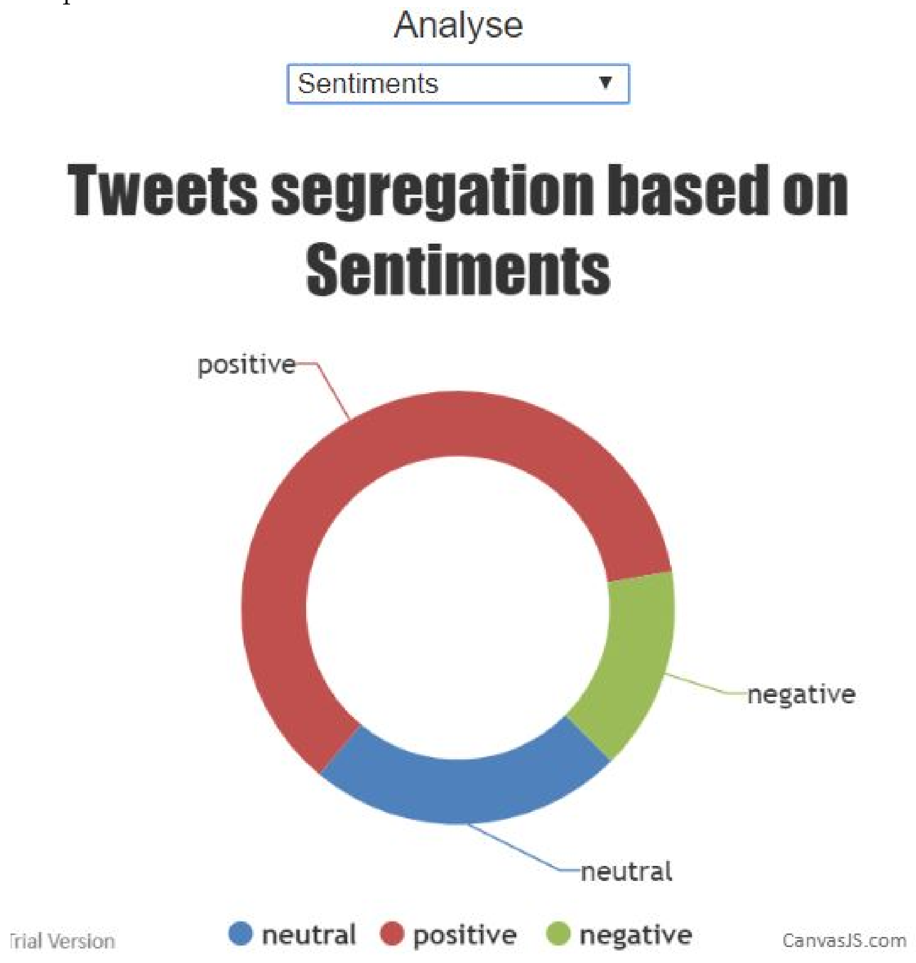
The time series analysis has been performed based on all the filters, i.e. how for a particular topic, language, or city the number of tweets has been varying on a span of 75 days. The analysis has been done on per day basis to get a smooth curve. Now results like where the query was popular during which duration can be extrapolated from this kind of analysis. Below is a figure which illustrates this:





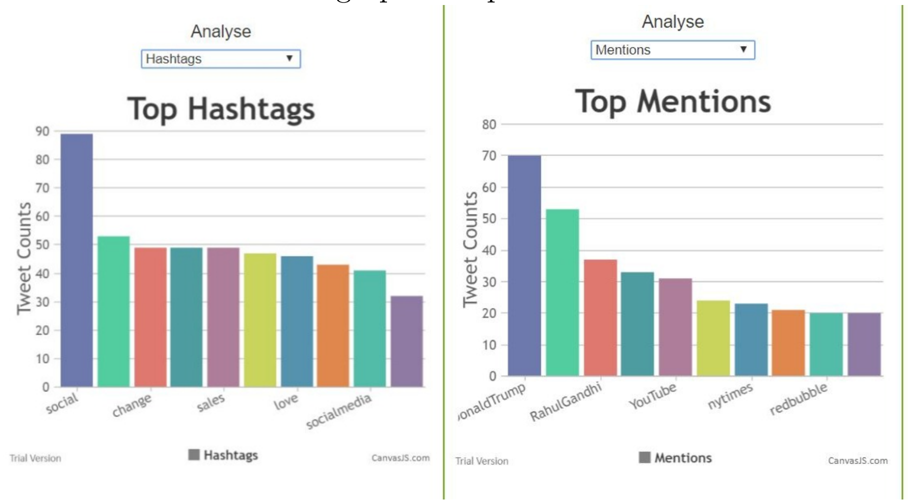
6.2 Sentiment Analysis

The sentiment analysis has been done for two languages as of now i.e. English and french, since most of the tweets are from the above two languages it can be used to extrapolate the results for entire corpora. Also using translation methods the analysis could have been done for all languages but due to time constraints we were not able to do that. Now the popular library textblob has been used from python which gives the result as the floating point number which better analysis the results. The value ranges between -1 and 1. Now the analytics has been done for all the tweets returned on a count based scale. Also time series analysis has been done to see how the sentiment has varied for that particular query with respect to all the filters. The results have been brought to scale to make sure the results can be used to draw some conclusions. Below is a donut chart which displays the sentiment analysis on total corpus:



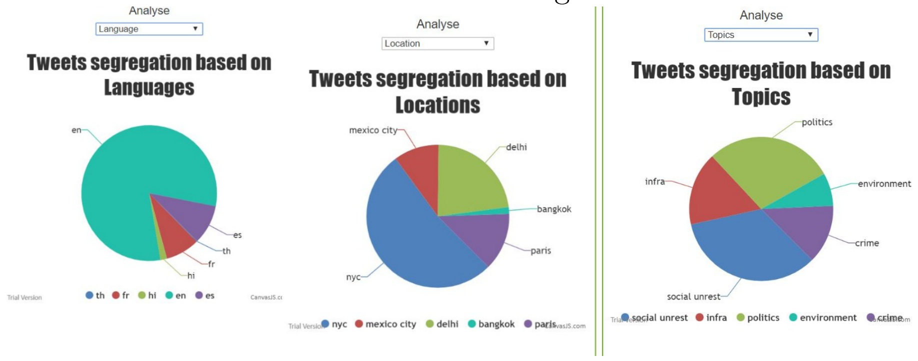
6.3 HashTag and Mentions Analysis

The use of mentions and hashtags helps users find others who have similar interests. So, we have analysed the top **10 Hashtags and Mentions** for the retrieved results and the graph is depicted below:



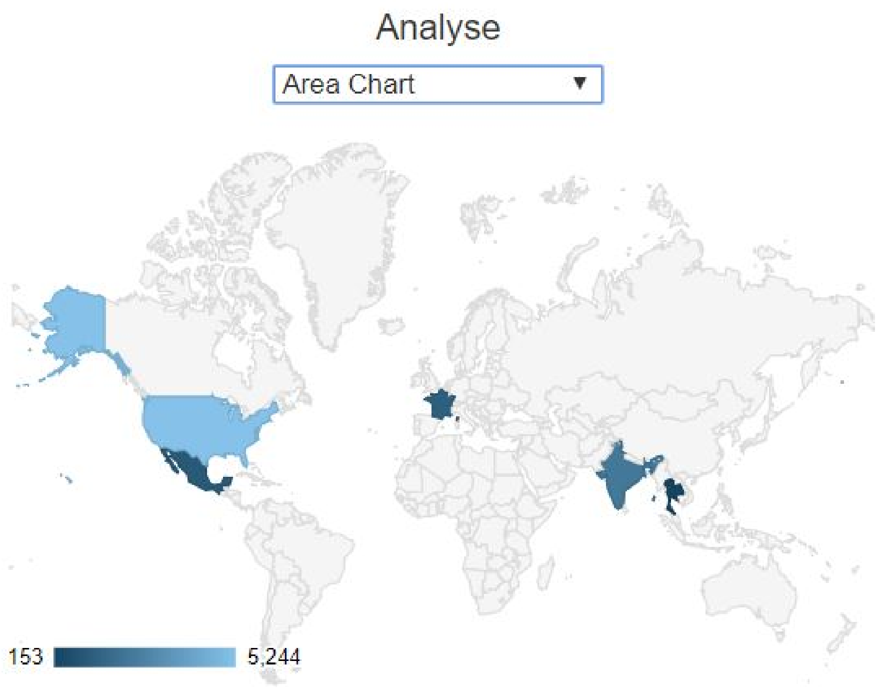
6.4 Count Based Analysis

We analyzed the query results dynamically for each query. For each query we analyze the search results based on **tweets from a specific location, tweets of a specific language and tweets from a specific topic**. We also analyzed the count of top **10 hash tags and mentions** for a particular query. Additionally, we implemented a Area chart which shows the spread of tweets based on location on the world map. This helps the user understand the distribution of tweets better.Below is a figure which illustrates this.



6.5 Region Wise Analysis

We have also analysed the count of tweets coming from each country based on the query and have plotted the same using google highcharts. The graph below depicts the same:



7 Future Improvements

Since the scope of the project is very vast a lot of improvements can be brought to the system like-

- The filtering based on dates, i.e. user can add a range of date to look for a particular query.
- User selecting how many results he wants. This will give the user to decide on freedom if he prefers recall or precision.

- Pseudo relevance can be expanded by giving the user the option to select the tweet on which he wants to expand the original query.
- Text summarization can be considered as a powerful tool for the purpose of analytics.
- Translate API can be used to translate the page into any language possible.
- User can be given an option to query based on the top hashtags or take them to the page of the top user mentions.
- Relevance can be considered as a strong tool for the user as the relevance can be taken into picture when returning the results later on.
- User data can be stored i.e. user based relavance can be implemented.
- Using the peaks and troughs of the graphs the top tweets, most relevant hash-tags on that day, Most re-tweeted tweets can be extracted to know what event caused it.

8 Conclusion

Implementing this project we understood how an end to end search engine works. How the data can be indexed. extracted, and analyzed. Here we dealt with a very specific type of data i.e. tweets which is very uniform i.e. consists of only 160 characters still the so much of analytics can be performed on such a restricted domain for extracting relevant data. We also got an opportunity to implement some concepts like query logs, faceted searches, pseudo-relevance which was studied theoretically as part of the course. Also how to make sure that the Web UI looks appealing to the user was understood. Different analytics patterns were studied like time-series, count based, sentiment based, etc. All in all an end to end system was designed and deployed.

References

[1] `http://archive.apache.org/dist/lucene/solr/ref-guide/apache-solr-ref-guide-6.2.pdf`

[2] http://lucene.apache.org/core/4_0_0/core/org/apache/lucene/search/similarities/TFIDFSi milarity.html?is-external=true

[3] http://trec.nist.gov/trec_eval/

[4] https://cwiki.apache.org/confluence/display/solr/The+Standard+Query+Parser