Information Retrieval
Project 4
Cross Lingual IR Model

Sumedh Ambokar	50207865
Saurabh Bajoria	50208005
Sucheta Mulane	50206855
Vidhi Shah	50207090

Table of Contents:

1. Overview

Information on the web is growing in various forms and languages. Though English dominated the web initially, now there are documents in other languages as well. The popularity of internet and availability of networked information sources have led to a strong demand for **Cross Lingual Information Retrieval** (CLIR) systems. It refers to the retrieval of documents that are in a language same or other than the query expressed.

We have implemented a Cross Lingual Information Retrieval system on data crawled from Twitter on a topic- **Trump.** Data crawled include tweets from multiple languages which are indexed using Solr. Users can query in any language of their preference and the search engine returns relevant results in a form that is useful to them, even when they have little or no linguistic competence in the target languages.

The rest of the report is organized as follows: Section 2 describes the Key features and functionalities included in the system, Implementation details are explained in section 3 and section 4 and 5 gives details of project team members and some query/results examples respectively.

2. Key Features

Below are some of the key features and functionalities we have implemented in the project:

1. Multilingual query: The system permits the user to query in any language of his preference and returns relevant results based on the query. The system also auto detects the language entered.
2. Document Language Translation: The system has a feature wherein the user can translate the language of the document (tweet) retrieved from one language to any other language, individually.
3. Relevance Sorting: An algorithm to sort the tweets based on the relevant language selected by the user. If a user selects a language to be most relevant, then the tweets of that language are displayed at the top. This is included under Advanced search option.
4. Language based filtering: Users can filter the retrieved tweets based on a particular language.
5. Date based filtering: Users can filter the retrieved tweets for a date range.
6. Location based filtering: Users can filter the retrieved tweets pertaining to a geo-location.
7. Date based sorting: Users can sort the tweets based on the date.
8. Popularity based sorting: Users can sort the tweet based on the popularity of the tweet, i.e. number of retweets.
9. User/Tweet information: One can navigate to the user’s profile or the tweet page on twitter just by clicking the Username or the tweet respectively.
10. Language distribution graph: A language distribution graph is plotted for the number of tweets retrieved.
11. Pagination: We have implemented pagination to display only 10 tweets per page. A user needs to navigate to the *next* pages for seeking more tweets.

3. Implementation Details

3.1 Tweet Crawling

We crawled a total of 10,000 tweets in 5 different languages as below on the topic provided, i.e. **Trump**

i. English
ii. Turkish
iii. Russian
iv. German
v. Hindi

We selected several fields and created a JSON file for indexing using SOLR.
Below are all the fields with their usage and type:

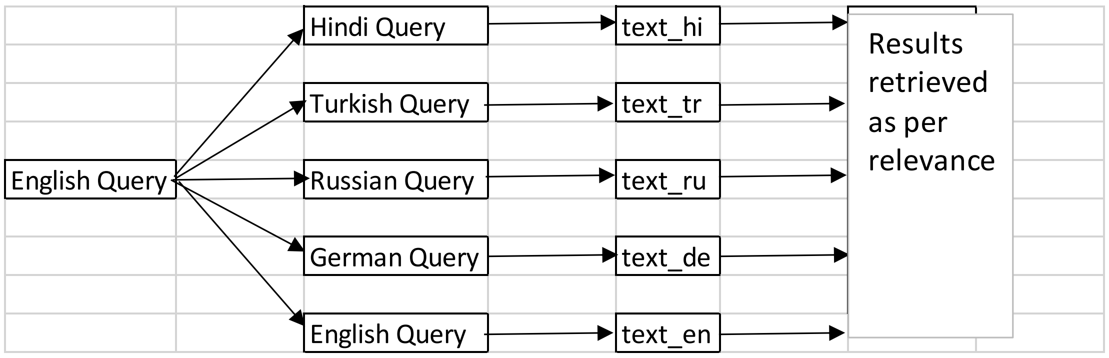
Field Name	Usage	Field Type
hashtags	Hashtags of the tweet	String,Multivalued
id	Tweet ID	String, Unique
mentions	Mentions of the tweet	String,Multivalued
retweet_count	Retweet count	String
text_de	Tweet text in German	text_de
text_en	Tweet text in English	text_en
text_hi	Tweet text in Hindi	text_hi
text_ru	Tweet text in Russian	text_ru
text_tr	Tweet text in Turkish	text_tr
topic	Topic of the tweet	String
tweet_date	TimeStamp of the tweet	String
tweet_emoticons	Emoticons of the tweet	String,Multivalued
tweet_image	URL of the tweet Image	String
tweet_lang	Language of the tweet	String
tweet_loc	Location of the tweet/user	String
tweet_text	Raw Tweet text	text_general
tweet_url	URL of the tweet	String
tweet_urls	URLs in the tweet	String
user_loc	Location of the user	String
User_Name	Name of the User	String
user_pic	URL of the user profile picture	String
User_ScreenName	ScreenName of the user	String
User_url	URL for the profile of the user	String

3.2 IR Model Implementation

We have implemented the **Vector Space Information retrieval model**.
To run in Schema mode, we have used schema.xml to define various fields, field types and the Classic Similarity Model (Vector space model).
Below are the techniques used to improve the relevance of the tweets to be retrieved based on the query:

3.2.1 Query Language Translation and Field specific querying

We wanted to retrieve documents from all the languages irrespective of the language of the query. For example, a query in Hindi would fetch results from all the languages.
For this we used Yandex Translator, which detects the language of the query automatically and translates it to the remaining 4 languages. Documents are retrieved based on these translated queries and displayed in order of relevance.
Please refer to the below diagram for more details:



Below query is an example of implementing this method:

```
q= text_en:(English Query) OR text_hi:( अंग्रेजी केरी) OR text_ru:( английский запрос) OR
text_de:( Englisch Abfrage) OR text_tr:( Ingilizce sorgu)
```

3.2.2 Boosting Text field using Language selected for highest relevance

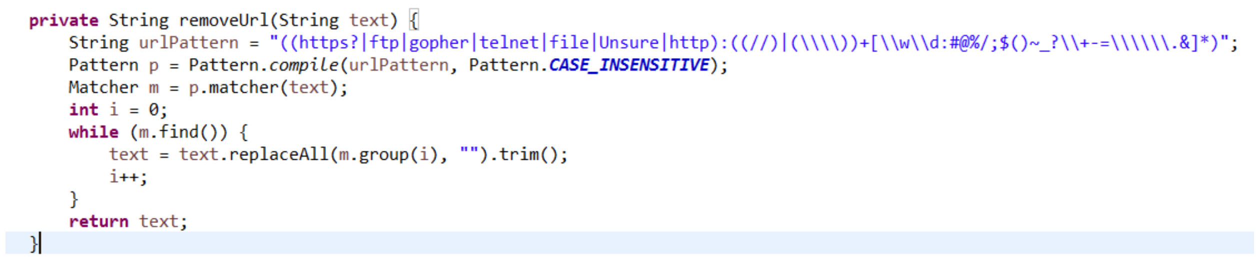
If a user opts for relevance sorting based on a particular language, this method can be used to boost the score of search in the text field of that language.
For example, if a user wants to get results with Hindi as the most relevant language, then the weight for *text_hi* is made higher than the other text fields.
In this case, the above query is modified as shown:

```
q= text_en:(English Query) OR text_hi:( अंग्रेजी केरी)^10 OR text_ru:( английский запрос) OR
text_de:( Englisch Abfrage) OR text_tr:( Ingilizce sorgu)
```

As you can see, the weight for text_hi is boosted by a factor of 10.

3.2.3 URL/Garbage (unwanted data) Filtering

The tweet text fields contain a lot of unwanted information like the URLs, special character which reduces the precision and thus relevant tweets are assigned lower scores.
We filtered out the URL from the text fields while crawling the tweet from twitter.
Please find below the snippet of the code for the same:



```
private String removeUrl(String text) {
    String urlPattern = "((https?|ftp|gopher|telnet|file|Unsure|http):((//)|(\\\\))+[\\w\\d:#@%/;$()~_?\\+-=\\\\\\.&]*)";
    Pattern p = Pattern.compile(urlPattern, Pattern.CASE_INSENSITIVE);
    Matcher m = p.matcher(text);
    int i = 0;
    while (m.find()) {
        text = text.replaceAll(m.group(i), "").trim();
        i++;
    }
    return text;
}
```

3.2.4 Phrase Boosting with Phrase Slop

Phrase Boosting

Using Phrase boosting, higher relevance is given to the documents that contain exact phrase as that in the query.
For instance, using the above example, higher relevance is given to a document that contains “English” and “query” together as one phrase “English Query”.
To include this, the above query is now modified as below:

q= text_en:(English Query) OR text_hi:(अंग्रेजी क्वेरी) OR text_ru:(английский запрос) OR text_de:(Englisch Abfrage) OR text_tr:(Ingilizce sorgu) OR text_en:**(“English Query”)^2**

Phrase Slop

Using phrase slop, one can specify the maximum distance of words from each other to be considered a phrase. Considering the same example as above, if Phrase slop=3, then there can be a maximum of 3 words between “English” and “Query” to still consider it as a phrase.
Slop is specified using ‘~’ sign and the above query is modified as shown:

q= text_en:(English Query) OR text_hi:(अंग्रेजी क्वेरी) OR text_ru:(английский запрос) OR text_de:(Englisch Abfrage) OR text_tr:(Ingilizce sorgu) OR text_en:**(“English Query”~3)^2**

3.2.5 Stop Word Filter Factory

Adding Stop Word Filter increase recall, thus affecting overall precision.
We added the StopFilterFactory in the schema.xml for all the language specific fields (text_en, text_hi, text_ru, text_tr, text_de and tweet_text) and also added the stop words for all the languages in the corresponding stopwords_*.txt file.

3.2.6 Language dependent tokenization

Stem filters are used for obtaining the root words of the queried words while indexing and querying. Stemming helps in improving recall. Use of stemmer also helps in obtaining more relevant documents. Stemming is specific to language based on the characteristics of the individual language.
Please find below the various stemming filters applied for each language field:

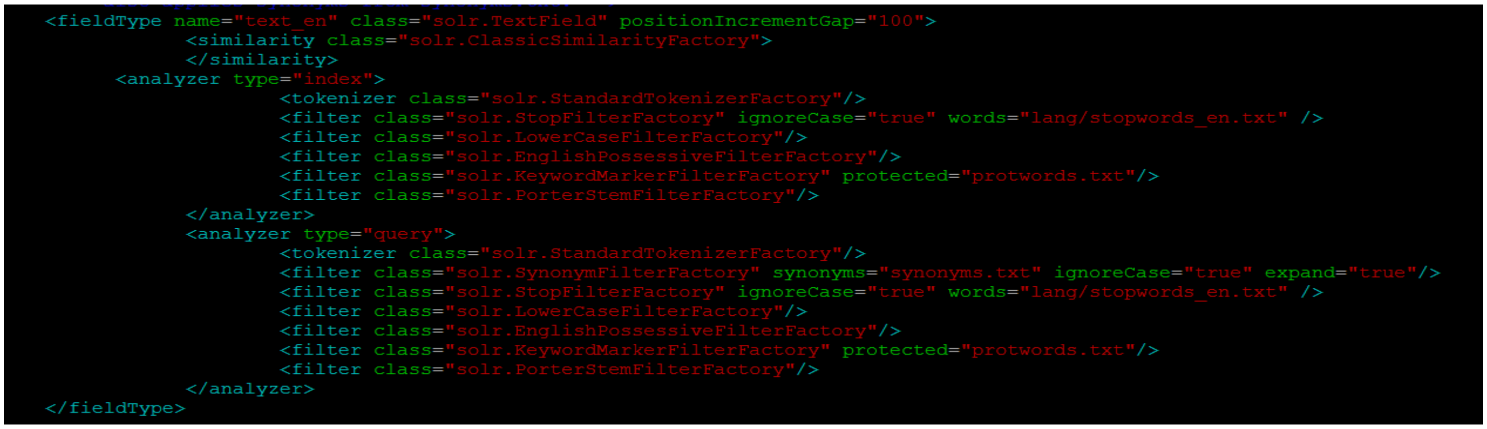
Field	Filter
text_en (English)	EnglishMinimalStemFilterFactory
text_hi (Hindi)	HindiStemFilterFactory
text_tr (Turkish)	SnowballPorterFilterFactory
text_de (German)	GermanLightStemFilterFactory
text_ru (Russian)	SnowballPorterFilterFactory

3.2.7 Query expansion using Synonym Filter Factory

We applied Synonym Filter Factory to the query field which increased the recall, matching more meaningful words. Synonyms for each language were added in the synonyms.txt file.

3.2.8 Use of KeyMarkerFilterFactory

Protects words from being modified by stemmers. A customized protected word list may be specified protwords.txt file. Any words in the protected word list will not be modified by any stemmer in Solr. For example, we can choose not to stem words like *use*, *used*, *user* etc. because if stemmed they will all map to user query US, where the user may mean United States.
This again help increase recall and thus affect overall precision.
Please find below a snapshot for the above discussed four implementations in text_en field:



```
<fieldType name="text_en" class="solr.TextField" positionIncrementGap="100">
            <similarity class="solr.ClassicSimilarityFactory">
            </similarity>
      <analyzer type="index">
                    <tokenizer class="solr.StandardTokenizerFactory"/>
                    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_en.txt" />
                    <filter class="solr.LowerCaseFilterFactory"/>
                    <filter class="solr.EnglishPossessiveFilterFactory"/>
                    <filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>
                    <filter class="solr.PorterStemFilterFactory"/>
            </analyzer>
            <analyzer type="query">
                    <tokenizer class="solr.StandardTokenizerFactory"/>
                    <filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt" ignoreCase="true" expand="true"/>
                    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_en.txt" />
                    <filter class="solr.LowerCaseFilterFactory"/>
                    <filter class="solr.EnglishPossessiveFilterFactory"/>
                    <filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>
                    <filter class="solr.PorterStemFilterFactory"/>
            </analyzer>
</fieldType>
```

Similar filters are applied to all the other text fields as well.

3.3 GUI Implementation

We have developed a website to provide an attractive GUI for the user. The Website is hosted on an Apache Web Server installed on a AWS instance.

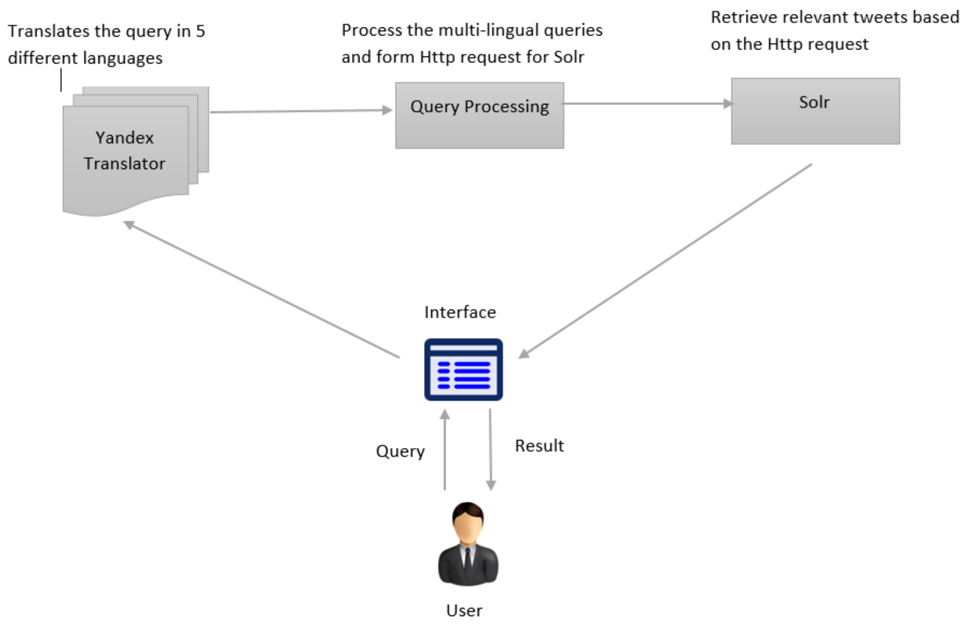
Key Technologies used:

- Jquery
- Bootstrap
- Yandex Translation JSON interface
- Solr JSON Request interface

3.3.1 Design

The user interface was designed using Bootstrap library to provide a responsive UI to the users. JQuery is used as the prime scripting language. It majorly used to perform the ajax jquery calls and to make dynamic modification to the page. Jquery Ajax has been used to integrate with the interfaces of Yandex and Solr. No API has been used for query processing or providing any other functionality.

3.3.2 Graphical Representation:


Translates the query in 5 different languages
Process the multi-lingual queries and form Http request for Solr
Retrieve relevant tweets based on the Http request
Yandex Translator
Query Processing
Solr
Interface
Query
Result
User

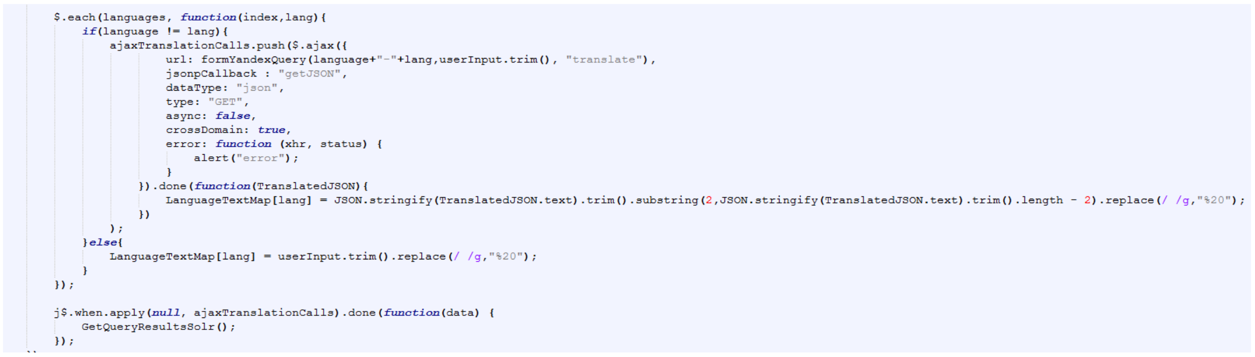
3.3.3 Code Overview:

Multiple integrations are performed using Jquery Ajax to process a query entered by a user. Below are few code snippets to provide insights of the code:

- To determine the language the user has entered the query in we use the detect interface of Yandex Translation API.

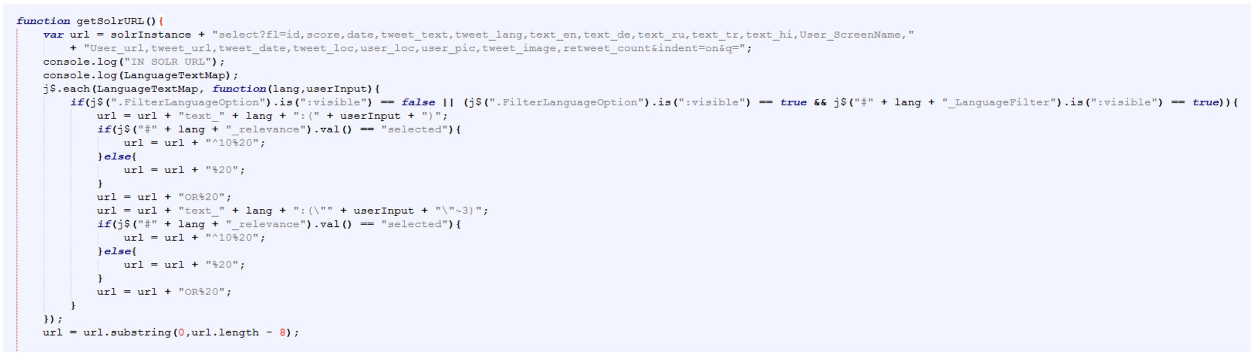
```
function getTranslatedText(lang, ele){
    console.log(ele);
    j$.when(
        j$.ajax({
            url: formYandexQuery(lang, ele.parent().find(".originalTweetText").val(), "translate"),
            jsonpCallback : 'getJSON',
            dataType: "json",
            type: "GET",
            async: false,
            crossDomain: true,
            error: function (xhr, status) {
                alert("error");
            }
        })).done(function(data) {
            ele.parent().find(".tweetText").html(data.text);
        });
}
```

- Once the language is determined then the query entered by user is translated into the four other languages using the translate interface of Yandex Translation API.



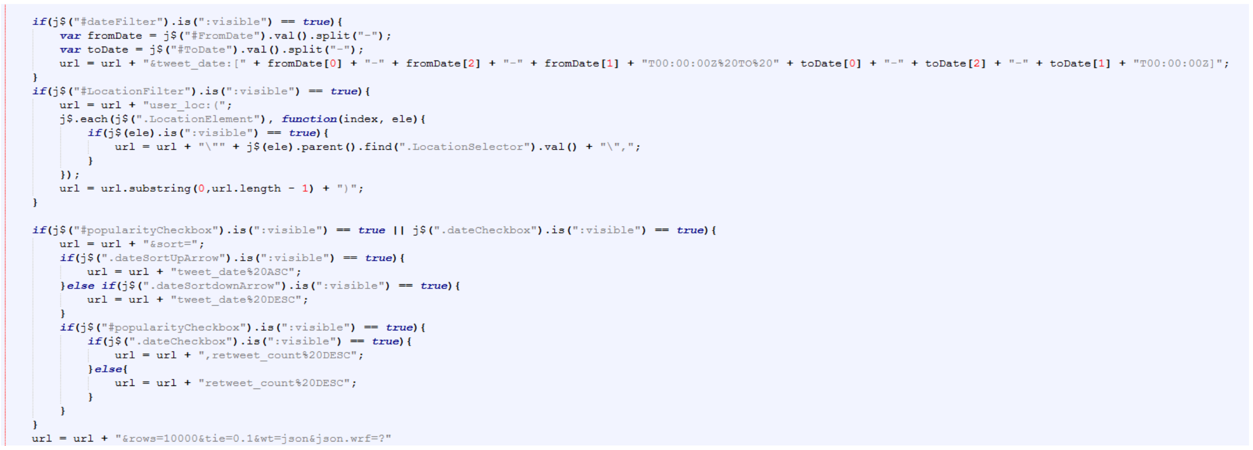
```
$.each(languages, function(index,lang){
    if(language != lang){
        ajaxTranslationCalls.push($.ajax({
                url: formYandexQuery(language+"-"+lang,userInput.trim(), "translate"),
                jsonpCallback : "getJSON",
                dataType: "json",
                type: "GET",
                async: false,
                crossDomain: true,
                error: function (xhr, status) {
                    alert("error");
                }
            }).done(function(TranslatedJSON){
                LanguageTextMap[lang] = JSON.stringify(TranslatedJSON.text).trim().substring(2,JSON.stringify(TranslatedJSON.text).trim().length - 2).replace(/ /g,"%20");
            })
        );
    }else{
        LanguageTextMap[lang] = userInput.trim().replace(/ /g,"%20");
    }
});

j$.when.apply(null, ajaxTranslationCalls).done(function(data) {
    GetQueryResultsSolr();
});
```

➢ Once the translated queries are obtained the queries are parsed into a Solr query to provide appropriate results based on the user input.



```
function getSolrURL(){
    var url = solrInstance + "select?fl=id,score,date,tweet_text,tweet_lang,text_en,text_de,text_ru,text_tr,text_hi,User_ScreenName,"
        + "User_url,tweet_url,tweet_date,tweet_loc,user_loc,user_pic,tweet_image,retweet_count&indent=on&q=";
    console.log("IN SOLR URL");
    console.log(LanguageTextMap);
    j$.each(LanguageTextMap, function(lang,userInput){
        if(j$(".FilterLanguageOption").is(":visible") == false || (j$(".FilterLanguageOption").is(":visible") == true && j$("#" + lang + "_LanguageFilter").is(":visible") == true)){
            url = url + "text_" + lang + ":(" + userInput + ")";
            if(j$("#" + lang + "_relevance").val() == "selected"){
                url = url + "^10%20";
            }else{
                url = url + "%20";
            }
            url = url + "OR%20";
            url = url + "text_" + lang + ":(\"" + userInput + "\"~3)";
            if(j$("#" + lang + "_relevance").val() == "selected"){
                url = url + "^10%20";
            }else{
                url = url + "%20";
            }
            url = url + "OR%20";
        }
    });
    url = url.substring(0,url.length - 8);
```

➢ The parsing logic takes care of all the user selected filters and sorts. Query boosting is also provided in case the user want to specify certain languages are the more relevant languages.

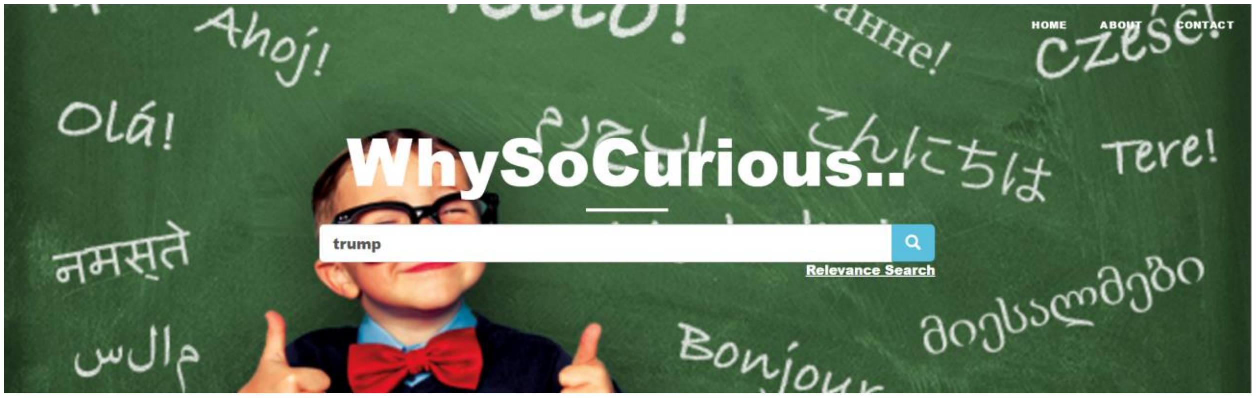


```
if(j$("#dateFilter").is(":visible") == true){
    var fromDate = j$("#FromDate").val().split("-");
    var toDate = j$("#ToDate").val().split("-");
    url = url + "&tweet_date:[" + fromDate[0] + "-" + fromDate[2] + "-" + fromDate[1] + "T00:00:00Z%20TO%20" + toDate[0] + "-" + toDate[2] + "-" + toDate[1] + "T00:00:00Z]";
}
if(j$("#LocationFilter").is(":visible") == true){
    url = url + "user_loc:(";
    j$.each(j$(".LocationElement"), function(index, ele){
        if(j$(ele).is(":visible") == true){
            url = url + "\"" + j$(ele).parent().find(".LocationSelector").val() + "\",";
        }
    });
    url = url.substring(0,url.length - 1) + ")";
}

if(j$("#popularityCheckbox").is(":visible") == true || j$(".dateCheckbox").is(":visible") == true){
    url = url + "&sort=";
    if(j$(".dateSortUpArrow").is(":visible") == true){
        url = url + "tweet_date%20ASC";
    }else if(j$(".dateSortdownArrow").is(":visible") == true){
        url = url + "tweet_date%20DESC";
    }
    if(j$("#popularityCheckbox").is(":visible") == true){
        if(j$(".dateCheckbox").is(":visible") == true){
            url = url + ",retweet_count%20DESC";
        }else{
            url = url + "retweet_count%20DESC";
        }
    }
}
url = url + "&rows=10000&tie=0.1&wt=json&json.wrf=?"
```

4. Detailed GUI tour

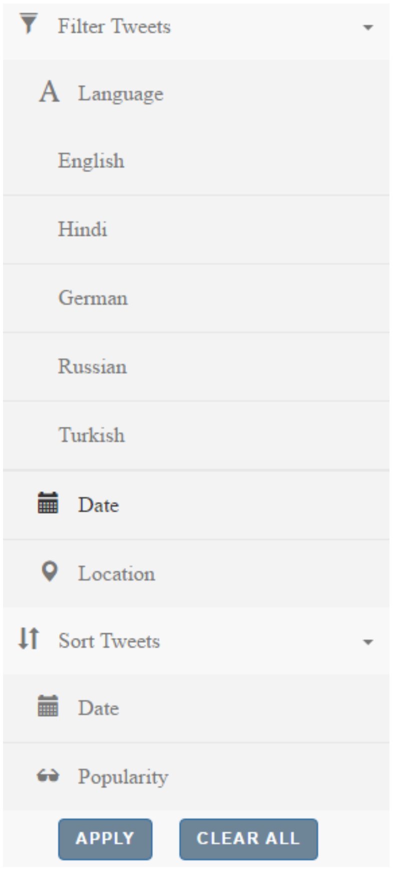
Please refer to the snapshots below with the descriptions:

1 Home Page



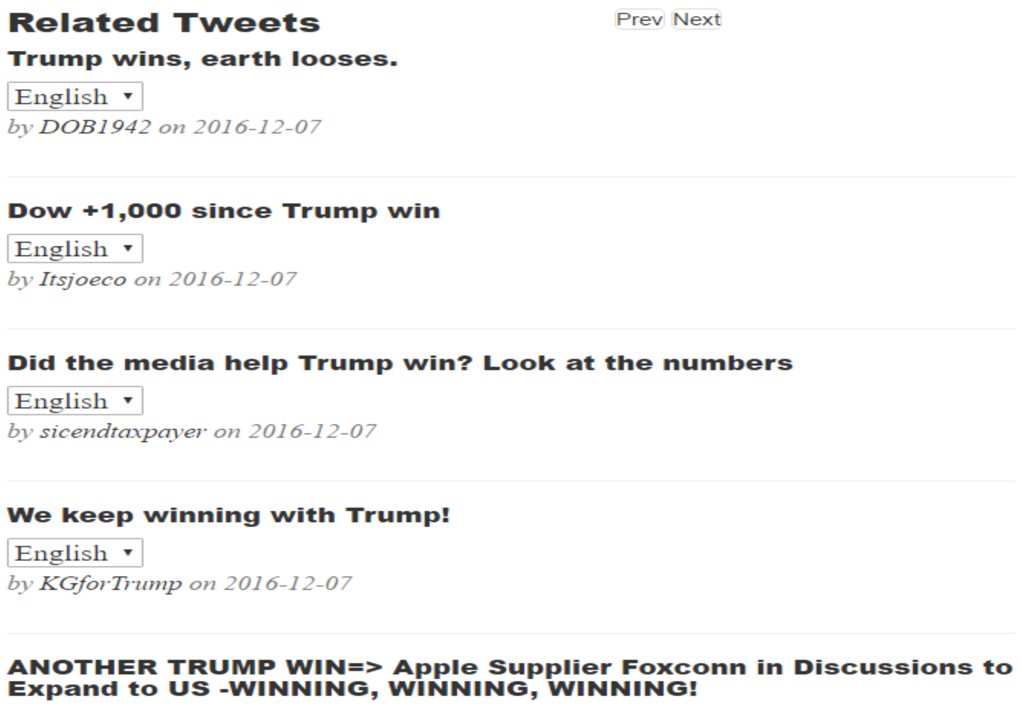
Home Page contains a Search box where the user can enter the query in any language of his preference and hit enter key to get the relevant tweets. Also, there is an option of *Relevance Search*, described in detail below.
Seen on the top right corner are buttons to navigate to the Home, About and Contact page.

2 Filter and Sort



The system provides three filters: Language, Date and Location. On clicking on Language, the filter bar expands to give the user the different language options. Similarly, on clicking on Date the filter bar expands and asks the user for FROM Date and TO Date. Filters for locations can also be added in the same way.
User can Sort the tweets based on two parameters: Date and Popularity.
Date sorting can be either ascending (Old to New) or descending (New to Old) whereas Popularity sorting considers the number of retweets for sorting. Ascending or Descending can be toggled by clicking multiple times on the Date button.
Once all the filters and sorting rules are set, the user can click on the *Apply* button. There is also an option to clear all the filters and sorting rules.

3 Related Tweets



Related Tweets

Prev Next

Trump wins, earth looses.

English ▾

by DOB1942 on 2016-12-07

Dow +1,000 since Trump win

English ▾

by Itsjoeco on 2016-12-07

Did the media help Trump win? Look at the numbers

English ▾

by sicendtaxpayer on 2016-12-07

We keep winning with Trump!

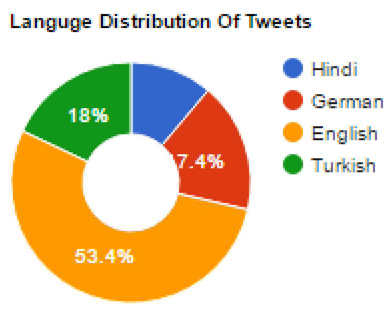
English ▾

by KGforTrump on 2016-12-07

ANOTHER TRUMP WIN=> Apple Supplier Foxconn in Discussions to Expand to US -WINNING, WINNING, WINNING!

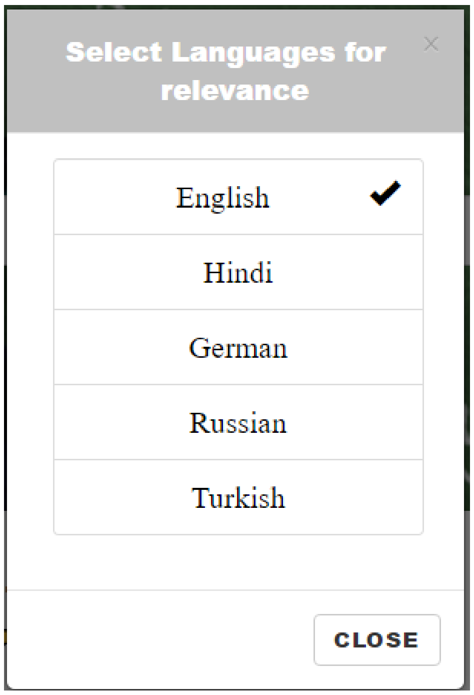
This is the heart of the page, where all the relevant tweets are displayed along with the User ScreenName, TimeStamp of the tweet and the media associated with the tweet. It also provides pagination, wherein only 10 tweets are displayed per page, with the top 10 tweets displayed on the first page.
It also has the option to navigate either to the tweet or the User's profile page by clicking on the tweet or the Username respectively.
An important feature incorporated here is a drop-down associated with every tweet to change the language of that tweet. It is defaulted to the language of the original tweet.

4 Graphs



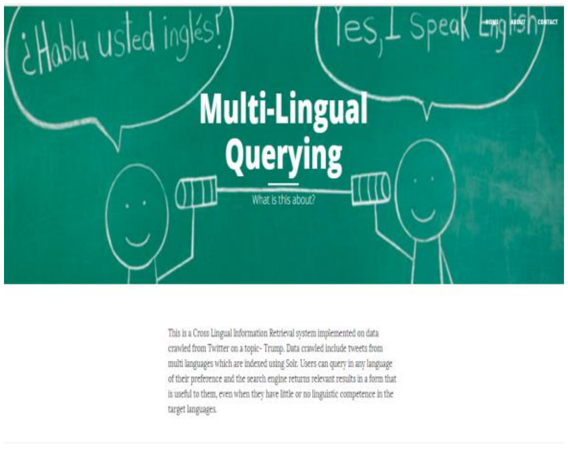
This section displays a graph plotted for the number of tweets against Language.

5 Relevance Searching

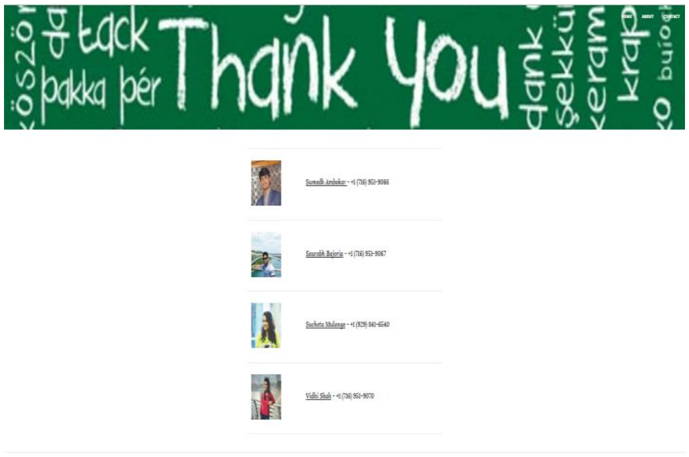


Relevance search is analogous to Advanced Search, used to give higher relevance to tweets from a particular language.

6 About and Contact



This is a Cross Lingual Information Retrieval system implemented on data crawled from Twitter on a topic- Trump. Data crawled include tweets from multi languages which are indexed using Solr. Users can query in any language of their preference and the search engine returns relevant results in a form that is useful to them, even when they have little or no linguistic competence in the target languages.



Sumedh Ambokar - +1 (716) 951-9066

Saurabh Bajoria - +1 (716) 951-9067

Sucheta Mulange - +1 (929) 841-6540

Vidhi Shah - +1 (716) 951-9070

Website: http://54.202.235.49/

5. Future improvements

Certain improvements can be made on the existing system based on the problems we encountered.

5.1 One major problem was of query/document translation.

- While translating from source language to target language, more than one translation may be possible. Selecting appropriate translation is a challenge and depends on the translator used.
  For example, the word *defeat* when converted to hindi changes to हार, but when converted back to English changes to Necklace.
- Translation of proper Noun can sometimes be vague and return wrong results.

5.2 The system could also suggest language relevance to the user based on comparison with the scores of the tweets in other languages.

6. Bibliography

- http://getbootstrap.com/
- https://jquery.com/
- https://cwiki.apache.org/confluence/display/solr/Apache+Solr+Reference+Guide
- https://en.wikipedia.org/wiki/Cross-language_information_retrieval

7. Addendum

index.html about.html index.html