**CSCI572 - HW5 Reports**

**Shreyansh Bhagwan Kakadiya – USC ID: 8056685392 –** [kakadiya@usc.edu](mailto:kakadiya@usc.edu)

**Steps in the Assignment:**

**SPELLING CORRECTION:**

1. To implement the spelling correction feature, I used the Peter Norvig’s Spelling corrector code in PHP to find out the candidates for the erroneous words.
2. Peter Norvig’s program requires big.txt as and input, a file containing all the terms in the inverted index of the Search Engine.
3. I used Beautiful Soup, Html2Text and lxml to parse the given html files and then passed it to enchant US dictionary, a python package that helps to identify if a given word is a valid English word.
4. I named this file parser.py (included in the zip) and parsed the Html to generate big.txt.
5. I used spellrun.php to check the SpellCorrect program and simultaneously generate the serialized\_dictionary.txt.
6. Peter Norvig’s code has a correct() method, which takes input as wrong word and outputs the correct word.
7. 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. To implement the autocomplete method, I first made the changes required in the solrconfig.xml as mentioned in the tutorials given under assignment five.
2. I used ajax xmlhttpresponse in order to communicate between my php file and the javascript.
3. The php file name getSuggestions.php (included in the zip) is called using the onkeyup event handler.
4. I then made a replica of Service.php as Service1.php to create a php client that would listen to the getSuggestions.php using the request handler “suggest”.
5. I changed the “const SEARCH\_SERVLET = suggest” so that a phpclient using the object of this apache class will use the request handler suggest instead of default search request handler.
6. I made the above return list into a php array and then echoed the JSON\_encode for the array.
7. The array was returned as a responseText to the javascript that calls the showList function which inturn attaches dynamically generates a ul list and then attaches it on the ul below the q box.
8. These li are clickable and will auto complete the word and return focus to the text box.
9. Once the user starts with a new word or a space separated word the ajax code is again activated.
10. This time the php code will only take the last word as all the previous words are already corrected.

**SNIPPET:**

1. I have used simple\_html\_dom parser to generate the plaintext out of the html pages given to us.
2. File\_get\_contents can be used to get the contents of the file.
3. I used an str\_to\_html method to convert a string.
4. The entire string was then filtered to remove special characters so that they don’t affect the matching of the keywords.
5. The string was then exploded to and array and I conducted string matching to find the index of the words in the query.
6. Using these indices, I determined the string that is to be printed as the snippet and return it to the calling function.
7. In case there are no snippet found, the code would search for a good match in title and description to serve the purpose.

**ANALYSIS OF RESULTS:**

**SPELLING CORRECTION:**

1. **brida -> bride**



1. ilegal -> illegal



1. **rusia -> Russia**



1. **nassa->nasa**



1. **Donlad trump -> Donald trump (checked in the Demo)**
2. **Illegal immigration -> illegal immigration (checked in the Demo)**

**AUTOCOMPLETE:**

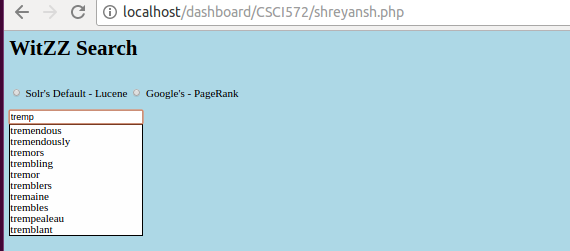
1. **nas**



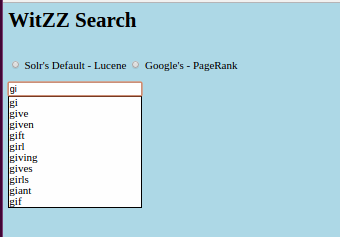
1. **brands**



1. **tremp**



1. **gi**



1. **rus**

