Homework 4
Report

1. YouTube link

https://youtu.be/E1DuemLn25Y

2. Spelling Correction

1. I have used the PHP version of the Perter Norvig Program for implementing spelling check.
2. My client script is called SearchBox.php
3. I have implemented a Java word count program which does the following:
    - Parse the list of HTML files (my dataset is CNN and USAToday)
    - Basically, I am reading the contents as text for the document title tag, anchor tag, image tag and body tag.
    - For every HTML file that I read as text, I am also creating a text file with the same name. This text file will contain the content that has been parsed the HTML file. This is needed for the snippet generation.
    - Moreover, each of these text files mentioned above will also have lines of text that are sorted in descending order of the length of the line.
    - This is needed to pick the longest matching line for displaying the snippet.
    - Therefore, one of the task/output of this program is to output a directory called "snippets" that would contain all the text files corresponding to each and every HTML file, with each text file contacting the sorted lines of text (based on their length) as mentioned earlier.
    - The next main functionality of this Java program is to tokenize the contents of each and every HTML file into words and store their counts.
    - Basically, I am using a Hash Map to implement this functionality.
    - The key of the Hash Map will be the token and the value would be the count of that token across all HTML files. So, this is a global Hash Map that has all the tokens and their counts across all the HTML files.
    - The key thing here is that I am only considering words that are more than 1 character in length (Due to the assumption that the first character is always typed correctly) and tokens beginning with a number.
    - I do not consider tokens consisting of the "/" (slash) character to avoid writing paths to the dictionary.
    - This dictionary is called as serialized_dictionary.txt and this needs to be placed in the same directory as the SearchBox.php file.

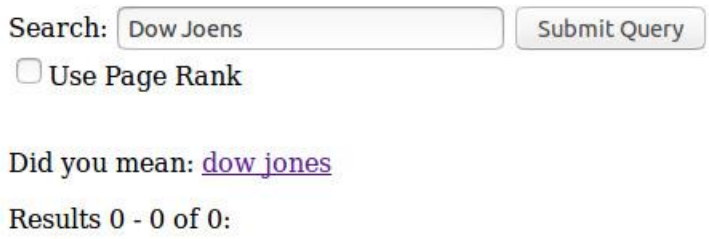
Modifications to the Peter Norvig Program:

1. I have modified the Peter Norvig program to read the serialized_dictionary.txt file and save it into a PHP dictionary. This is a one-time activity and the saved dictionary is reused for subsequent queries.

2. This is the main modification that I have done to the program and rest of the SpellCorrector.php is as before.
3. So, I use the saved dictionary containing the counts of each token to make a decision on which value to return as a “correction value” for the query. If the query keyword is directly found in the dictionary, then it is returned back to the client program.
4. Else, the Peter Norvig program computes the edit distances and finally returns the word that has the highest count value in the dictionary as the “correct value”.

Client program: Spell checking

1. My client program (SearchBox.php) calls the Peter Norvig program and uses the value returned from the Peter Norvig program as the “spell check” value.
2. If the user types a wrong keyword, the program will attempt to search for documents matching the incorrect keyword, but it would also tell the user like this:
“Did you mean:” and the word with the spelling corrected.
3. This word is a link and the user can click on it to search for the documents matching the correct word. The following screenshot shows the spell check:



2. Autocomplete

1. I added the search component into solrconfig.xml for the autocomplete.
2. The search component was defined as specified in the SpellAndAutocompleteInSolr.pdf document
3. The field used to obtain the terms for suggestion is the “_text_” which was previously indexed.
4. I then added a request handler with the url “/suggest” to solrconfig.xml for serving the suggestion requests.
5. I have used the FuzzyLookupFactory for autocomplete and the “suggest.count” has been set to 10 in the solrconfig.xml

Client Program: Autocomplete

1. I have used jQuery along with Ajax requests to implement the autocomplete functionality in the frontend.
2. I have also taken stop words from
http://algs4.cs.princeton.edu/35applications/stopwords.txt

3. The stop words in this list are compared to prevent showing them in the autocomplete suggestions.
4. The Ajax call takes the following URL: http://localhost:8983/solr/cscihwcore/suggest?q= appended with the last term that was entered by the user.
5. The suggest component receives a URL of the form: (for example) http://localhost:8983/solr/cscihwcore/suggest?q=hello&wt=json

3. Snippets

1. As mentioned earlier, my Java program creates the text file for each of the HTML files. This text file contains the lines of text sorted in descending order of their length.
2. All these text files are created in a folder called "snippets". This folder must be placed at the same directory as the client program. (SearchBox.php)

Client Program: Snippets

1. In the client program, for every document that is fetched for a particular query, I take the document id (only the last portion of the path that contains the base name without the file extension).
2. I then check if a file with this name is available in the snippets folder. I open the matching text file and read every line until I find the first line where the query term occurs.
3. Since the lines are arranged in descending order, the first match is the longest sentence containing the query term.
4. I then compute the position of the query keyword in the line and return part of the string between the positions index–100 and index+100 as the snippet

Analysis of results:

1. Spell check:

These are the 5 examples of spell check that are handled correctly by the program:

Query: Dow Joens
Corrected value: dow jones

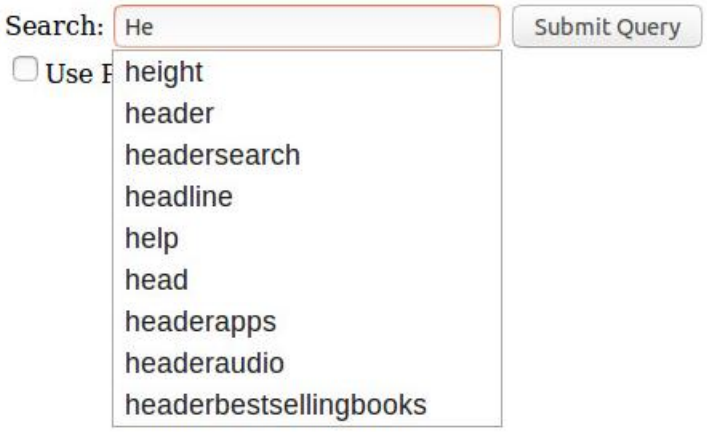
Query: Rio Olmypics
Corrected value: rio olympics

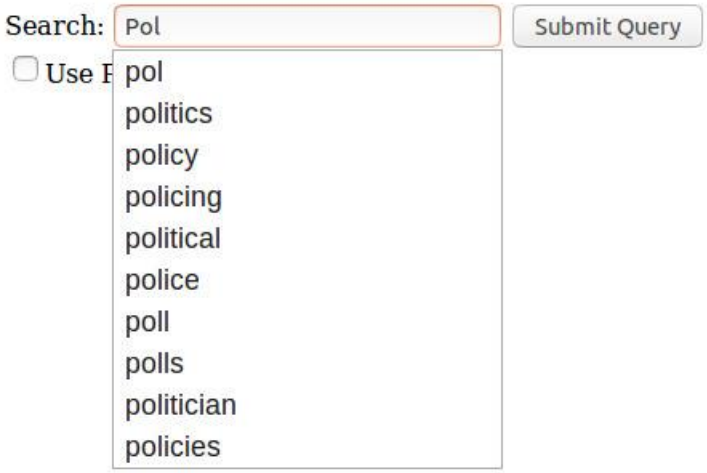
Query: Poekmon Go
Corrected value: pokemon go

Query: Rio Doanld Trump
Corrected value: donald trump

Query: Brzail
Corrected value: brazil

2. Auto completion:





Assumptions for generating the video

1. In the test case to interchange the 3rd and 4th characters, I have considered the longest word in a query and then interchanges the 3rd and 4th characters. For example, in “Rio Olympics”, I have considered “Olympics” to interchange the 3rd and 4th characters. So, the query becomes “Rio Olyimpcs”.
2. Similarly for the test case to rotate the 4th, 5th and 6th letters, I have considered the longest word in the query as above to rotate the letters. So, for example, “Pokemon Go” becomes “Pokmeno Go”.
3. For shorter words, I have just taken the last 3 characters and rotated them. So, “NATO” becomes “NOAT”.