



NOODLE WEB SEARCH ENGINE

IR Project - Part C



DECEMBER 22, 2015

UNIVERSITY AT BUFFALO

Submitted By:

| | |
|-----------------------------|----------|
| Agradeep Khanra | 50169196 |
| Twinkle Asthana | 50169071 |
| Akshay Kumar | 50169103 |
| Oshin Patwa | 50169203 |
| Prithvi G Indrakumar | 50169089 |
| Uttara Asthana | 50168804 |

Table of Contents

- Introduction.....2**
- Data Collection and Source.....2**
- Index.....2**
- Front-End Description.....2**
- Back-End Description.....2**
- Project Components.....2**
 - Content Tagging.....3
 - Cross Lingual Search.....4
 - Data Analytics.....6
 - Faceted Search.....8
 - Ranking Tweets.....10
- Conclusion.....11**

▪ INTRODUCTION

The main aim of this project is to build a multilingual faceted search system having a GUI that allows users to search and browse multilingual data based on various criteria like multi-lingual search, hashtag faceted search, data analytics, etc. We are using twitter and Facebook data as our multilingual social media corpus. Some of the topics reflected in the dataset are Syrian refugee crisis, recent Paris attacks, California shooting incident and some other prominent topics related to politics, sports, technology and recent news. Our corpus is not restricted to English language we have also included German, French and Russian posts. We have implemented Content Tagging, Faceted Search, Cross Lingual Retrieval Analysis, Ranking Tweets and Graphical Analysis as some of our functionalities.

▪ DATA COLLECTION AND SOURCE

Sources: Facebook, Twitter and News websites

Languages chosen: English, German, Russian and French

Topics: Paris Attacks, Syrian Refugee crisis, 2016 US Presidential elections, California shooting incident, Technology, Sports, Politics, Recent news

Around 12000 posts were collected on all the above mentioned data and languages.

▪ INDEX

We have handled removal of duplicate tweets while indexing, that is retweets will not be indexed. We have handled this by initially storing all the retrieved data in the text field which is handled in the Solr schema. By setting a unique key on the text field and not tokenizing it we have managed to remove duplication of tweets. On querying this data we get unique data in Json format (as wt=json) . Following is the query used to display unique data,

http://agra1992.koding.io:8983/solr/newcore/select?q=%3A*&rows=10000&wt=json&indent=true

This newly obtained data can then be saved in another file which can be re-indexed. The newly indexed data will not have any repeated tweets. We have chosen VSM (Vector space model) IR model for query processing.

▪ FRONT-END DESCRIPTION

The front-end of this project has been developed using various technologies. We used HTML5/CSS3 to build the overall feel of the website. jQuery was used to add more dynamic functionality to our website like check box persistence for faceted search and other such functionalities. Our site is built using the Twitter Bootstrap framework which allowed us to lay the contents of the main search page in three distinctive columns. The first column stores the facet options and the other two columns displays the required information. The overall look and feel of the site has been kept rather up-beat and it uses a card-system to display data as in many popular social media sites like Facebook, Tumblr, etc.

▪ BACK-END DESCRIPTION

Behind our gorgeous front-end, we have added the main crux of functionality of our website using PHP and Solr. We are using PHP to dynamically interact with Solr to retrieve query results from the server. All query results are returned as JSON and they are appropriately parsed and formatted into the front-end. We are also interacting with several APIs using PHP to dynamically retrieve more information about tweets and format them onto the Front End. Overall, we have been able to achieve a great sync between the look and feel of the website and the actual back-end functionality.

PROJECT COMPONENTS

1. CONTENT TAGGING

The main programming environment is in the “**noodle.php**” file. We are accessing all the API functionalities through the API object constructed for the class alchemy API.

Tagging refers to the metadata that is assigned to a piece of content by the content creator and the readers/users of the content (the latter called collaborative tagging). Content Tagging is fundamentally a means to classify content to make it structured, indexed and – ultimately – useful. In today’s generation of social network we can appreciate how a #hashtag that is used at the end of a tweet; structures, groups and orders that single tweet in the context of other tweets that contain the same hashtag. We have implemented content tagging on the “Hashtag” field which is perhaps the most popular example of tagging in everyday parlance.

We have stored the entity information in Json format in a variable \$response and are processing further only if the response is set to ‘OK’. In case there is no data or content to tag we are handling this situation by displaying the message “No content tagging available”. There is a modal associated with every post. Each view button which is an object of the GUI environment has a unique ID associated with it so that they can uniquely point to a separate modal. Thus the ‘view’ button displays the text, type, relevance and sentiment associated with each entity as shown in the Fig 2 and the code for the same is presented in Fig 3.

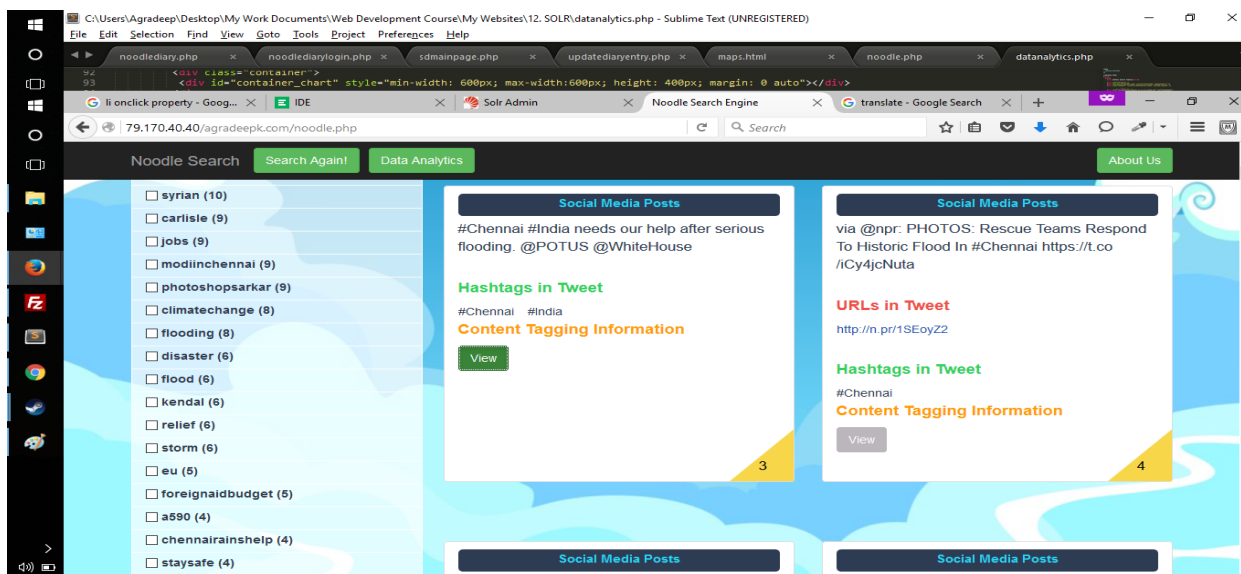


Fig 1: Output of content tagging functionality after querying “Chennai floods”

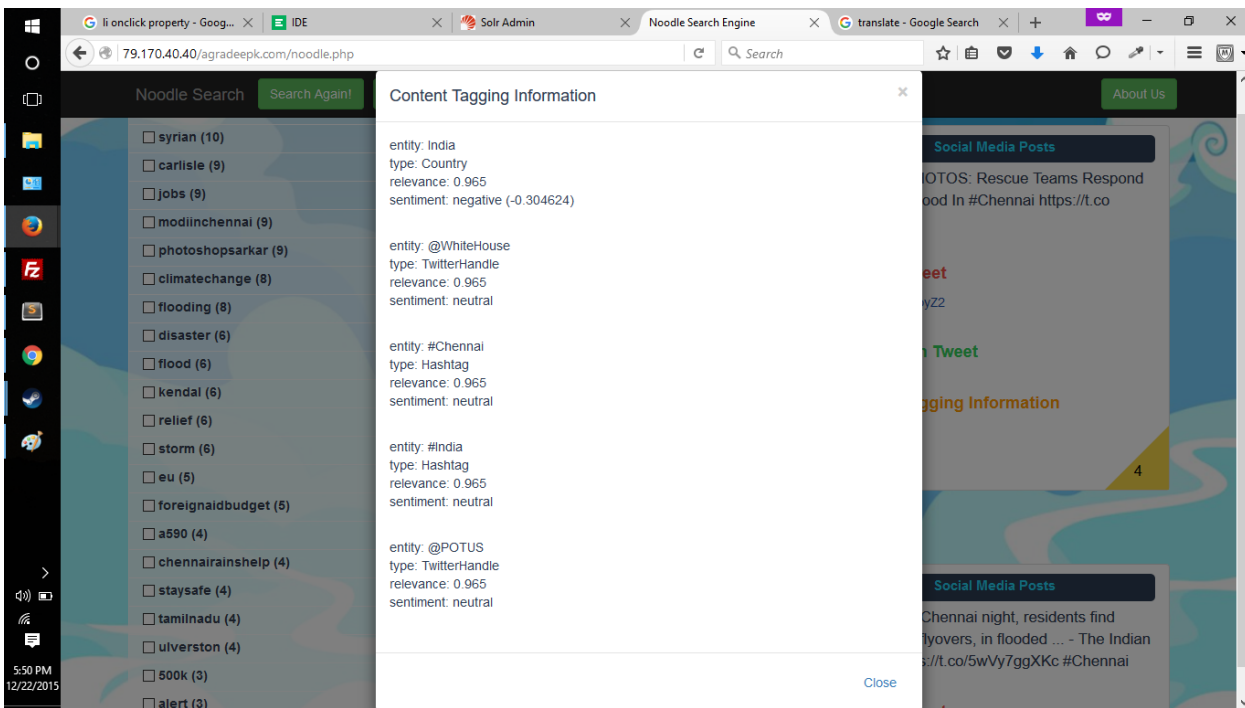


Fig 2: Output after clicking the ‘view’ button for displaying the fields associated for each entity

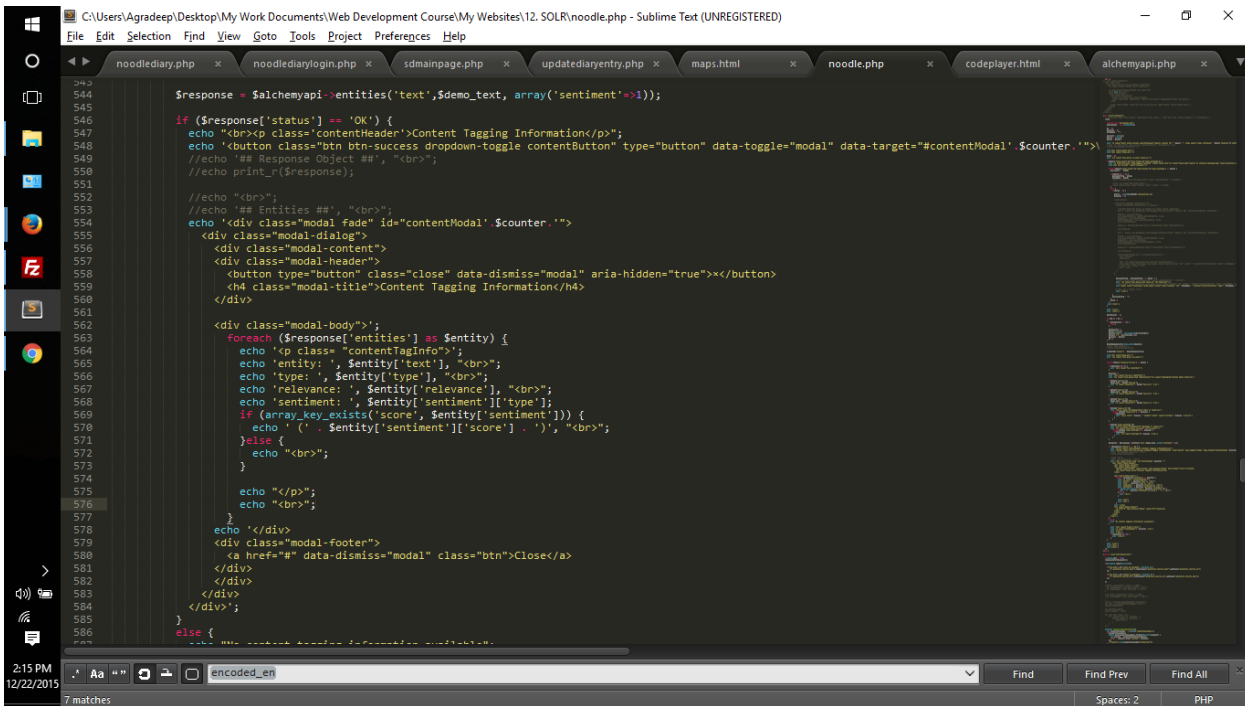


Fig 3: Code snippet for content Tagging

2. CROSS LINGUAL ANALYTICS

Cross lingual Information Retrieval (CLIR) refers to the information retrieval activities in which the query and/or documents may appear in different languages. Cross-Language information retrieval is a subfield of information retrieval dealing with retrieving information written in a language different from the language of the user's query. For example, a user may pose their query in English but retrieve relevant documents written in French.

We have used the google translate API for which we have retrieved our own API key. Google translate API is an automatic text translation tool. Whenever a user enters a query the API selects which language it is in and translates it into all other languages our system is handling that is German, English, Russian and French. Thus the main query is converted into all three languages and it is compared with its respective language indexed dataset. On executing the query given below the search system returns results in English, German, Russian and French. (implemented as \$finalQuery shown in Fig 4).

"agra1992.koding.io:8983/solr/newcore/select?q=text_en%3A(".\$encoded_en.")+OR+text_ru%3A(".\$encoded_ru.")+OR+text_de%3A(".\$encoded_de.")+OR+text_fr%3A(".\$encoded_fr.")&rows=100&fl=text_en%2Ctext_ru%2Ctext_de%2Ctext_fr%2Ctweet_urls%2Ctweet_hashtags&wt=json&indent=true&facet=true&facet.field=tweet_hashtags";

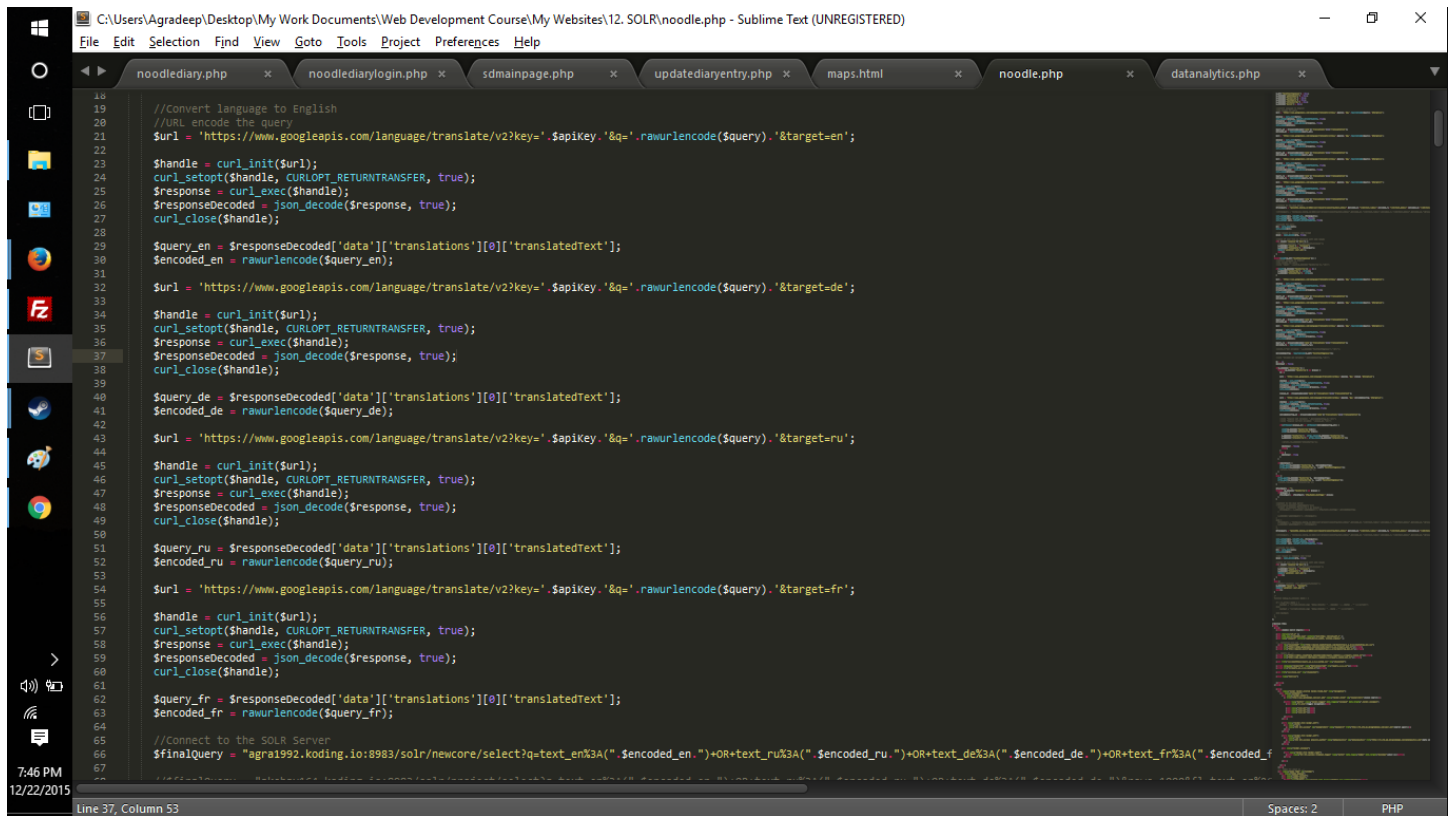


Fig 4: Code snippet for Cross Lingual Analytics functionality

On querying the Russian phrase 'Вмешательство России в Сирии' as shown in Fig 5 the output for the cross lingual functionality is displayed in Fig 6. As we can see we get the English as well as German language results as well in the 3rd and 6th post respectively on querying in Russian language.

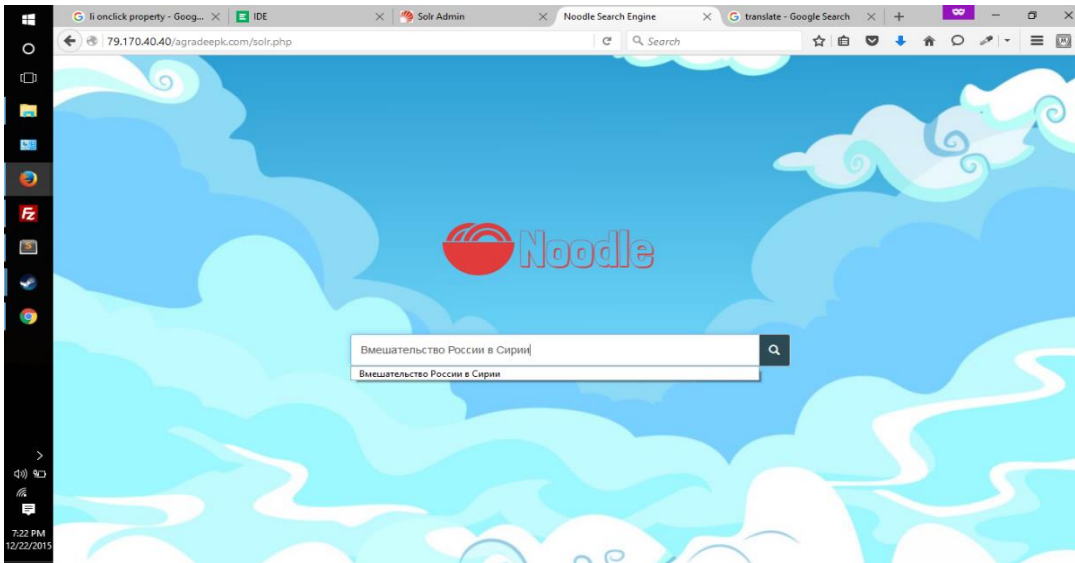


Fig 5: Russian query executed on our search system Noodle

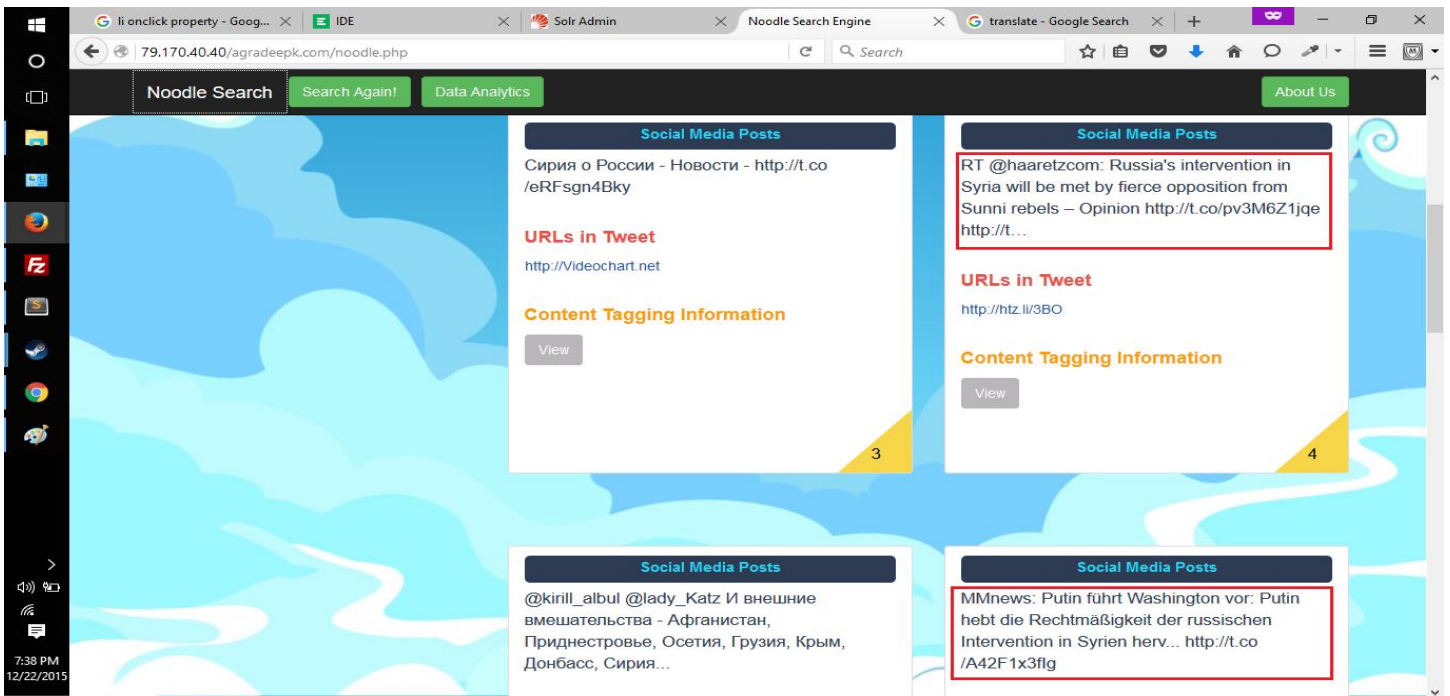


Fig 6: Output of the Russian query for displaying the cross lingual functionality.

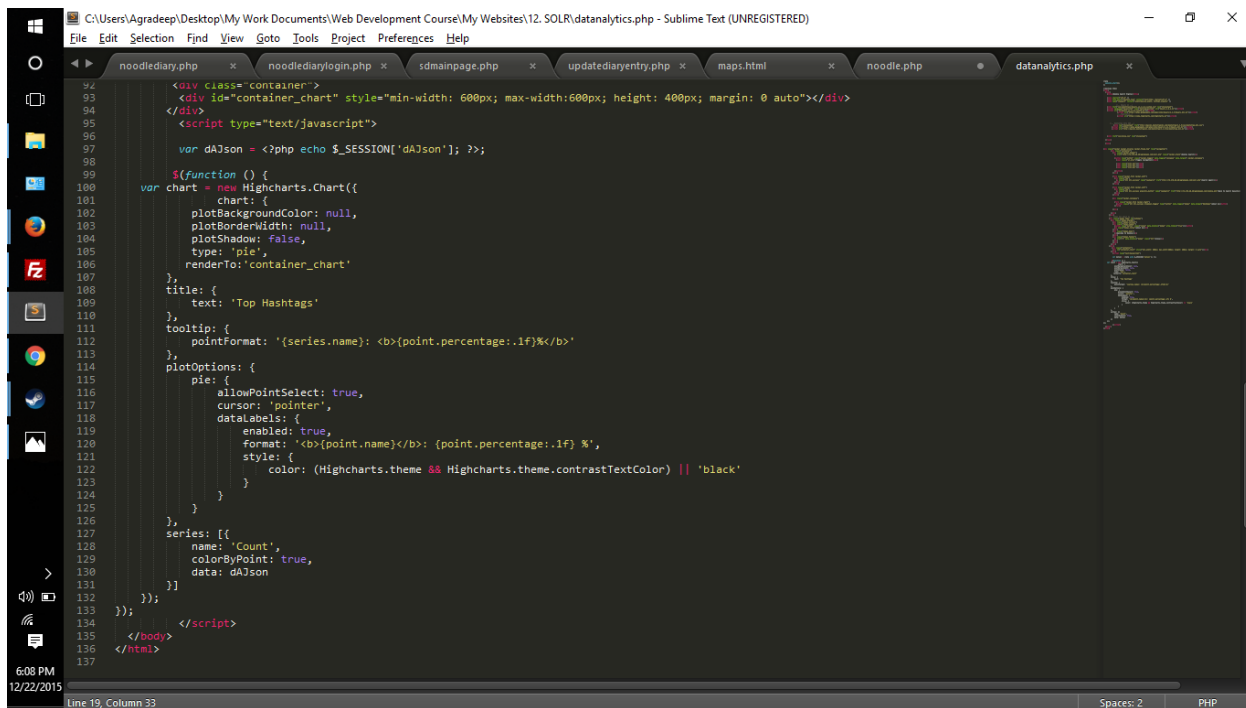
3. DATA ANALYTICS

Data Analytics is the process of examining **data** to uncover hidden patterns, unknown correlations and other useful information that can be used to make better decisions. Data is extracted and categorized to identify and analyze behavioral data and patterns, and techniques vary according to organizational requirements. In our search engine we have analyzed the data based on the top 10 hashtags of the posts.

```
473
474     $intCounter = 0;
475
476     for($i=0; ; $i++)
477     {
478         if($intCounter == 10) {
479             break;
480         }
481
482         $intCounter++;
483         $sinner=array();
484         $sinner['name'] = rawurldecode($current[$i]);
485         $sinner['y'] = $current[($i+1)];
486         $souter[] = $sinner;
487         $i++;
488     }
489
490     $jsonDataAnalytics=json_encode($souter);
491     //echo "data analytics";
492     //print_r($jsonDataAnalytics);
493
494     $_SESSION['dAJson'] = $jsonDataAnalytics;
```

Fig 7: Code snippet of Data Analytics functionality for displaying the top 10 hashtags fields

Based on the query entered, the graphical analysis of the hashtags related to the queried are displayed in the form of a pie chart. We are using High charts library for the graphical representation of data as shown in Fig 8.

The image shows a web browser window with a pie chart titled "Top Hashtags". The chart displays two segments: a large blue segment representing "floods" at 38.7% and a smaller red segment representing "chennai" at 20.1%. The browser's developer tools are open, showing the PHP code that generates the chart. The code uses the Highcharts library to create a pie chart from the data stored in the session variable \$dAJson. The chart is styled with a blue background and a white border. The data labels are displayed on the chart segments. The code also includes a tooltip for the chart points.

```
<div class="container">
<div id="container_chart" style="min-width: 600px; max-width: 600px; height: 400px; margin: 0 auto"></div>
<script type="text/javascript">
var dAJson = <?php echo $_SESSION['dAJson']; ?>;
$(function () {
var chart = new Highcharts.Chart({
chart: {
plotBackgroundColor: null,
plotBorderWidth: null,
plotShadow: false,
type: 'pie',
renderTo: 'container_chart'
},
title: {
text: 'Top Hashtags'
},
tooltip: {
pointFormat: '{series.name}: <b>{point.percentage:.1f}%</b>'
},
plotOptions: {
pie: {
allowPointSelect: true,
cursor: 'pointer',
dataLabels: {
enabled: true,
format: '{b}<b>{point.name}</b>: {point.percentage:.1f} %',
style: {
color: (Highcharts.theme && Highcharts.theme.contrastTextColor) || 'black'
}
}
}
},
series: [{
name: 'Count',
colorByPoint: true,
data: dAJson
}]
});
});
</script>
</body>
</html>
```

Fig 8: Code snippet for Data Analytics functionality using Highcharts

As shown in Fig 9, the query is "chennai floods", the data analytic's shows that the most used hashtag is "floods" (38.7%) followed by "chennai" (20.1%).

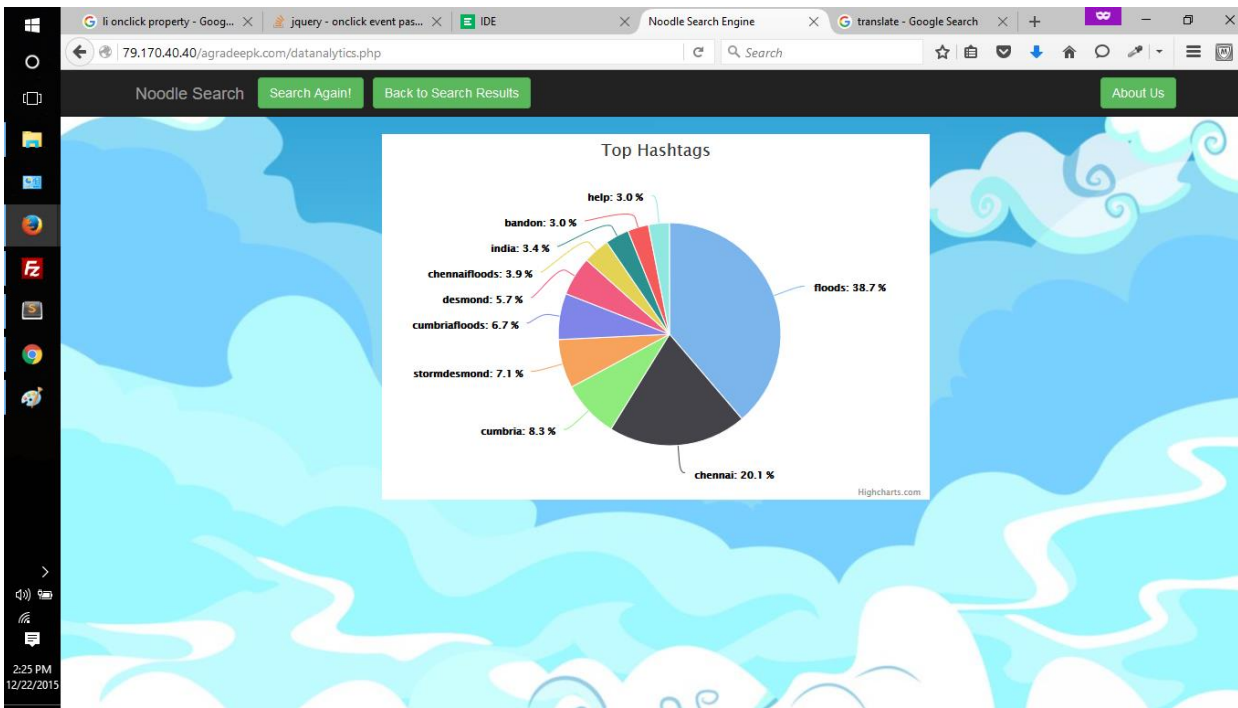
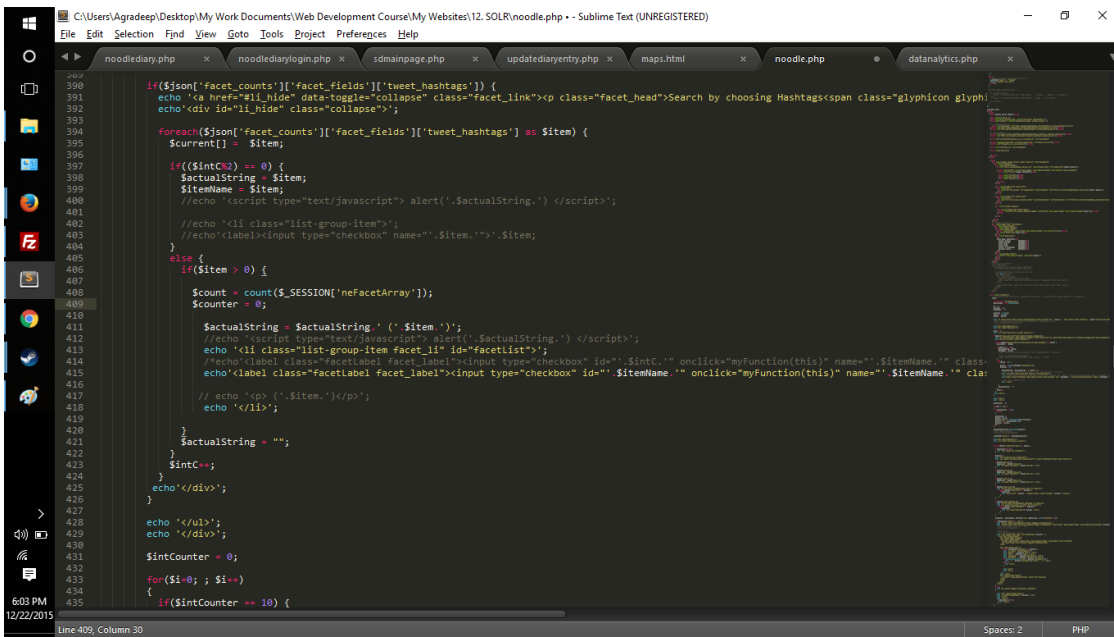


Fig 9: Graphical representation of the top 10 hashtags fields using pie charts (Data Analytics functionality)

4. FACETED SEARCH

Faceted search, also called faceted navigation or faceted browsing, is a technique for accessing information organized according to a faceted classification system, allowing users to explore a collection of information by applying multiple filters. A faceted classification system classifies each information element along multiple explicit dimensions, called facets, enabling the classifications to be accessed and ordered in multiple ways rather than in a single, pre-determined, taxonomic order. Faceted (or navigational) search uses a hierarchy structure (taxonomy) to enable users to browse information by choosing from a pre-determined set of categories. This allows a user to type in their simple query, then refine their search options by navigating/drilling down. In reality, it's an advanced search going on in the background, but instead of the user having to think of the additional search categories, it's been made easier for them by the visible folder structure.

As we can see in the code snippet for faceting for every tweet_hashtags field in \$json facet_counts we are retrieving tweet_hashtags and its corresponding count number and displaying them as a list in a separate 'div'.



```
390 if($json['facet_counts']['facet_fields']['tweet_hashtags']) {
391     echo '<a href="#li_hide" data-toggle="collapse" class="facet_link"><p class="facet_head">Search by choosing Hashtags<span class="glyphicon glyphi
392     echo '<div id="li_hide" class="collapse">';
393
394     foreach($json['facet_counts']['facet_fields']['tweet_hashtags'] as $item) {
395         $current[] = $item;
396
397         if(($intCnt == 0) {
398             $actualString = $item;
399             $itemName = $item;
400             //echo '<script type="text/javascript"> alert('.$actualString.') </script>';
401
402             //echo '<li class="list-group-item">';
403             //echo '<label><input type="checkbox" name="'.$item.'">'.$item;
404
405         } else {
406             if($item > 0) {
407                 $count = count($_SESSION['neFacetArray']);
408                 $counter = 0;
409
410                 $actualString = $actualString.' ('.$item.')';
411                 //echo '<script type="text/javascript"> alert('.$actualString.') </script>';
412                 echo '<li class="list-group-item facet_li" id="facetList">';
413                 //echo '<label class="facet_label"><input type="checkbox" id="'.$intC.'" onclick="myFunction(this)" name="'.$itemName.'" class=
414                 echo '<label class="facet_label facet_label"><input type="checkbox" id="'.$itemName.'" onclick="myFunction(this)" name="'.$itemName.'" cla
415                 // echo '<p ('.$item.')</p>';
416                 echo '</li>';
417
418             }
419             $actualString = "";
420         }
421         $intC++;
422     }
423     echo '</div>';
424 }
425
426 $intCounter = 0;
427
428 for($i=0; $i++ )
429 {
430     if($intCounter == 10) {
```

Fig 10: code snippet for faceted search functionality

We have also handled checkbox persistence as shown in Fig 11.



```
577
578
579
580 function repopulateFormElements(){
581     var elementValuesNew = $.cookie('elementValuesNew');
582     if(elementValuesNew){
583         Object.keys(elementValuesNew).forEach(function(element) {
584             var checked = elementValuesNew[element];
585             $("#" + element).prop('checked', checked);
586         });
587         $("button").text(elementValuesNew["buttonText"])
588     }
589 }
590
591 function updateCookie(){
592     var elementValuesNew = {};
593
594     $(".:checkbox").each(function(){
595         elementValuesNew[this.id] = this.checked;
596     });
597     $.cookie('elementValuesNew', elementValuesNew, { expires: 7, path: '/' })
598 }
599
600 $(".:checkbox").on("change", function(){
601     updateCookie();
602 });
603
```

Fig 11: code snippet for check box persistence handling

As we can see in Fig 12 and Fig 13 on checking the box for ‘putin’ and ‘Russia’ those posts which have putin as well as Russia in their hashtags text field are getting displayed and on deselecting ‘Russia’ the page reloads to display only those posts which have ‘putin’ as their hashtag text field.

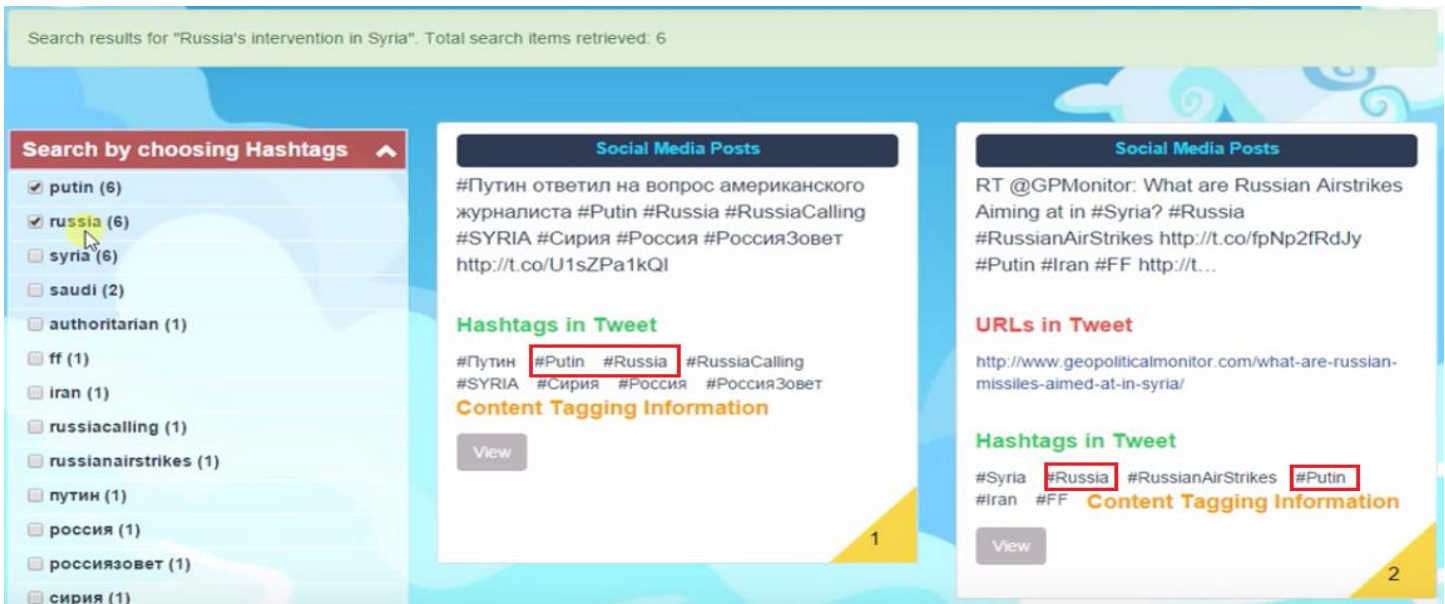


Fig 12: Output of the faceted search functionality on selecting 'putin' and 'russia'

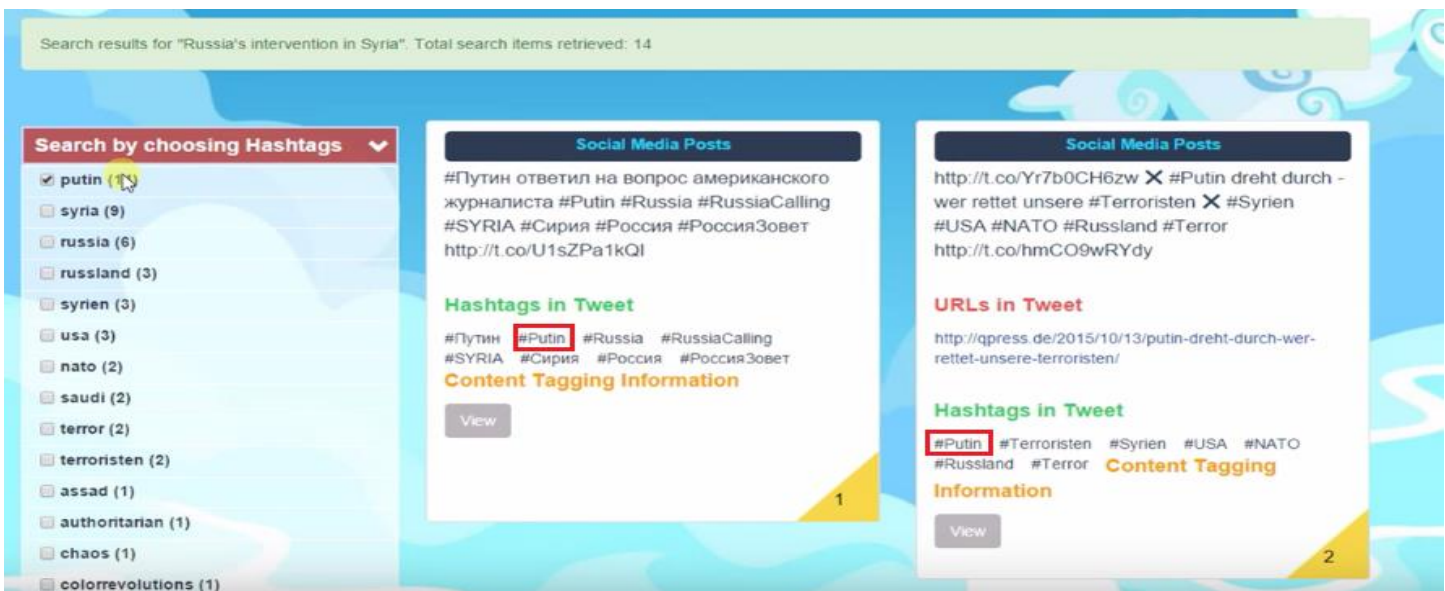


Fig 13: Output of the faceted search functionality only for 'putin' on deselecting 'russia'.

5. RANKING TWEETS

We have used the VSM model to rank our tweets. Both the teams from Project-B collaborated with each other and decided that for Project-C it'd be best to use the VSM search model. All of us felt that for our indexed data, the VSM model gave the best results. Also, we have provided appropriate query and field boosts to further improve the ranking of our search system. We did notice, however, that the best results were achieved once we incorporated the cross-lingual search functionality into our project, which boosted many relevant files to the top of the search results.

▪ CONCLUSION

Thus we implemented a full end-to-end multilingual faceted search system using various technologies like CSS3/HTML5, Bootstrap and jQuery as front end and PHP and Solr Apache as back end. The content tagging functionality is implemented on the hashtags field and have successfully implemented the multilingual search using Google translate API. Besides we are also graphically displaying top 10 hashtags fields through our Data Analytics functionality. We have also handled mynute details like duplicate tweet removal, check box persistence and re loading of the page after checking different hashtags field.