CSE 535 Information Retrieval

Fall 2016

University at Buffalo

Cross Lingual Information Retrieval

Team Extreme! – Report

Chaitanya Vedurupaka

Venkata Saicharan Kolla

Anirudh Yellapragada

Saani Ausaf

1.Overview

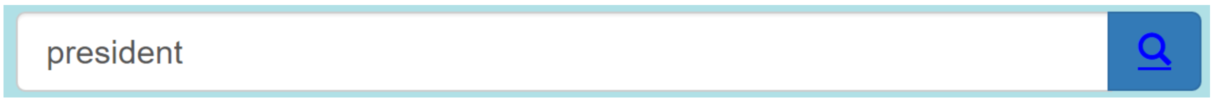
Cross lingual Information retrieval is a subfield of Information retrieval dealing with retrieval of information in a language different from the language of the input query. For example, when the input query given by the user is in Spanish language, the relevant documents in English, Italian and other languages should also be retrieved. To achieve this, our CLIR system mainly focused on translation and expansion of the input query. The number of languages chosen for the document collection counts to 6. The input query has been translated to remaining 5 different languages and then the query is expanded by including the translated terms. Now the new expanded query is searched for in the indexed multi lingual indexed data collection and the relevant documents are retrieved. The IR model used for retrieval is Divergence from Randomness which is a type of probabilistic model. In this model, the weights of the term are computed by measuring the divergence between a term distribution produced by a random process and the actual term distribution. 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 language of the user's query. For example, if the language of the input query is English, out of the retrieved relevant documents, boosting factor has been given to the documents that are originally in English. 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 the number of documents retrieved in different languages, the number of hash-tags available in the documents retrieved, the locations mentioned in the retrieved documents in the form of pie chart and bar chart. The UI is designed also to provide the user with a custom search option where the results can be filtered based on language, date, hash-tags etc. As the data collection chosen for the project is tweets, the date, the user handle, the profile picture of the user, the original tweet, the url's, the date of the tweet, the language of the tweet, the time of the tweet have been retrieved from the indexed data.

2.Technologies Used

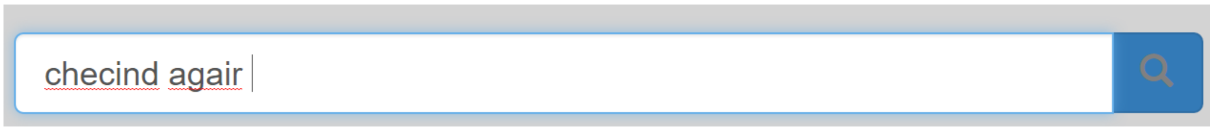
UI : JavaScript, jQuery, ajax, bootstrap , html , css
Web Services : Microsoft Azure API's (for query translation),Alchemy(sentimental analysis), Thesaurus(for synonyms list in all the languages), JavaScript spellcheck(for spell check on input search box), Microsoft bing translator widget(for translation of text on UI).
Backend: Solr

3.User Interface

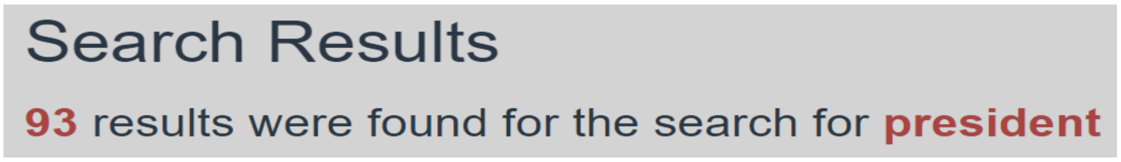
We have a single line text box with the function of accepting user inputs to be searched for in the data which has been indexed. As an enhancement, the search box is also accompanied with spell check feature. As an input query is given by the user, the system will create a http request to the Solr instance where the data has been indexed. The http response obtained from Solr contains JSON objects, which are parsed and processed to display the relevant content to the user.



A sample search input with spell check is shown below:



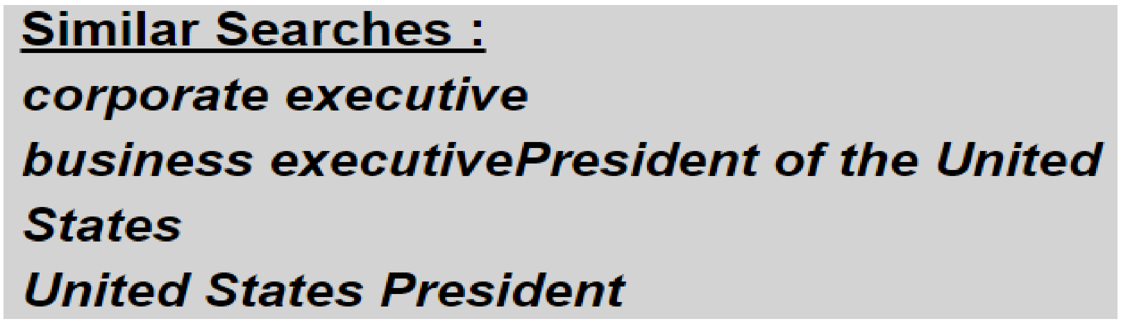
The UI has been designed to display the number of results retrieved as below.
The query is highlighted in the text which displays the number of results.



Search Results

93 results were found for the search for **president**

A ‘Similar search’ is a way of providing suggestions to the user. We have designed the UI to provide related words to the input query given by the user. This may be a trivial addition but not an invalid enhancement to improve the user experience. This is achieved using Thesaurus web service. A sample snap shot is shown below:



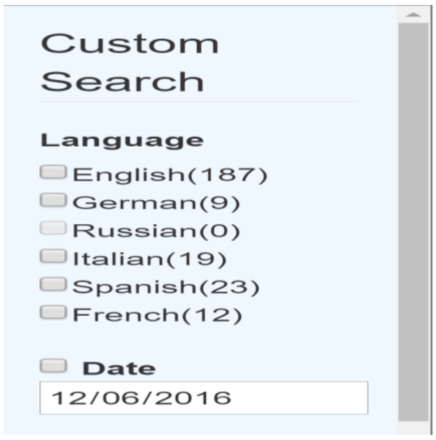
Similar Searches :

corporate executive

business executivePresident of the United States

United States President

3.1) Custom Search: User is provided with multiple options to filter the collection of information. This feature has been implemented by a technique called Faceted Search. This technique is used for accessing information organized according to a systematic order. The user is provided with different filters such as language and date. The checkboxes are available only if the relevant results contained the language, if the user wants to filter by language. For example, in the below figure, the Russian language check box has been disabled as there are no documents relevant to the input query which are in Russian language. The user can also choose the date to retrieve relevant documents pertaining to that specific date, meaning, the tweets tweeted on the chosen date. The hash-tags which appeared in the retrieved relevant tweets, are filtered and top 10 are displayed on the UI. The user has an option to choose the documents/tweets containing the selected hash-tags.



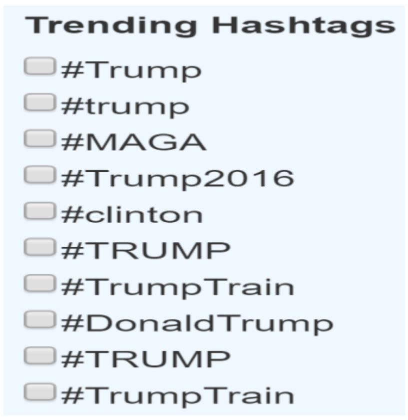
Custom Search

Language

- English(187)
- German(9)
- Russian(0)
- Italian(19)
- Spanish(23)
- French(12)

Date

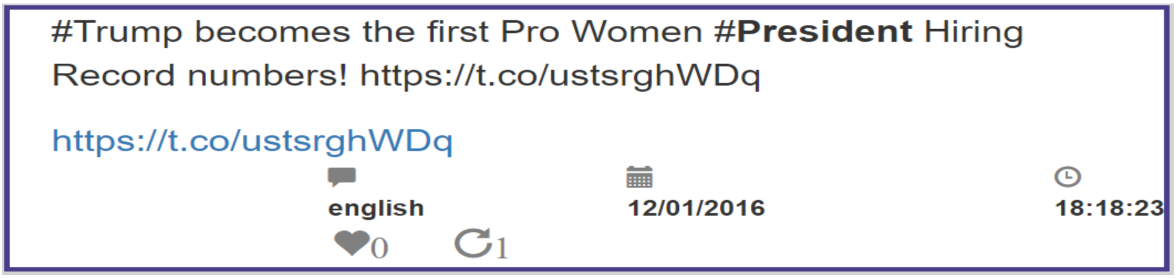
12/06/2016



Trending Hashtags

- #Trump
- #trump
- #MAGA
- #Trump2016
- #clinton
- #TRUMP
- #TrumpTrain
- #DonaldTrump
- #TRUMP
- #TrumpTrain

3.2) Sample Result: Below is the sample tweet being displayed. The various components of the tweet are the text, the URL mentioned in the text, the language of the tweet, the date and time of the tweet, favourite count and retweet count. Apart from these, the screen name, and the profile image of the user has been displayed. You can also see that the query term i.e "**President**" is being emphasized in the result text below

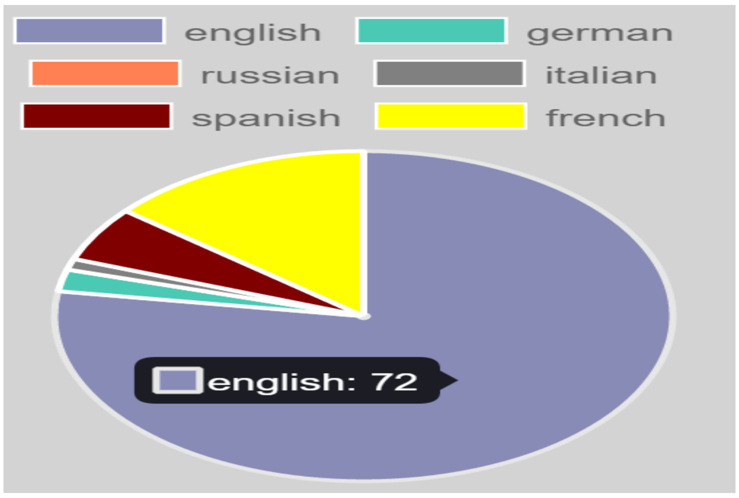

#Trump becomes the first Pro Women #**President** Hiring Record numbers! https://t.co/ustsrghWDq

https://t.co/ustsrghWDq

english 12/01/2016 18:18:23

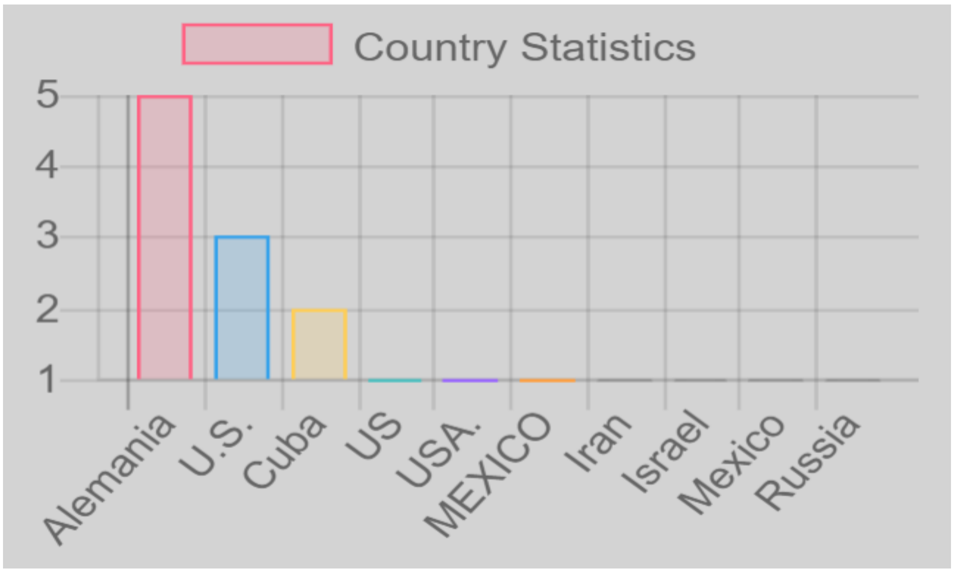
0 1

3.3) Charts: Graph is an easy form of representation of information. UI has been designed to display a pie chart which show cases the information on number of a tweets of a particular language with respect to all other languages of the results obtained. This has been implemented using the available open source chart.js java script.



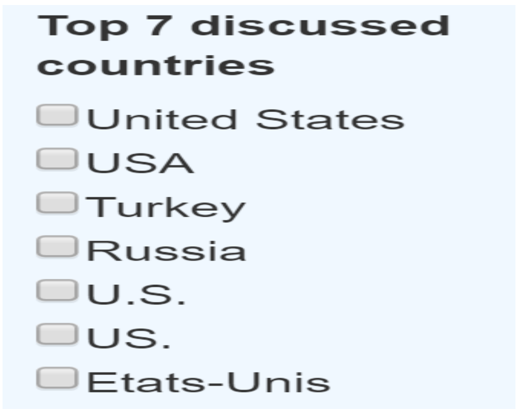
In addition to pie chart discussed, a bar chart which represents the frequency of the names of the countries specified in the tweets has been added to the UI.

The below statistics has been gathered using a Alchemy API.

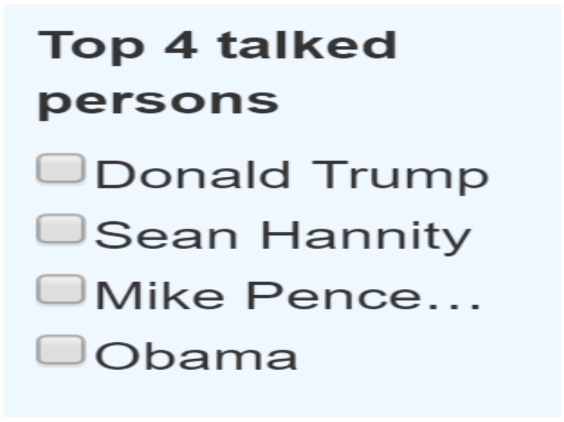


3.4) Other Statistics: The JSON file containing the tweets collected, has been processed to further add fields like Person and Country. This has been achieved using the Alchemy API. Alchemy API uses a natural language processing technology and machine learning algorithms to extract semantic meta-data from the content, such as information on people, places, companies, topics, facts, authors and so on. We have made best use of this API to add filed like Country and Person to the JSON file. These fields are also indexed along with many other existing and relevant fields. The below figures show a sample semantic information from the text of the tweets retrieved using the Alchemy API. The content displays the top results of the countries spoken of, in the tweets that are found to be relevant based on the input query. Also, the top results of the persons whose names are involved in the tweets have been successfully retrieved. This technique of analysing and extracting relationships between the tweets and the terms using sophisticated natural language processing techniques to get a high-level understanding of the content is semantic analysis.

The below fields have been added to the Custom Search option on the UI, thereby providing user, the convenience to browse results based on the Countries or Persons spoken of, in the relevant tweets.


Top 7 discussed countries

- ☐ United States
- ☐ USA
- ☐ Turkey
- ☐ Russia
- ☐ U.S.
- ☐ US.
- ☐ Etats-Unis


Top 4 talked persons

- ☐ Donald Trump
- ☐ Sean Hannity
- ☐ Mike Pence...
- ☐ Obama

3.5) Number of results per page : The number of results/tweets displayed per page is limited to 5. A maximum of 50 pages are supported, and to navigate to other pages, the user has been provided with page numbers of the result pages, which when clicked display additional results.

4.Data Collection

Type of data	:	Tweets.
Topic	:	Trump
# of languages	:	6
Total # of tweets	:	30,000(Test Collection).
Language Spread	:	English, Russian, Spanish, German, French, Italian.

4.1.) Data Retrieval Using Python Crawler: The tweet data is collected using the twitter REST API. We have downloaded the tweet data into a json file over a period of time and in different languages. The crawler is designed in python language using libraries and twitter access tokens.

4.2) Pre - Processing: The data collected has many fields relating to a tweet. We designed a python code to retrieve the relevant fields and copy them to newly declared fields and write the data to a json file. The date field in collected raw data is not in the format expected by the Solr(Example:

tweet_date format is not as expected by Solr) and we make changes using python libraries to format the date as can be recognized by solr. Indexing a huge amount of unnecessary data might cause a search engine to slow down or show some delay in displaying results and hence, the raw data has been processed in the above manner.

4.3)List of fields extracted and indexed:

tweet_lang, tweet_text, tweet_text_en, tweet_text_ru , tweet_text_es , tweet_text_fr, tweet_text_de, tweet_text_it, tweet_favourite_count, tweet_id, tweet_date, tweet_retweet_count, tweet_username, tweet_hashtags, tweet_url, screen_name, profile_image_url, Country , Person.

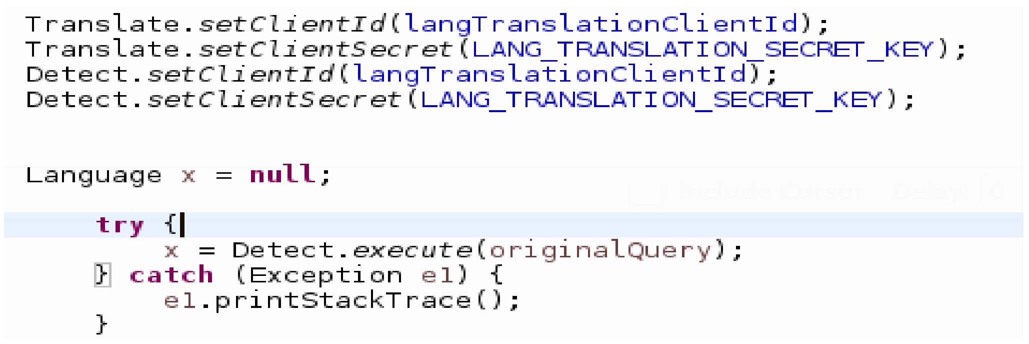
5.Input Query

5.1) Query Expansion on UI side: After the input query is given on the UI search box, it is expanded using a thesaurus list (which acts as the Synonyms.txt) obtained using Thesaurus, a web service which provides synonyms in different languages. This expanded query is given as input to solr.

5.2) Query Parser: We have designed our own query parser that takes the input user query and converts into the required languages, thereby resulting in query expansion. This expanded query is returned in a new Extended dismax query parser to the solr, and so the tweets are returend in all the indexed languages.

5.3) Query Translation:

5.3.1) Detecting language of query: The input query is detected using the Microsoft Azure api (Microsoft translator) with the generated access tokens. The dependency .jar file has the Detect class which is used in query language detection.



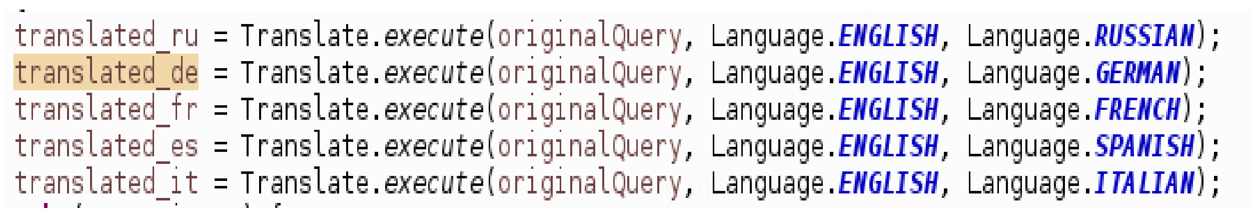
```
Translate.setClientId(langTranslationClientId);
Translate.setClientSecret(LANG_TRANSLATION_SECRET_KEY);
Detect.setClientId(langTranslationClientId);
Detect.setClientSecret(LANG_TRANSLATION_SECRET_KEY);


Language x = null;

    try {
        x = Detect.execute(originalQuery);
    } catch (Exception e1) {
        e1.printStackTrace();
    }
```

5.3.2) Query Translation using Microsoft Azure:

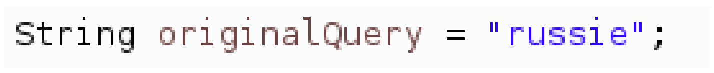
Based on the detected language, the original query is translated into other languages using the Language and Translate classes. Based on the translated queries, the final query to the parser is expanded such that the translated queiries are given to the respective fields. For a given english query translation is done as stated below:



```
translated_ru = Translate.execute(originalQuery, Language.ENGLISH, Language.RUSSIAN);
translated_de = Translate.execute(originalQuery, Language.ENGLISH, Language.GERMAN);
translated_fr = Translate.execute(originalQuery, Language.ENGLISH, Language.FRENCH);
translated_es = Translate.execute(originalQuery, Language.ENGLISH, Language.SPANISH);
translated_it = Translate.execute(originalQuery, Language.ENGLISH, Language.ITALIAN);
```

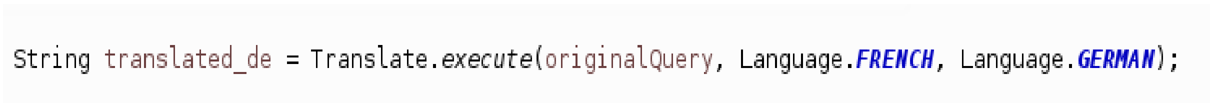
5.4) Sample input/output:

Input



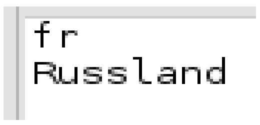
```
String originalQuery = "russie";
```

Commands: Language detection as stated above. For translation into German:



```
String translated_de = Translate.execute(originalQuery, Language.FRENCH, Language.GERMAN);
```

Output: Language (here, French) and translated word into other language (here, German). *Output*



```
fr
Russland
```

5.5) Indexing Model:

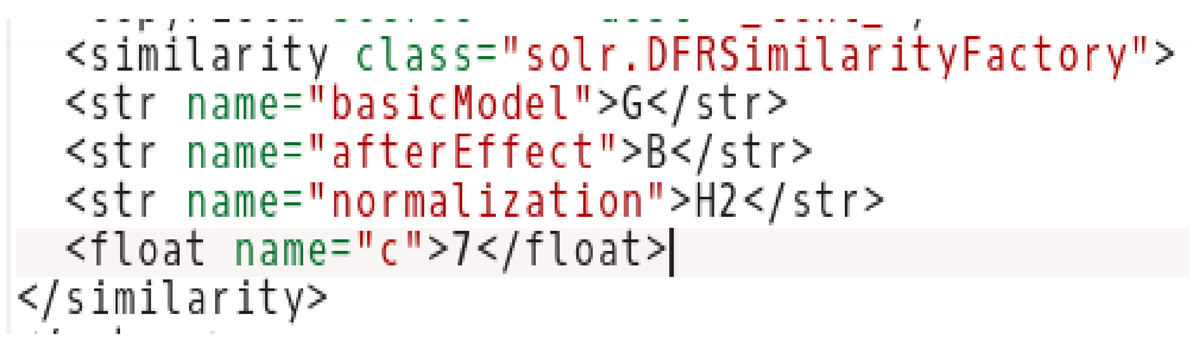
The indexing is based on the relevance of output results. The following stated methods were implemented to improve the relevance of the results and thus, the indexing is done based on thses models.

5.5.1) Relevance:

Relevance has been improvised based on 2 things, first the similarity IR model and second the language of the input query.

a) Relevance based on similarity IR model:

We tried indexing with different techniques similarity model techniques (VSM , BM25 and DFR) with different parameters and observed the top-20 results for the same query among the three models. We determined the output relevancy (for the first top 20 tweets) by judging each of the output tweet based on the query and determined the precision for each model. From the above observations, we found that for our corpus, the DFR model with the below parameters is giving more relevant output i.e. more precision and hence chosen that similarity model.



```
  <similarity class="solr.DFRSimilarityFactory">
  <str name="basicModel">G</str>
  <str name="afterEffect">B</str>
  <str name="normalization">H2</str>
  <float name="c">7</float>
</similarity>
```

b) Relevance based on query language:

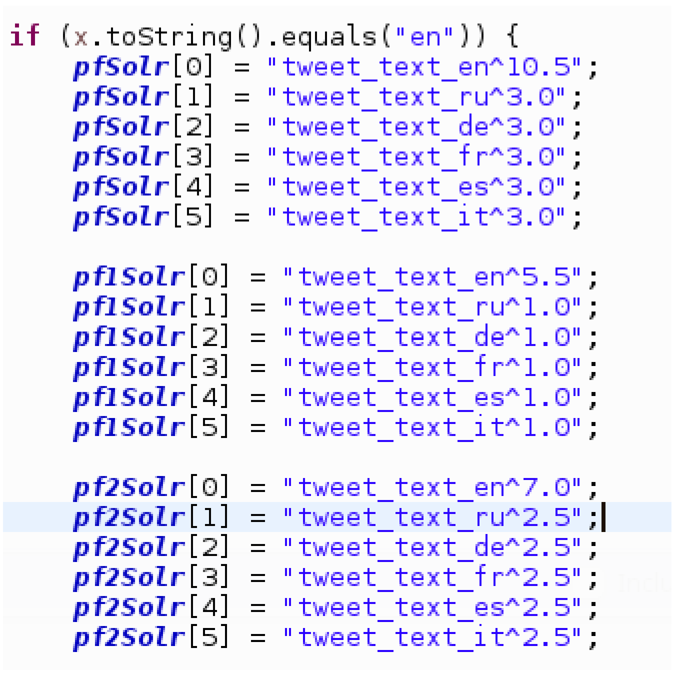
Assumption: It is highly likely that the language of the input query is the language spoken by the user. With this assumption in mind, if the language of the input query is Spanish, it is highly likely that the user is interested in tweets that are tweeted in Spanish and hence these tweets are more relevant to the user. So out of the results obtained, Spanish tweets are given a bit higher boost factor than the other language tweets. But since the motto of the project is to have output spread across different languages we have given boost factor to other languages as well (after query translation) but with a lesser boost factor than the query language. If a query and results of the query displayed are of different languages, it is not a user-friendly search engine if there is no means to translate the result to the language chosen by the user. Therefore, for the convenience of the user, a translate widget has been added on the UI. The widget is implemented using Microsoft

Bing translator. When the user launches this widget he/she has the option to translate all the text in the result page up to 50 languages.

c) Boosting:

Boosting can be done in our designed query parser as it extends the eDismax Query parser properties. The required fields are selected and the boosting to these fields are given on various conditions.

d) Text fields: Based on the language of the original query given by the user, the boosting is given high to the text field of that language. And for the remaining languages, their corresponding text fields are given equal boosting. When the language of the original query is other than those that are indexed, the boosting is equally given to all text fields.



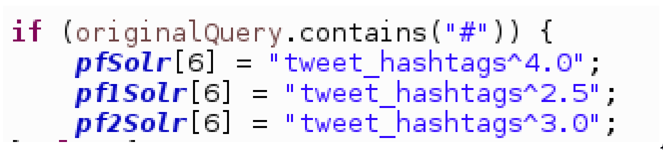
```
if (x.toString().equals("en")) {
    pfSolr[0] = "tweet_text_en^10.5";
    pfSolr[1] = "tweet_text_ru^3.0";
    pfSolr[2] = "tweet_text_de^3.0";
    pfSolr[3] = "tweet_text_fr^3.0";
    pfSolr[4] = "tweet_text_es^3.0";
    pfSolr[5] = "tweet_text_it^3.0";

    pf1Solr[0] = "tweet_text_en^5.5";
    pf1Solr[1] = "tweet_text_ru^1.0";
    pf1Solr[2] = "tweet_text_de^1.0";
    pf1Solr[3] = "tweet_text_fr^1.0";
    pf1Solr[4] = "tweet_text_es^1.0";
    pf1Solr[5] = "tweet_text_it^1.0";

    pf2Solr[0] = "tweet_text_en^7.0";
    pf2Solr[1] = "tweet_text_ru^2.5";
    pf2Solr[2] = "tweet_text_de^2.5";
    pf2Solr[3] = "tweet_text_fr^2.5";
    pf2Solr[4] = "tweet_text_es^2.5";
    pf2Solr[5] = "tweet_text_it^2.5";
```

When the given query is in English, then the boostings are given as shown in the figure.

e) Hashtags: When the original query given contains hashtag (for example, #trump), then the boosting for the field tweet_hashtags is increased in order to retrieve more tweets that contain hashtags.



```
if (originalQuery.contains("#")) {
    pfSolr[6] = "tweet_hashtags^4.0";
    pf1Solr[6] = "tweet_hashtags^2.5";
    pf2Solr[6] = "tweet_hashtags^3.0";
```

f) Other fields:

For all the input queries, the tweet_followers_count and tweet_retweet_count are given boosting in order to retrieve more precise tweets based on these fields. This boosting adds to the scoring of tweets and increases the precision of returning results.

g) Attempted Techniques:

Clustering Analysis: Clustering is the task of grouping set of document in a collection into one group such that the documents in the same group are more similar to each other than to the documents in the other group. The goal of this technique is to let the user explore and narrow the focus to a group of similar documents. This has been implemented in Solr using an algorithm made available by the Carrot2 open source project.

Reason to attempt: As this is Cross Lingual IR system, we tried clustering tweets irrespective of the language of the tweet. This was done to enrich the user experience by displaying the clusters and the language spread inside a cluster.

<u>Implementation Details:</u>

1. Clustering Component

```
<searchComponent name="clustering"
                 enable="${solr.clustering.enabled:false}"
                 class="solr.clustering.ClusteringComponent" >
```

2. Declaration of Clustering algorithm.

```
<lst name="engine">
  <str name="name">lingo3g</str>
  <bool name="optional">true</bool>
  <str name="carrot.algorithm">com.carrotsearch.lingo3g.Lingo3GClusteringAlgorithm</str>
  <str name="carrot.resourcesDir">clustering/carrot2</str>
</lst>

<lst name="engine">
  <str name="name">lingo</str>
  <str name="carrot.algorithm">org.carrot2.clustering.lingo.LingoClusteringAlgorithm</str>
  <str name="carrot.resourcesDir">clustering/carrot2</str>
</lst>

<lst name="engine">
  <str name="name">stc</str>
  <str name="carrot.algorithm">org.carrot2.clustering.stc.STCClusteringAlgorithm</str>
  <str name="carrot.resourcesDir">clustering/carrot2</str>
</lst>

<lst name="engine">
  <str name="name">kmeans</str>
  <str name="carrot.algorithm">org.carrot2.clustering.kmeans.BisectingKMeansClusteringAlgorithm</str>
  <str name="carrot.resourcesDir">clustering/carrot2</str>
</lst>
```

3. A request handler to demonstrate clustering component.

```
<requestHandler name="/clustering"
                startup="lazy"
                enable="${solr.clustering.enabled:false}"
                class="solr.SearchHandler">
  <lst name="defaults">
    <bool name="clustering">true</bool>
    <bool name="clustering.results">true</bool>
    <str name="carrot.title">name</str>
    <str name="carrot.url">id</str>
    <str name="carrot.snippet">features</str>
    <bool name="carrot.produceSummary">true</bool>
    <int name="carrot.numDescriptions">5</int>
    <bool name="carrot.outputSubClusters">false</bool>

    <str name="defType">edismax</str>
    <str name="qf">
      text^0.5 features^1.0 name^1.2 sku^1.5 id^10.0 manu^1.1 cat^1.4
    </str>
    <str name="q.alt">*:*</str>
    <str name="rows">100</str>
    <str name="fl">*,score</str>
  </lst>
  <arr name="last-components">
    <str>clustering</str>
  </arr>
</requestHandler>
```

With this technique, we observed that the relevant tweets in top 20 results were decreased and hence discarded this.

6.Results & Analysis

- It has been observed that the top results in the output is from the query language, which must be due to the reason that we are giving higher boosting factor to query language.
- We have observed that without using query translator the output has mostly the tweets from the query language only, but after applying query translation the returned results had significant outputs from other languages as well.
- Upon thorough analysis for different queries, we found that the most of the top returned tweets are relevant to the query posed.

7.Limitations

- The maximum result pages are limited to 50 as we focussed only on the most relevant tweets pertaining to the query.
- The JavaScriptSpellCheck and Microsoft Bing translator web services may not work sometimes (due to internal server error), so the user cannot launch widget or see spellcheck errors.
- 'Similar searches' field in the result page is retrieved using thesaurus list, so the suggestions displayed in that field may not actually give the relevant search suggestions, but nonetheless it gives relevant suggestions to some extent.
- The date range of the tweets is limited between 22ndNov,2016 and 6th Dec,2016.
- For a noun, the actual query translated is to be noun but sometimes the word can also be a verb etc. So, the tweets vary based on the language and context of the query.
- When the query is not based on "hashtags", we are decreasing the boost parameters to retrieve more relevant tweets than those tweets which have more hashtags.
- The country names and person fields may not be 100% correct as the Alchemy API is not giving accurate outputs every time.

8.Team Contribution

Chaitanya Vedurupaka: Developed most of the UI. Implemented using html, css, jQuery, ajax and bootstrap. Also, implemented thesaurus for query expansion, spell check for input query.

Anirudh Yellapragada: Developed some part of the UI. Implemented widget translator for language translation on UI. Worked on Alchemy API. Hosted the webpage on aws server.

Sai Charan Kolla: Collected around 30000 tweets from different languages using python crawler designed, pre-processed and indexed on solr. Worked on Microsoft Azure translator api's for translation of the query. Design of new Query parser extending the Extended Dismax Query Parser Plugin.

Saani Ausaf: Worked on solr clustering and contributed to document report.

9.Future Work

- The number of languages used can be extended to many languages although only few of the languages work correctly for certain web services like alchemy, thesaurus etc.
- This project has got a lot of scope for enhancement like auto complete, suggestions based on the history.
- Clustering can also be used to give summarizations for different topics in different languages.
- The UI can certainly be improved to include customized dashboards spanning across variables of indexed data for awesome visual graphical representation of the data.

10.References

1. http://www.alchemyapi.com/
2. https://www.microsoft.com/en-us/translator/
3. https://en.wikipedia.org
   http://www.chartjs.org/
4. http://www.javascriptspellcheck.com/
5. http://thesaurus.altervista.org/thesaurus/

6. https://www.microsoft.com/en-us/translator/widget.aspx
7. https://msdn.microsoft.com/en-us/library/dd576287.aspx
8. https://wiki.apache.org/solr/SchemaXml
9. https://cwiki.apache.org/confluence/display/solr/The+Extended+DisMax+Query+Parser
10. https://wiki.apache.org/solr/SolrPlugins