SPELL CORRECTION AND AUTO SUGGEST IMPLEMENTATION- REPORT- HOMEWORK- 4

There are two components that have been implemented in this project as part of phase 4 namely

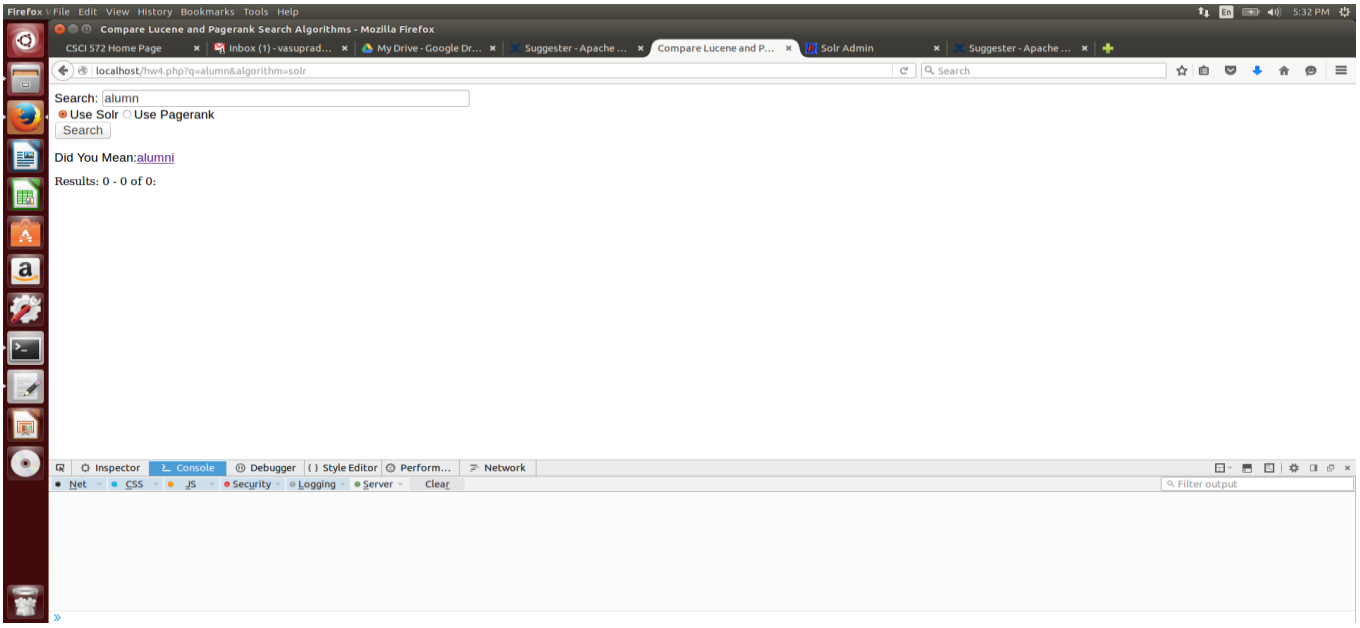
1) Spell Correction
2) Auto Suggestion

PART 1 : SPELL CORRECTION: Using Peter Norvig's Spell Corrector Program

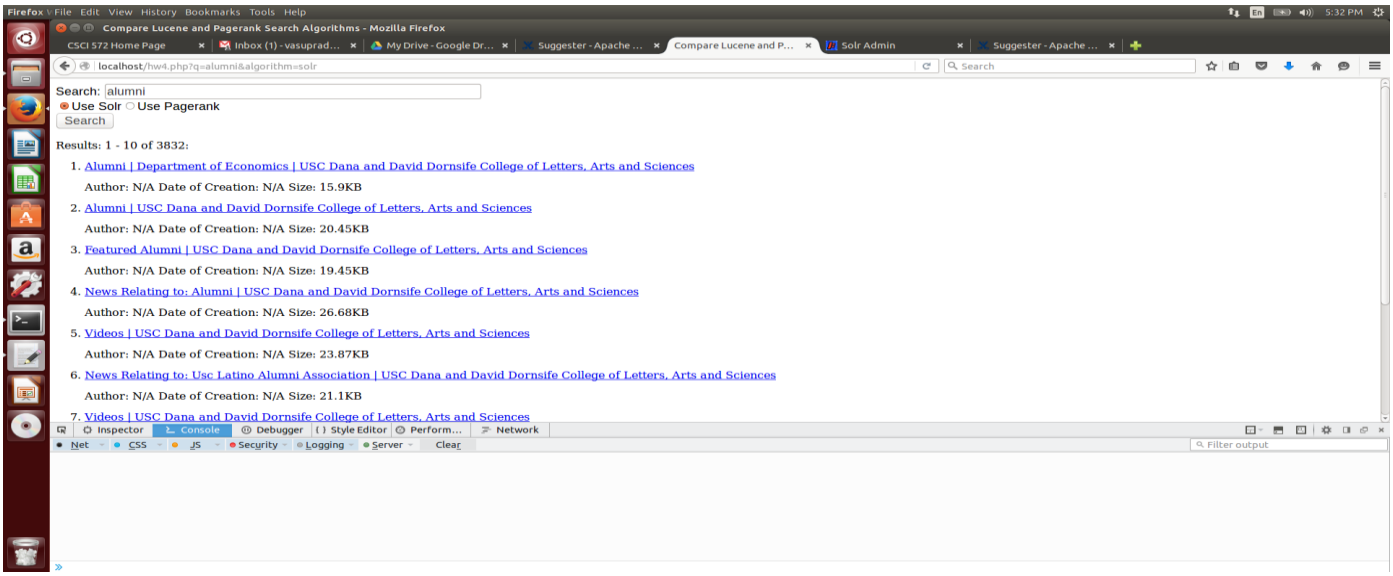
1. **Downloaded Peter Norvig's Spell Corrector PHP** program from *http://www.phpclasses.org/package/4859-PHP-Suggest-corrected-spelling-text-in-pure-PHP.html#downloadand* and placed it along with the php client files in the www folder.
2. **Parsed content** from crawled html and pdf files by using Third-Party program - Apache Tika HTML parser and Apache Tika PDF parser.The code was taken from the following sites and modified to traverse through all the files in the folders - separated the html and pdf documents into separate folders
Html Parser - http://www.tutorialspoint.com/tika/tika_extracting_html_document.htm
PDF Parser - http://www.tutorialspoint.com/tika/tika_extracting_pdf.htm
3. **The complete parsed file - big.txt** was 34.7 MB long as it included chunks of trailing whitespace and the client program would take around 6-7 seconds to load and return the result.
In order to reduce the size of the file, The command - **grep -o -E '\w+' big.txt** was used in order to extract all the words from the file and display it as a list
To replace all newline characters to space - The command - **tr '\n' ' ' < big.txt > check_dic.txt**
The file size reduced to 24.5 MB.
4.**Modifications made to Spell Corrector Program -**
Added a line - ini_set('memory_limit','-1') before the SpellCorrector Class Definition as the by default PHP scripts can load a maximum of upto 2 MB and this removes any upper bound limit.
5. **The SpellCorrector.php** is included in the client side program and the Spell::Corrector API is called for each word that belongs to the query.
6. If the query entered by the user contains multiple words, it is tokenized and each word is spell-corrected using the API and the result returned is concatenated into a new string. This string is displayed as a hyperlink to the user emulating the
Did You Mean:<Correct Spelling of query> feature implemented by Google.
Eg : If the User enters - " Universty of Southe California" in the search box
Displayed to User - *University of Southern California*
The hyperlink when clicked will fetch results based on which option - Solr or PageRank had been selected earlier for the misspelled query.
7. If the query entered is correctly spelled then, no spelling suggestions are displayed to the user.
8. The spelling correction has been restricted to only words that belong to the school crawled - (Dornsife College of Arts,Letters and Sciences) extracted by parsing the files that were downloaded during the crawling process.
9. Since Norvig's Spell Corrector algorithm works based on frequency of occurrence of a word present in the dictionary (in this case - check_dic.txt), and on edit distance(max 2 only).It calculates a probability for each word and returns the word that has the best probability of being the correct spelling.
Thus there might be some words that do not get spell corrected to the expected word if it is not present in the dictionary generated.

SCREENSHOTS:

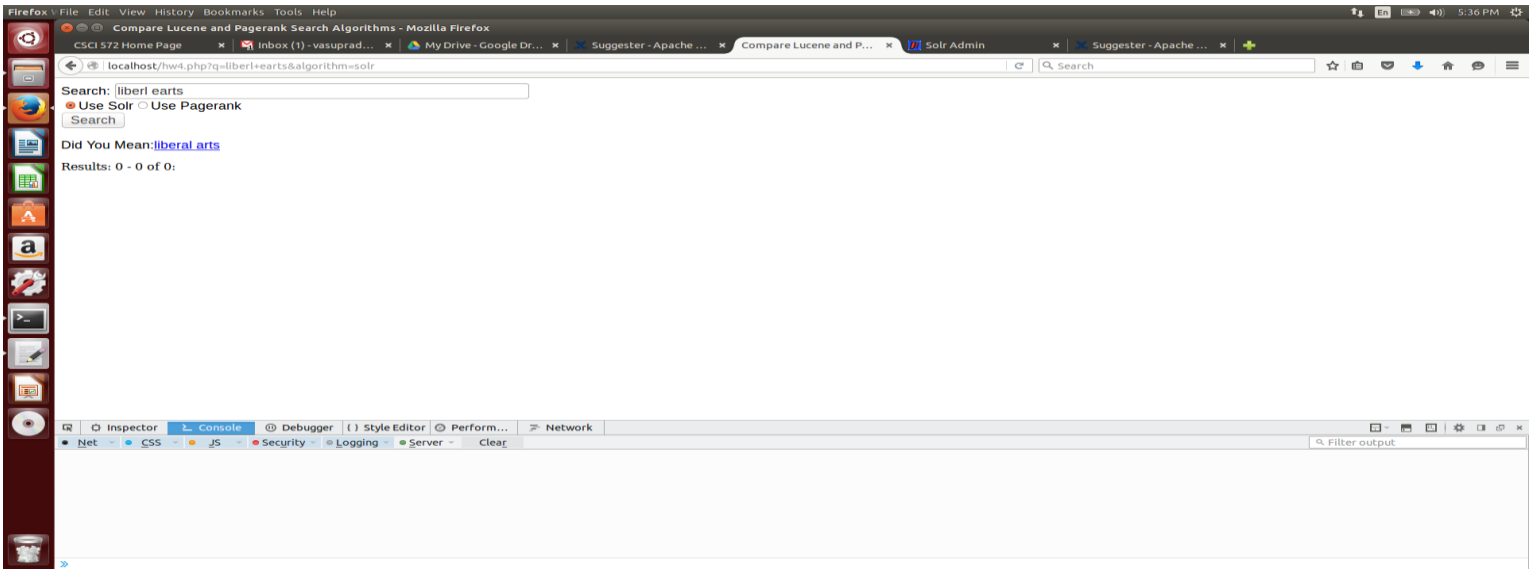
1) Single Word Spell Correction : Input : Alumn Output : alumni



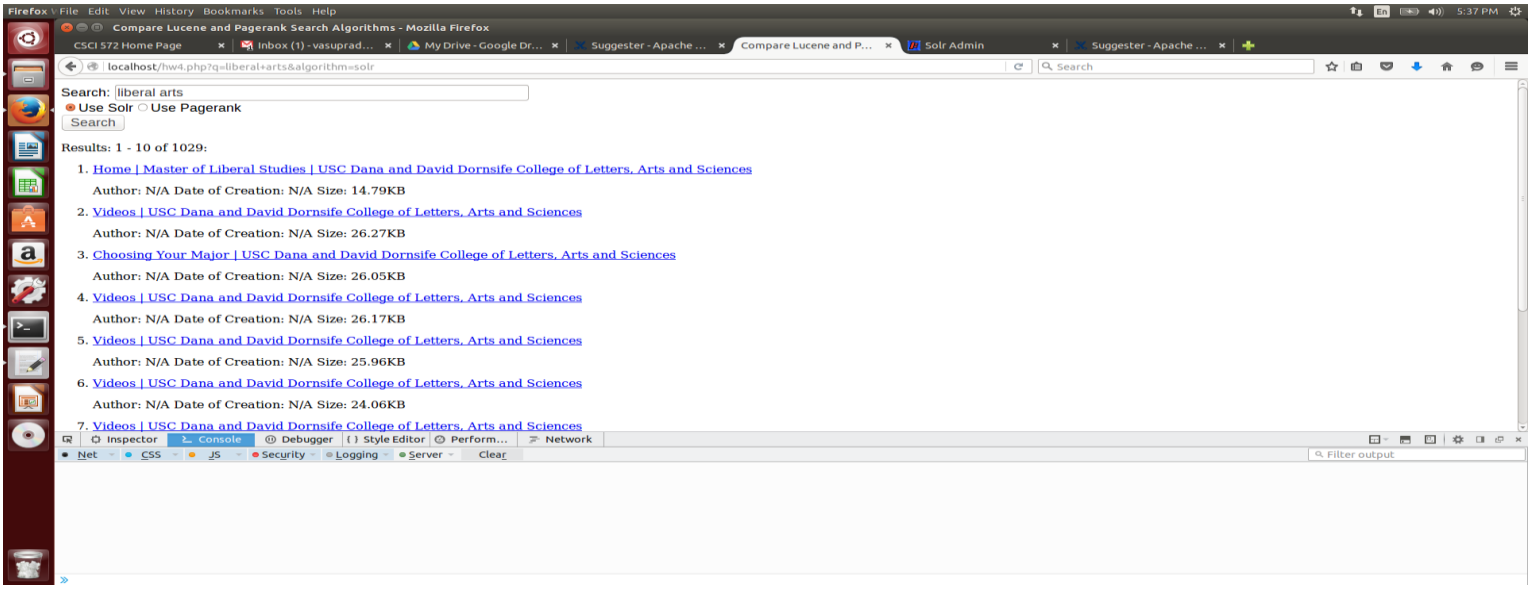
After the alumni is selected:



2) Multiple Word Spell Corrections: Input: Liberl earts Output: liberal arts



After the liberal arts is selected:



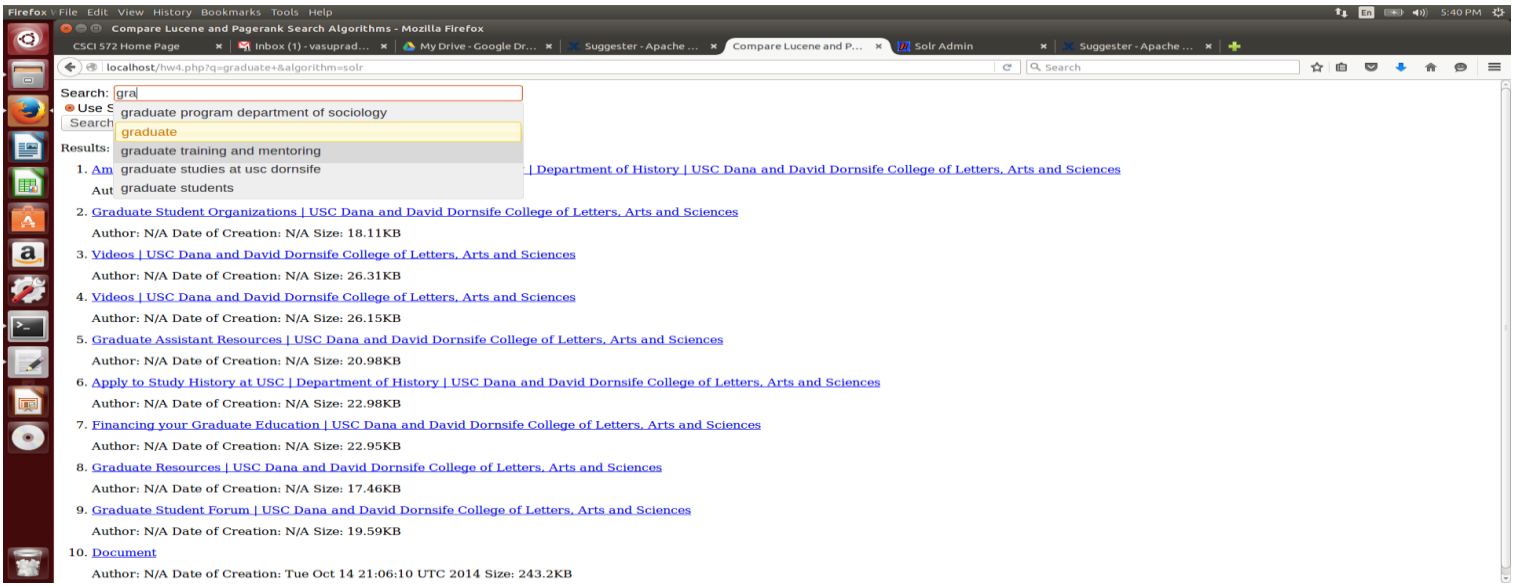
PART 2 : AUTO SUGGEST/COMPLETION:

Solution Design:

Phrase suggestions have been implemented as part of the auto suggest feature as opposed to suggestions for each word entered in the search box. This was the approach that seemed more appropriate as it will give suggestions that make sense contextually instead of treating each word individually and give a list of top 5 suggestions for each.

For eg : if the user enters : **'gra'**

Based on top weighted suggestions handled by solr's auto-suggest component internally, it returns the following results as shown in the screenshot below –



In order to implement multi-word suggestions, the files were indexed based on multiple fields such as content, description, keywords, title etc.

Changes made in Managed-Schema:

A new field - suggest_phrase_field is defined of type suggest_phrase with parameters 'indexed' and 'stored' set to true.This is the field that will return the suggestion.

```
<field name="suggest_phrase_field" type="suggest_phrase" indexed="true" stored="true"/>
```

Then CopyFields are defined for the fields:

```
<copyField source="author" dest="suggest_phrase_field"/>
<copyField source="content" dest="suggest_phrase_field"/>
<copyField source="content_type" dest="suggest_phrase_field"/>
<copyField source="description" dest="suggest_phrase_field"/>
<copyField source="keywords" dest="suggest_phrase_field"/>
<copyField source="resourcename" dest="suggest_phrase_field"/>
<copyField source="title" dest="suggest_phrase_field"/>
```

Analyzers are defined for the index and query using Solr's inbuilt various Tokenizers and Filters such as
KeywordTokenizer: Treats the entire text field entered as a single token
PatternReplaceCharFilterFactory - regex match and replace with required pattern
LowerCaseFilterFactory- to transform all characters to lower case
TrimFilterFactory- to remove all trailing whitespaces

StopFilterFactory- to remove stopwords from the suggestion.Stopwords have been defined in the stopword.txt file
HunspellStemFilterFactory- Functions based on the en_GB dictionary and affix files and stems the words
Eg: Calculating is stemmed to ----> Calculate.
It is a more robust stemmer than the PorterStemmer Factory as it looks at context before stemming

```
<fieldType name="suggest_phrase" class="solr.TextField"
multiValued="true"positionIncrementGap="100">
 <analyzer type="index">
 <charFilter class="solr.PatternReplaceCharFilterFactory" pattern="([^a-zA-Z0-9\'\s]+)" replacement=""/>
  <tokenizer class="solr.KeywordTokenizerFactory"/>
  <filter class="solr.LowerCaseFilterFactory"/>
  <filter class="solr.TrimFilterFactory"/>
  <filter class="solr.StopFilterFactory" words="stopwords.txt" ignoreCase="true"/>
  <filter class="solr.HunspellStemFilterFactory" affix="en_GB.aff" dictionary="en_GB.dic" ignoreCase="true"/>
   <filter class="solr.PatternReplaceFilterFactory" replace="all" replacement=" " pattern="(\'\{\}\[\]\(\)\*\-
;\?_\\\|\.\:])"/>
</analyzer>

<analyzer type="query">
    <charFilter class="solr.PatternReplaceCharFilterFactory" pattern="([^a-zA-Z0-9\'\s]+)" replacement
=""/>
  <tokenizer class="solr.KeywordTokenizerFactory"/>
  <filter class="solr.LowerCaseFilterFactory"/>
  <filter class="solr.TrimFilterFactory"/>
  <filter class="solr.StopFilterFactory" words="stopwords.txt" ignoreCase="true"/>
  <filter class="solr.HunspellStemFilterFactory" affix="en_GB.aff" dictionary="en_GB.dic" ignoreCase="true"/>
   <filter class="solr.PatternReplaceFilterFactory" replace="all" replacement=" " pattern="(\'\{\}\[\]\(\)\*\-
;\?_\\\|\.\:])"/>
    </analyzer>
      </fieldType>
```

Changes Made in Solrconfig.xml

A searchComponent and RequestHandler are defined for the suggest feature as follows

```
<requestHandler name="/suggest" class="solr.SearchHandler">
   <lst name="defaults">
   <str name="suggest">true</str>
     <str name="suggest.count">15</str>
     <str name="suggest.dictionary">suggest</str>
    </lst>
    <arr name="components">
      <str>suggest</str>
    </arr>
```

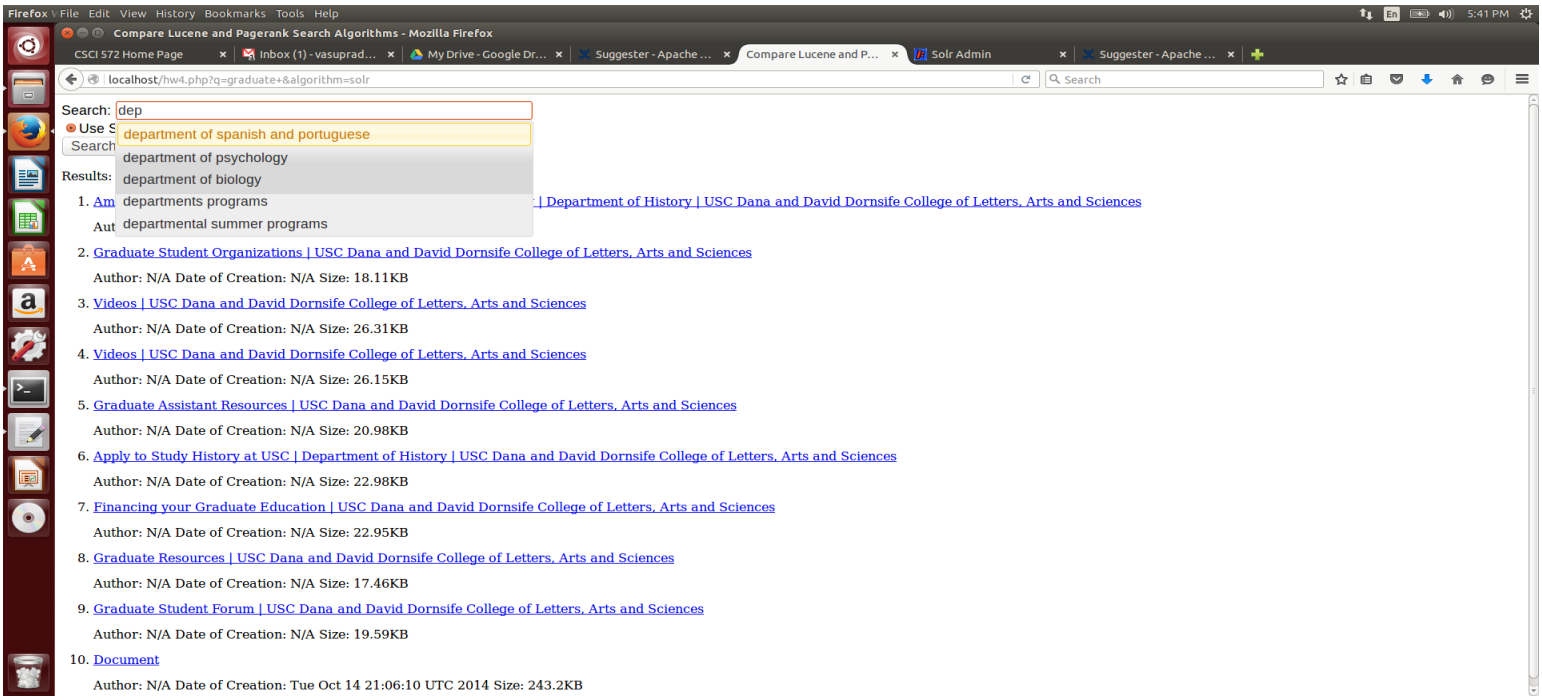
```
</requestHandler>

<searchComponent class="solr.SuggestComponent" name="suggest">
  <lst name="suggester">
    <str name="name">suggest</str>
    <str name="lookupImpl">BlendedInfixLookupFactory</str>  //  LookUp  factory  <str
name="field">suggest_phrase_field</str> // field name
    <str name="suggestAnalyzerFieldType">suggest_phrase</str> //field type name
        <str name="highlight">false</str>
  </lst>
  </searchComponent>
```

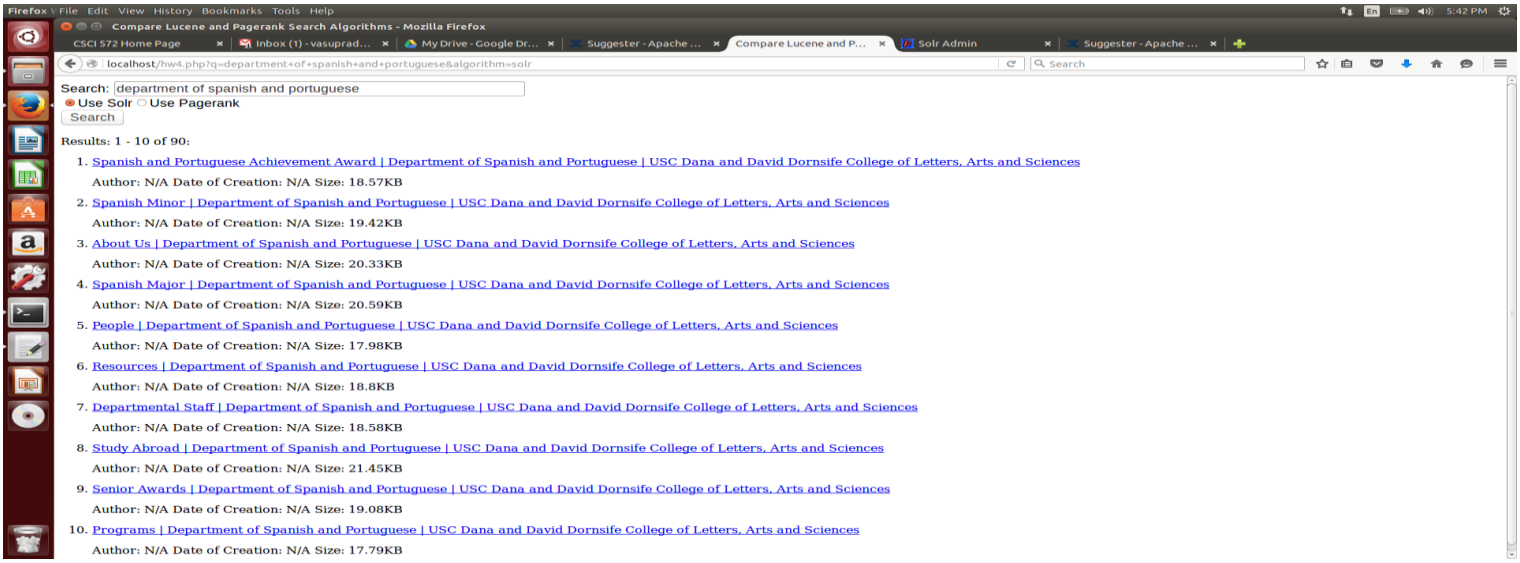
BlendedInfixLookupFactory : Performs weighted prefix matches across the matched documents.It gave better suggest results than using FuzzyLookupFactory.

Phrase Query Suggestions:

Entered input - 'dep' Suggestions: Results with prefix 'dep'



After selecting top suggestion:



Displaying Suggestions in the UI:

Used Jquery UI plugin and Ajax calls in order to populate and get suggestions in the form of a drop down as soon as the user types into the search box.Incorporated it into the client side php script.

An ajax call is made to the solr core with the suggest handler ie **http://localhost:8983/solr/hw3/suggest.**
The query is sent as the input and the results are retrieved in json format. The json output is traversed and looped through in order to access the suggest term.

It is expected that there will be noisy data (irrelevant) or suggestions that can be filtered out. The filtering is handled in **Javascript**. All the retrieved suggestions are appended into an array. Jquery **autocomplete** function makes use of this array as a source and populates the suggestion dropdown.The number of suggestions have been limited to 5 as mentioned in the homework instructions.

NOTE:
As phrase suggestions have been implemented, if a particular phrase that is being searched does not match with any prefix of the indexed terms, then the suggestions stop coming up. Even though the word might exist deeper in the term, it cannot be found. This is a drawback of the BlendedInfixFilterFactory.
Further tokenization of data and the index lookup can be refined using Solr's inbuilt methods to ensure that suggestions are more relevant.