**Maria Divya Fernandes**

**8350120470**

**Report on Adding Spell Checking, AutoComplete and Snippets to Your Search Engine:**

**Steps in the Assignment:**

**SPELLING CORRECTION:**

1. Beautiful Soup, Html2Text and lxml was used to parse the given html files and then pass it to enchant US dictionary, a python package that helps to identify if a given word is a valid English word. I created a file parser.py to parse the Html files to generate big.txt.
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: Insert/Delete/Transposition/Substitution/Replace

b. From the words calculated, we return the most probable word from the corpusPeter Norvig’s program requires big.txt as and input, a file containing all the terms in the inverted index of the Search Engine.

1. spellrun.php is used to check the SpellCorrect program and simultaneously generate the serialized\_dictionary.txt.
2. Peter Norvig’s code has a correct() method, which takes input as wrong word and outputs the correct word.
3. Thus, before running the query, I called the function from the spell corrector and gave suggestions to the user and an option to choose both the words.

**Autocomplete:**

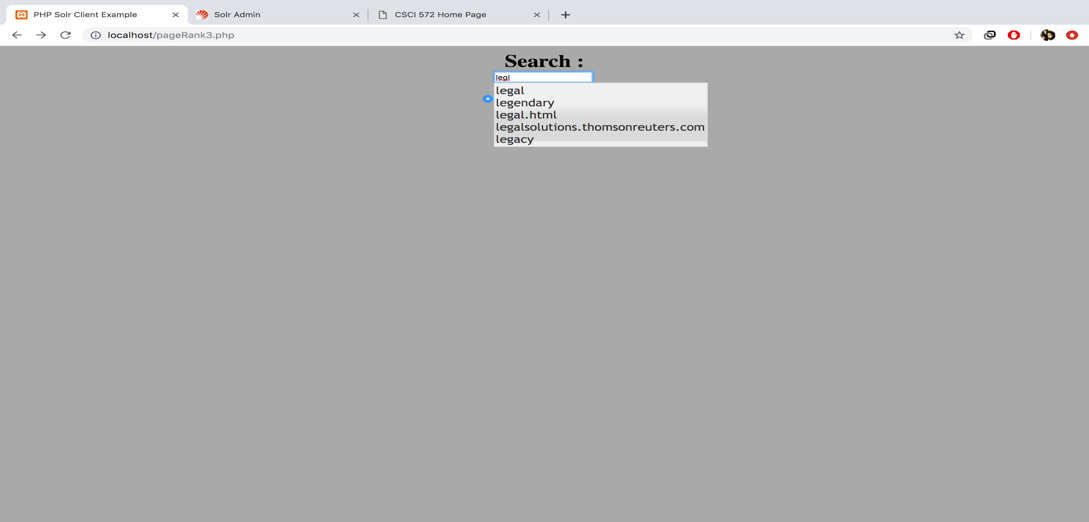
1. For Autocomplete suggestions we made changes to the solrconfig.xml by adding and a requestHandler to allow to configure default parameters for suggestion requests.Once the requestHandler is added , we need to reload the core.
2. A source containing the terms that will be provided for suggestions in the drop-down menu.
3. serialized\_dictionary.txt is generated when spellrun.php is used to check the SpellCorrect program.
4. To ensure autocomplete works for n-term queries, we split the query terms based on space. Autocomplete suggests for the latest term being entered returning an autocomplete for the last term tagged to all the initial query terms.
5. The autocomplete results are clickable, will autocomplete the word when clicked and return focus to the text box.
6. Once the user starts with a new word or a space separated word the ajax code is again activated.

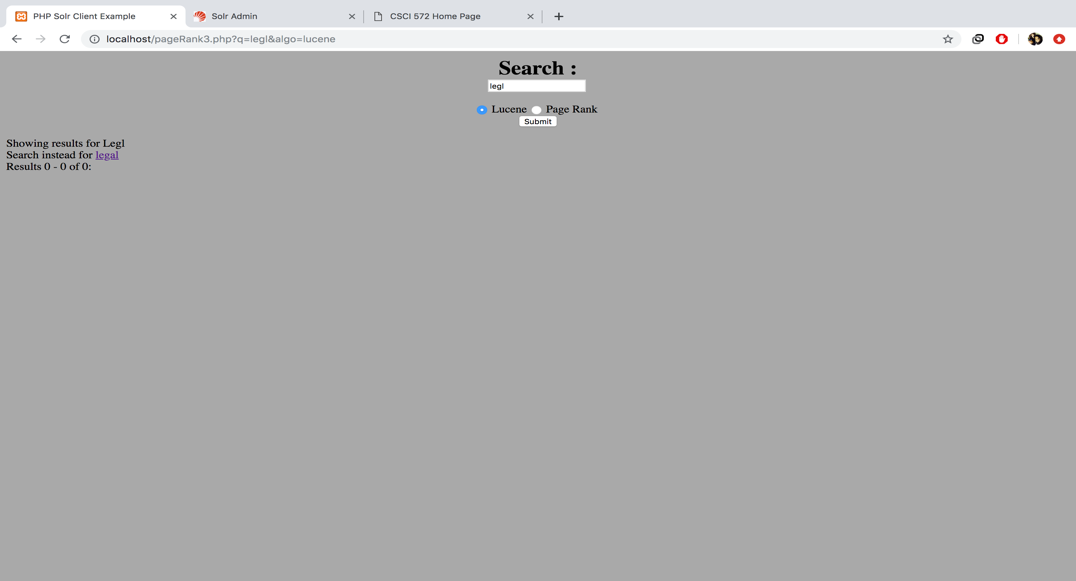
**Snippets:**

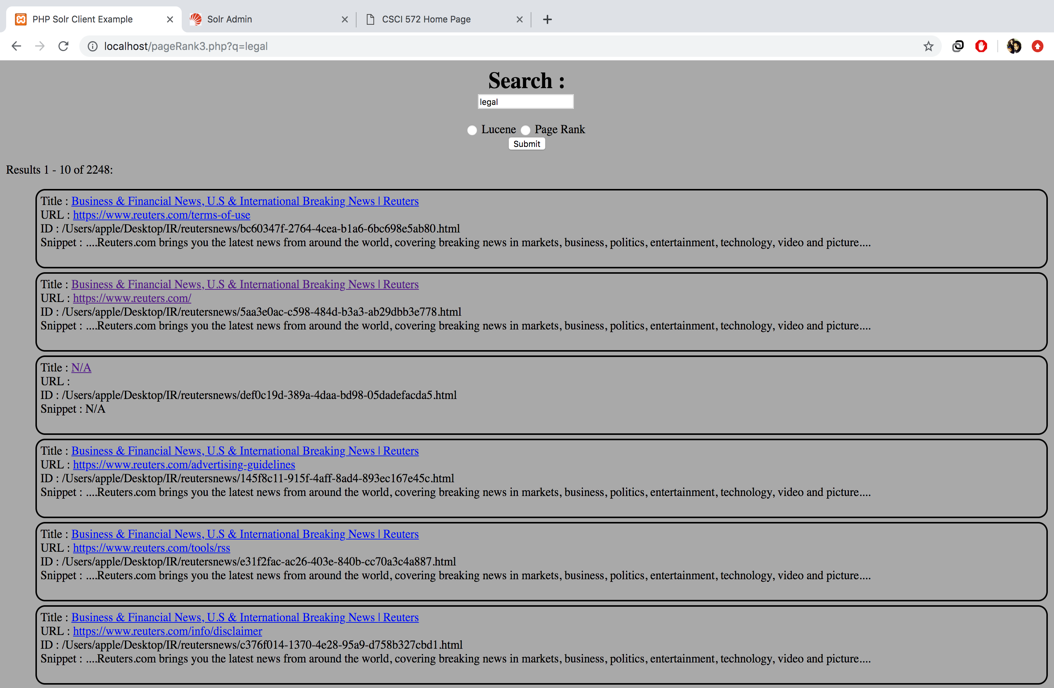
1. simple\_html\_dom parser to parse plaintext from html pages and file\_get\_contents to get the contents of the file and str\_to\_html method to convert a string.
2. The content was then split into an array & strings matched to find the index of the words in the query. Using these indices, I determined the string to be printed as the snippet. The snippet lines were chosen as follows.

For multiple term queries, try finding a sentence with all the terms together.If not, return the sentence that has all the terms in it, even if they are not together or in same order. If none above cases fail are found, return the first sentence with at least one query term in it.If no match is found, then display text

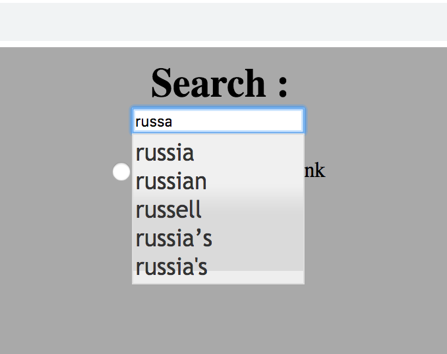
1. I selected the starting and the ending positions of the snippet lines that would be printed. In case there are no snippet found, the code would search for a good match in the description.
2. Legl ->legal

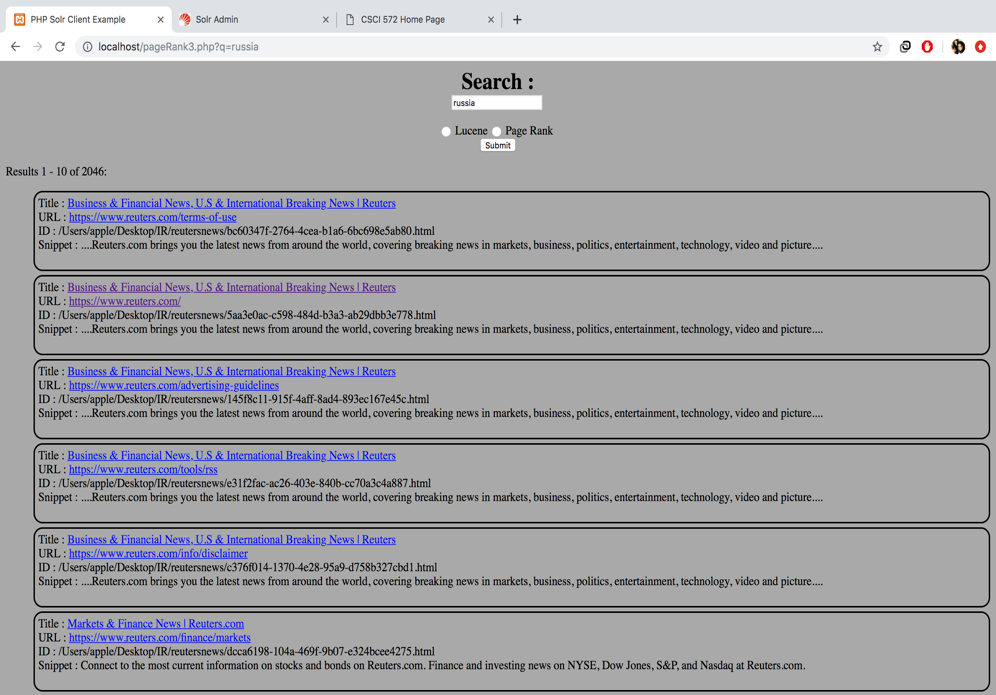




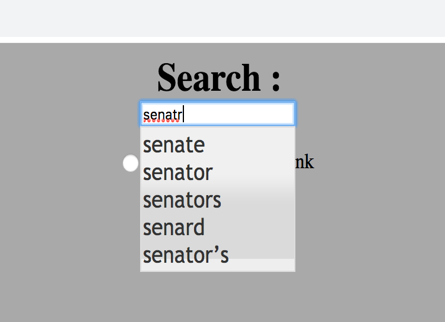


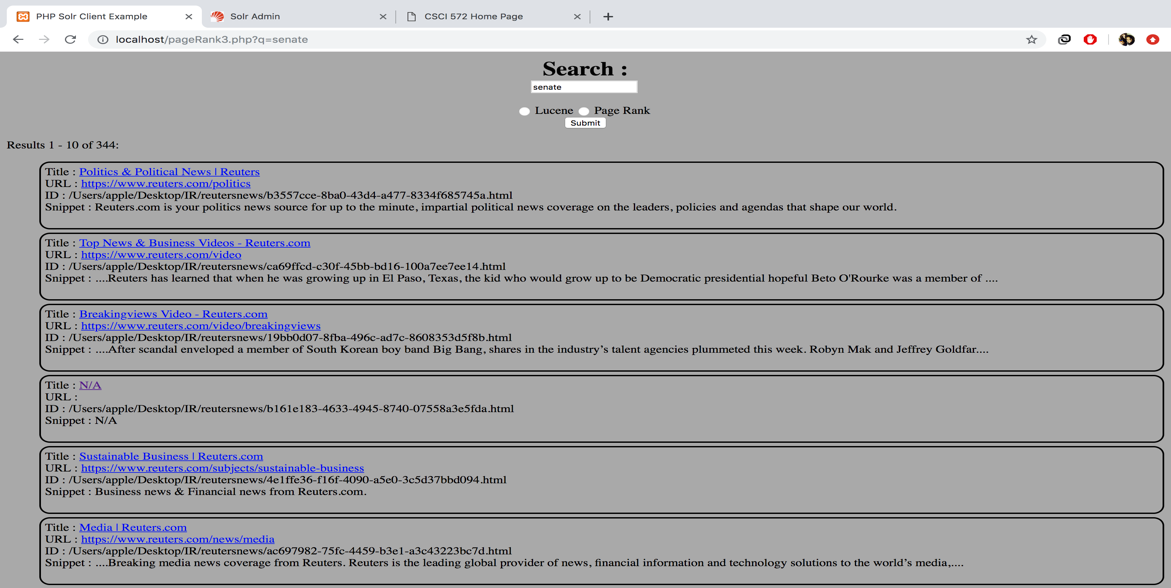
1. Russa ->Russia

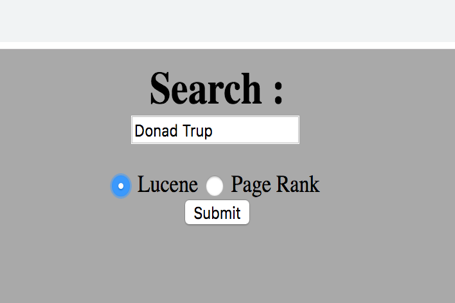


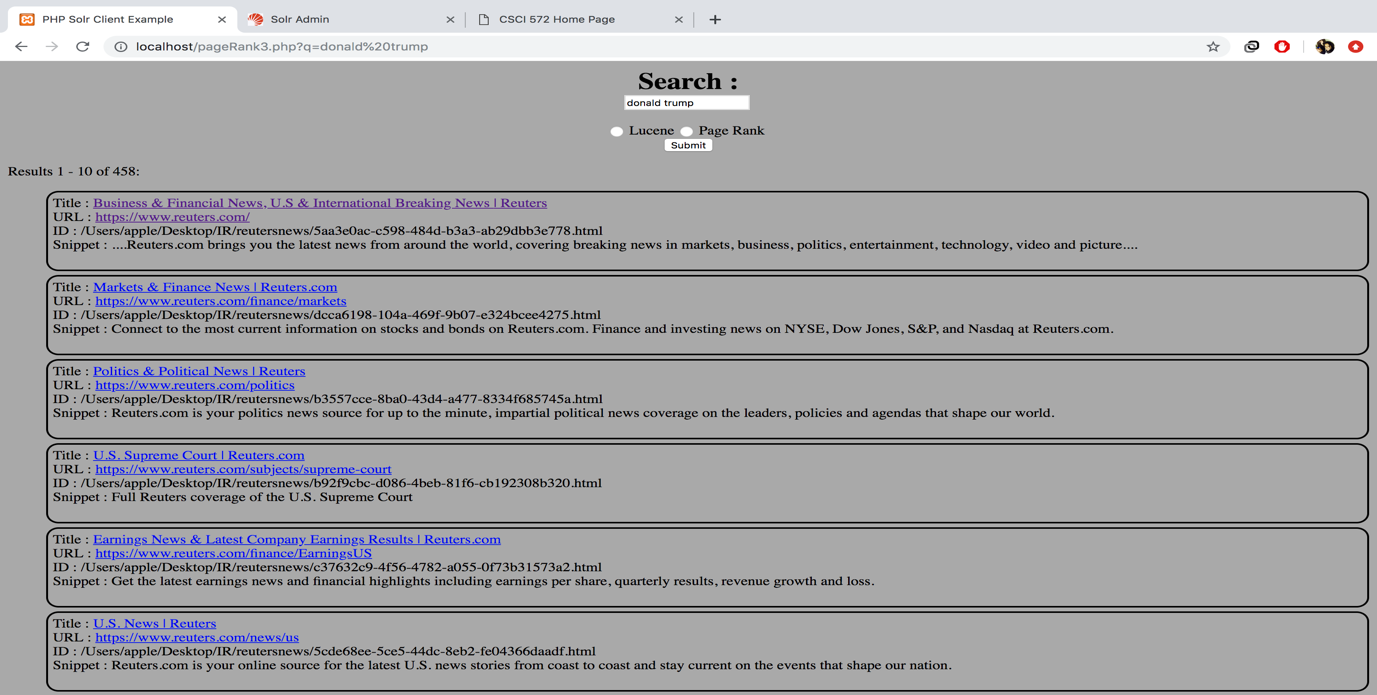
3.Senatr ->senate

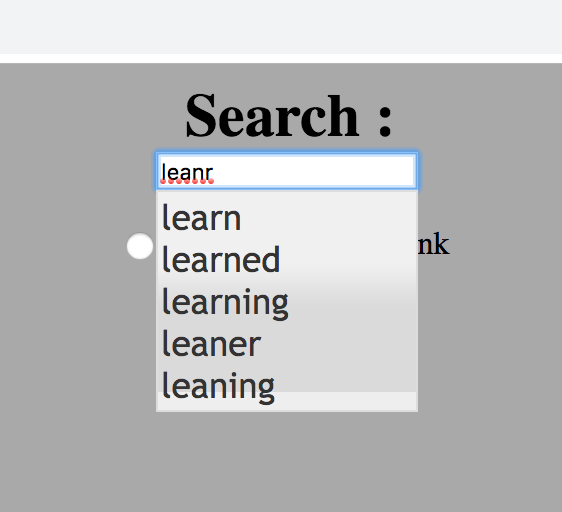
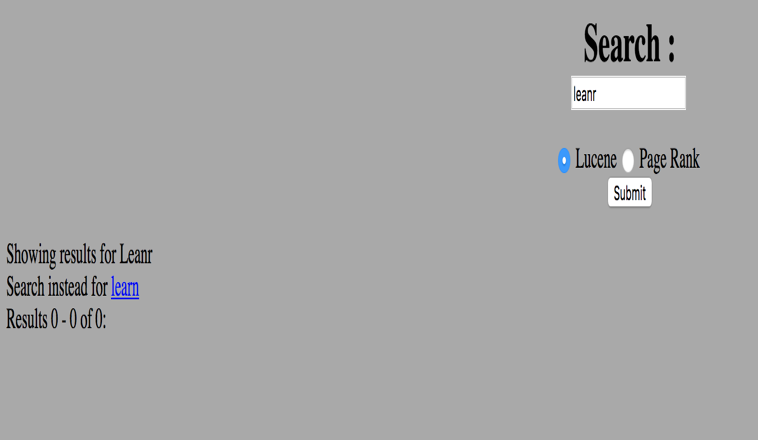


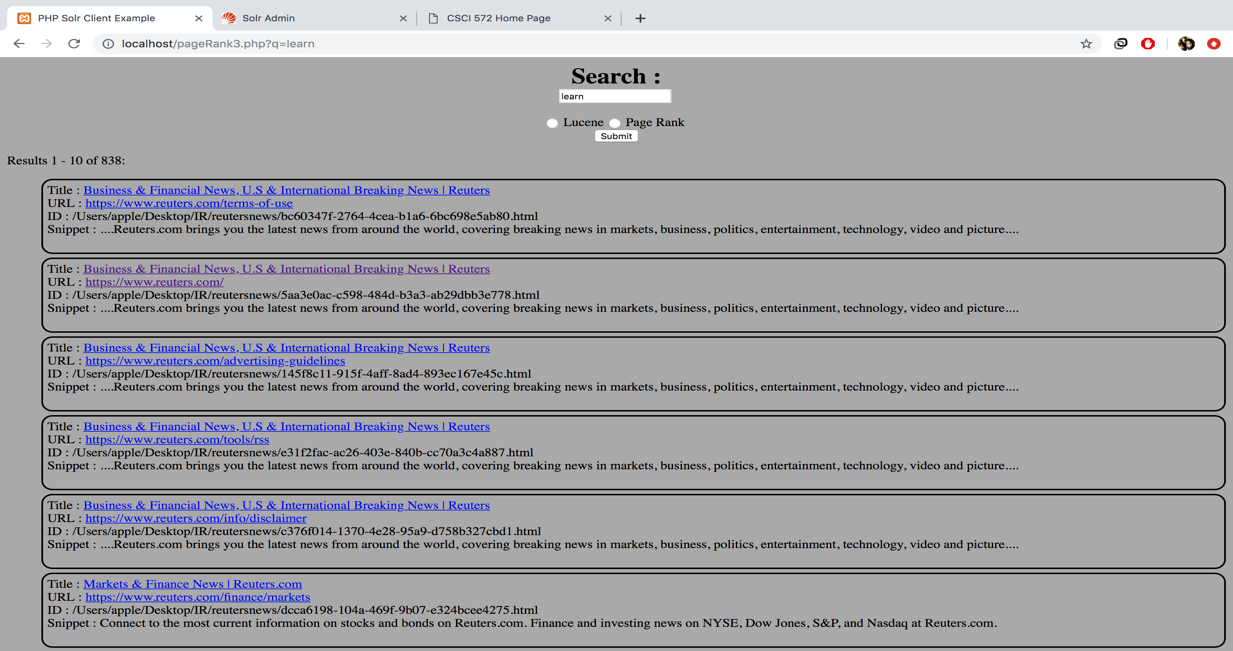
4.Donald Trup -> Donald Trump



5.leanr -> learn



Autocomplete examples

