Shriraj L Salian
8705640634

Report on Autocomplete, Spell Correct and Snippets:

Spell Correct:

The spell-correction functionality was implemented in two parts:

1. Creation of big.txt using Apache Tika
    a. Using the Apache Tika jars I extracted the html content of all webpages and created the big.txt using regular expression in Java. This list was used in Peter Norvig's spell-correction program
2. Using Peter Norvig's spell correction
    a. Here, we calculate all the possible words from an edit distance of 1 and 2. The operations used to calculate the possible words are:
        i. Insert
        ii. Delete
        iii. Transposition
        iv. Substitution/Replace
    b. From the words calculated, we return the most probable word from the corpus

For this we include the spell corrector program using "include SpellCorrctor.php" and then make a call to the corrector() function using "echo SpellCorrector::correct('octabr');"('octabr is used as an example here)

Autocomplete:

1. Autocomplete was implemented using multiple technologies. I used javascript and jquery & ajax's autocomplete for this purpose.
2. Autocomplete requires a source containing the terms that will be provided for suggestions in the drop-down menu
3. For Autocomplete suggestions we made changes to the solrconfig.xml by telling solr to use the SuggestComponent. We then add a requestHandler to allowing to configure default parameters for suggestion requests. This was used to build the 'source'.
4. To ensure autocomplete works for n-term queries, we split the query terms based on space. (Assuming queries are separated by space) Autocomplete suggests for the latest term being entered returning an autocomplete for the last term tagged to all the initial query terms. These suggestions were finally stored in an array.
5. The array was returned as a responseText to the javascript.
6. The auto-complete results are clickable, will auto-complete the word when clicked and return focus to the text box.
7. Once the user starts with a new word or a space separated word the ajax code is again activated.

Snippets:

1. I used simple_html_dom parser to parse plaintext from html pages. File_get_contents was used to get the contents of the file.

2. The content was then split into an array & strings matched to find the index of the words in the query. Using these indices, I determined the string to be printed as the snippet. The snippet lines were chosen as follows:

    i) For multiple term queries, try finding a sentence with all the terms together.

    ii) If not, return the sentence that has all the terms in it, even if they are not together or in same order.

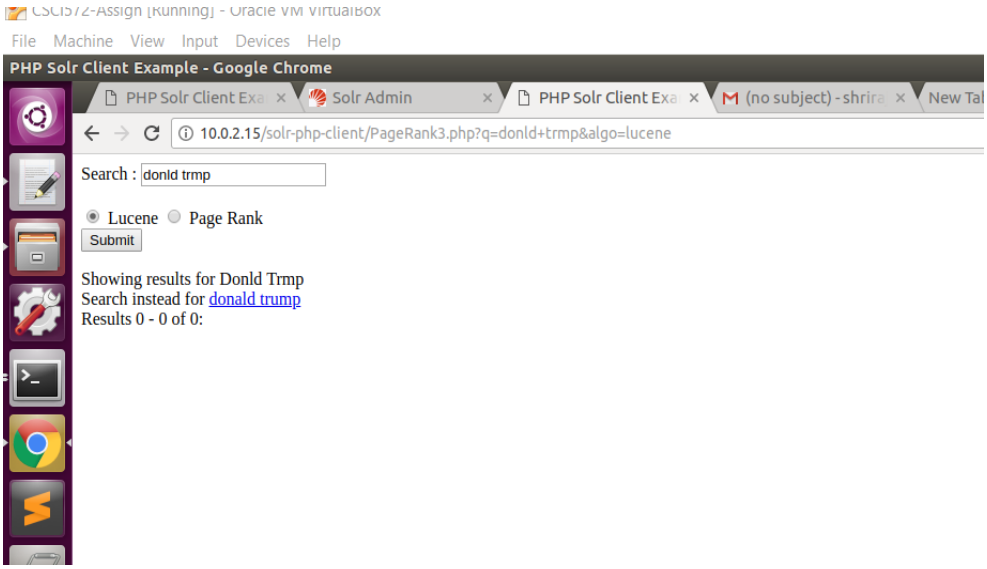
iii) If none above cases fail are found, return the first sentence with at least one query term in it.

iv) If no match is found, then display text

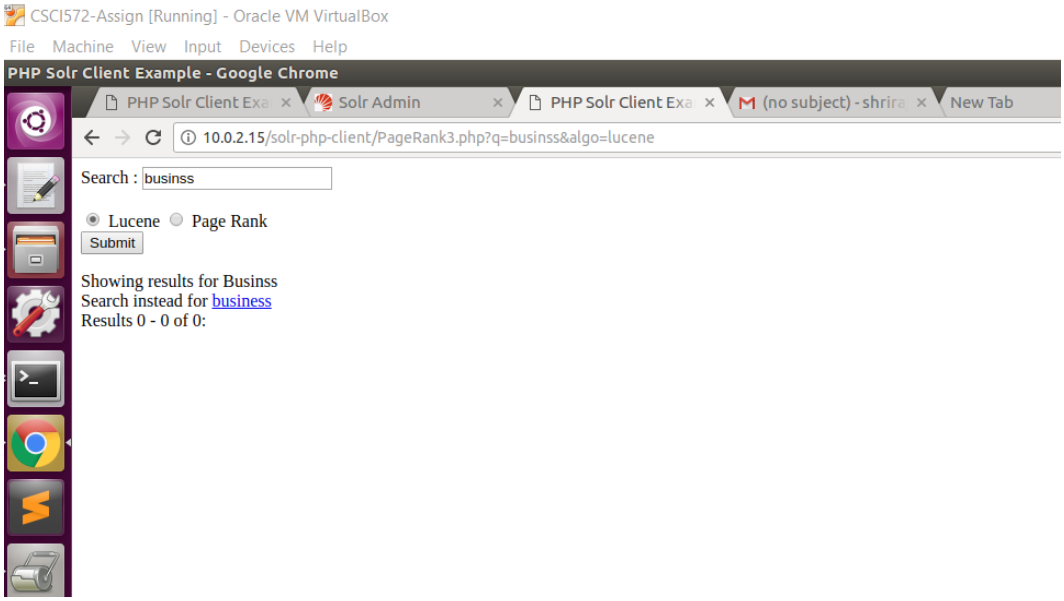
3. I selected the starting and the ending positions of the snippet lines that would be printed. In case there are no snippet found, the code would search for a good match in the description
4. I then highlighted the query terms that appeared in the snippet

Autocomplete

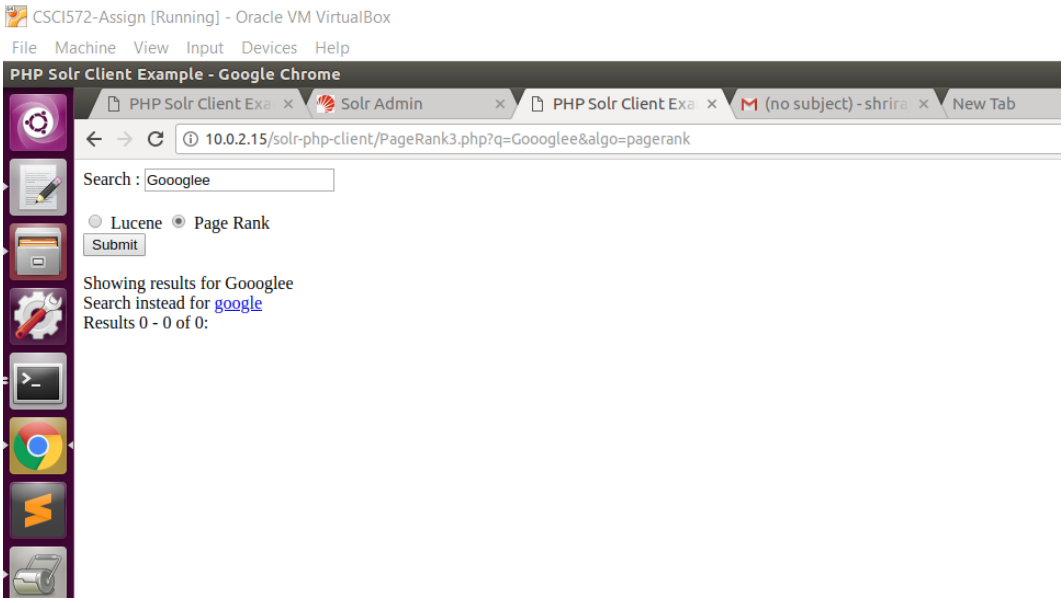
1.Donld Trump



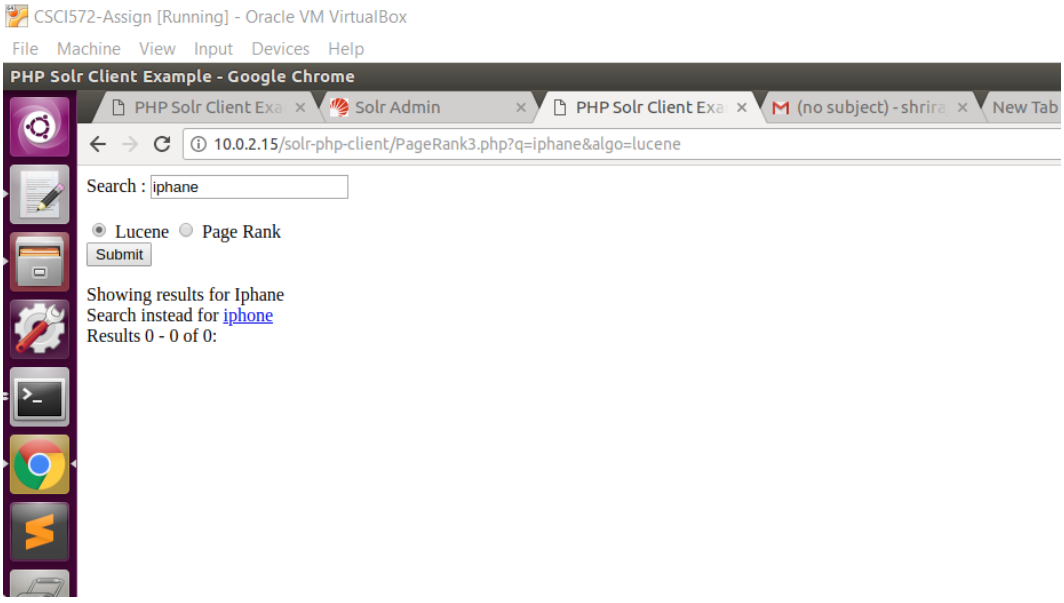
2.Business



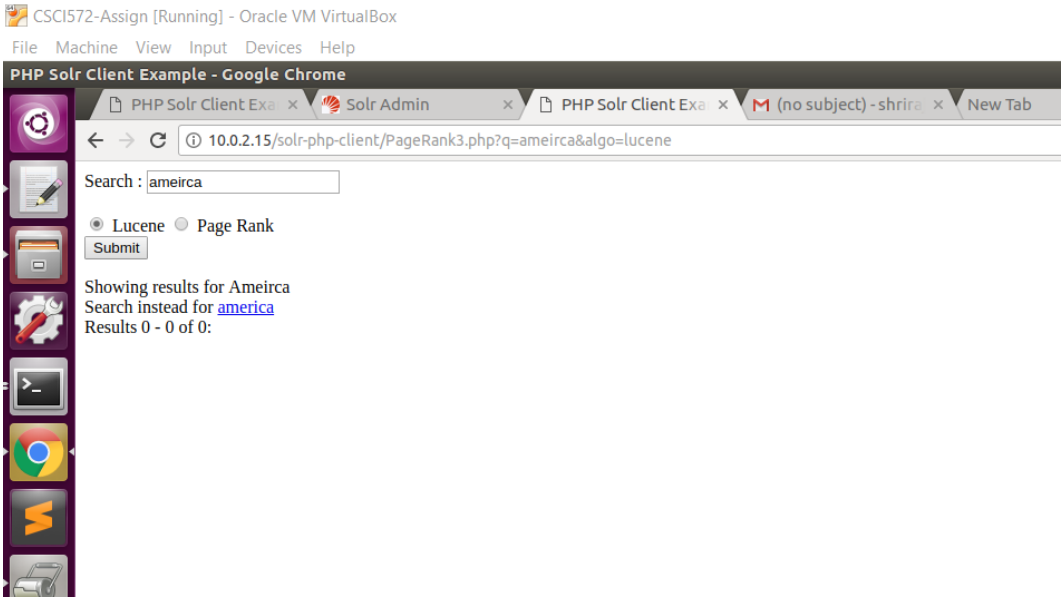
3.Goooglee



4.Iphane

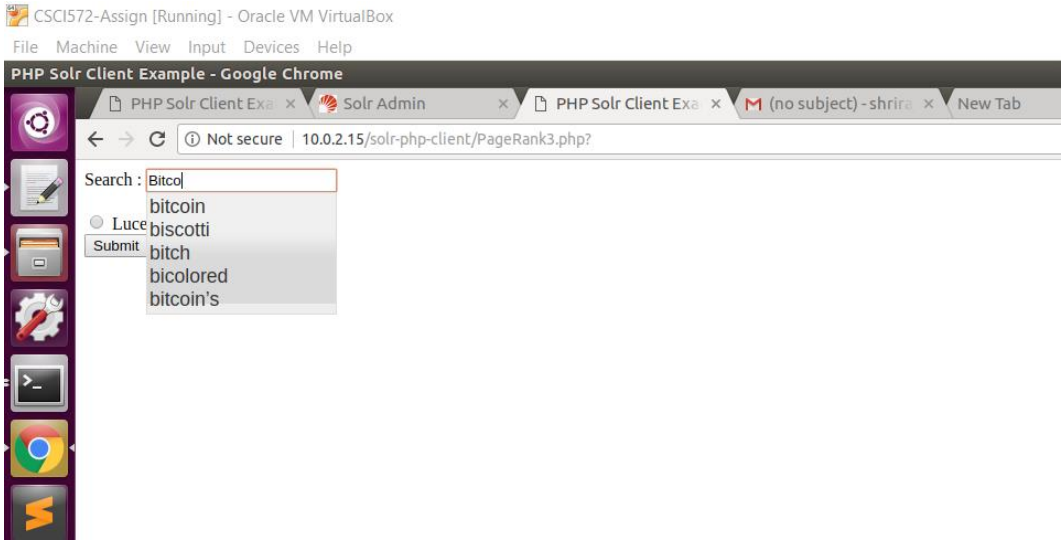


5. ameirca

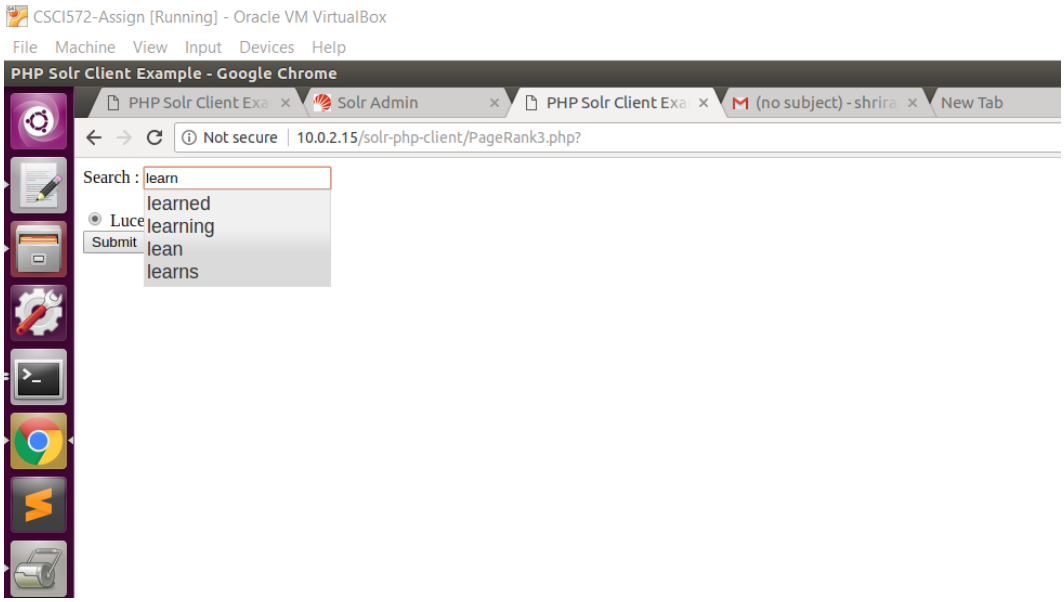


Autocomplete

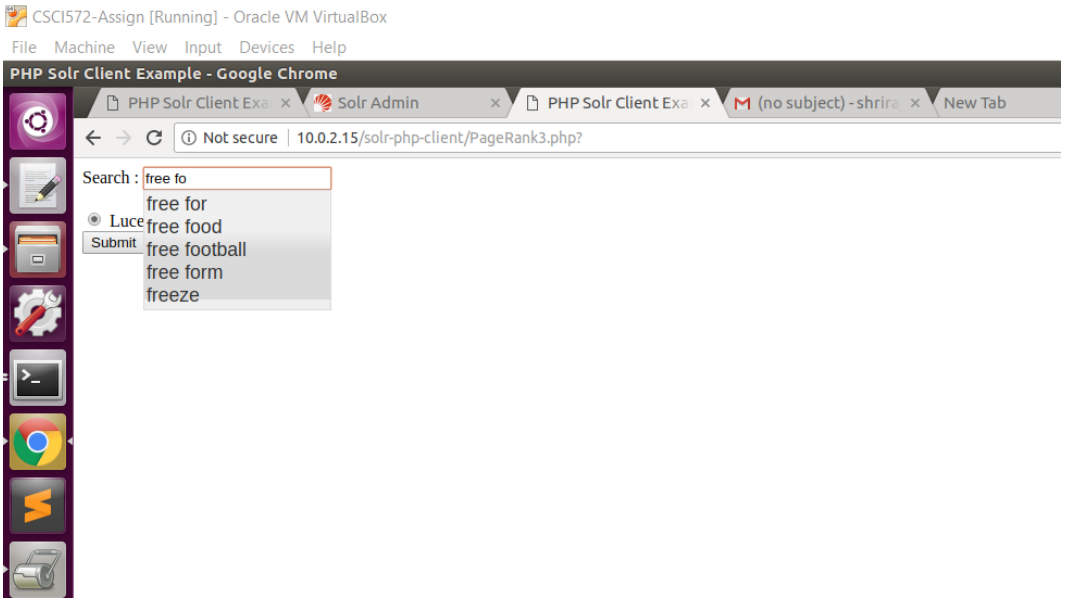
1. Bitcoin



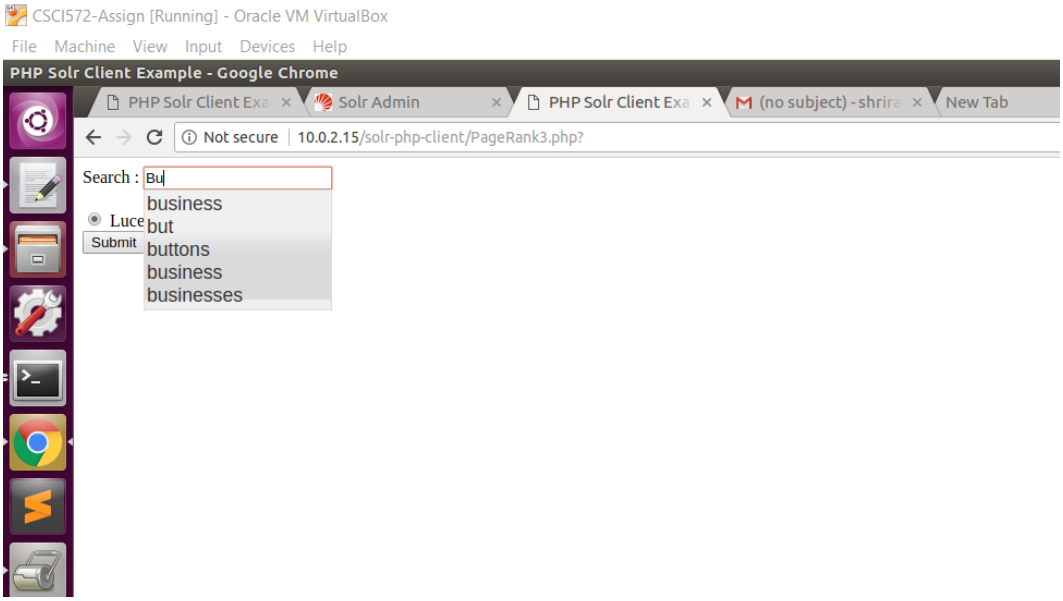
2. Learn



3.Free foo



4. Busin



5. South Ko