Enhancing the Solr Search Engine

By Akshay Bhatia; email: akshaybh@usc.edu ; USC ID: 4983495776

Steps followed to complete this assignment

I completed the assignment on Ubuntu as follows (steps 1-7 are a summary of hw4):

1. Installed **Solr** on Ubuntu
2. Created a new core, “myexample” in Solr for the assignment
3. Used **SimplePostTool** to create Solr’s index using the HTML files provided
4. Wrote a Java program on Eclipse to create an edge list using the **JSoup** library
5. Wrote a Python program to calculate PageRank using the **NetworkX** library, using the output of the previous step as input.
6. Configured Solr to allow sorting on the basis of **PageRank** by editing the managed-schema and solrconfig.xml files as specified.
7. Created a **PHP** program (`index.php`) in VS Code as follows:
    a. Used **solr-php-client** to allow accessing Solr and querying it.
    b. Create an associative array mapping file names to URLs, to fetch URLs from in case they weren’t present in Solr output.
    c. Wrote PHP code to specify the sorting algorithms, giving a choice between Lucene (score desc) and PageRank (pageRankFile desc)
    d. Used **Bootstrap 4** to style the page to look similar to how Google displays search results.
8. Configured Solr to make suggestions using its **FuzzyLookupFactory** as instructed to us.
9. Used the library **jQuery-autocomplete** by DevBridge to add autocomplete to the input text box.
    a. Wrote a custom PHP backend service (`suggest.php`) to generate a list of suggestions by applying the autocompletion per word, to work with the above library.
10. Implemented the recommended **PHP Norvig’s Spell Corrector (**`SpellCorrector.php`**)**:
    a. Wrote a Java program using **Apache Tika** to parse HTML to generate the required **`big.txt`.**
    b. Implemented spell check support into the autocomplete backend service (`spellcheck.php`)
    c. Like Google, displayed the spellcheck as the first suggestion in the list of suggestion, without repeating it in the autocomplete suggestions
    d. Like Google, displayed the **“showing results for…”** dialog to automatically search for the spelling-corrected results by default.
11. Implemented **snippets** in PHP (`snippets.php`):
    a. Used **`HTML2Text`**`.php`, a PHP library to parse extract content from HTML pages
    b. Used **RegEx** to split generated text into sentence-like blocks.
    c. Selected sentences (from the description if possible, if not from the text) based on the rules specified and shortened them to 160 characters.
    d. Highlighted the query keywords by wrapping them in strong tags.

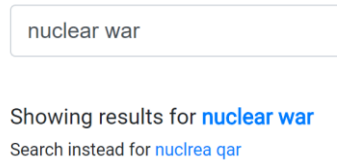
12. Used **Apache** to host a web server on Ubuntu for the created PHP file.

Analysis of the results

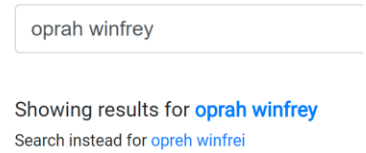
Spelling correction

The application handles misspellings just like Google does – while typing a misspelled query, the first suggestion is the correctly spelled query, and if the user proceeds to search with it anyway, it uses the correctly spelled query to search by default (but still allows the user to go with the misspelling)

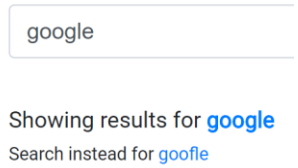
1. *nuclrea qar*



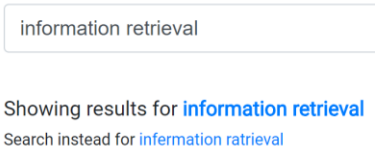
2. *opreh winfrei*



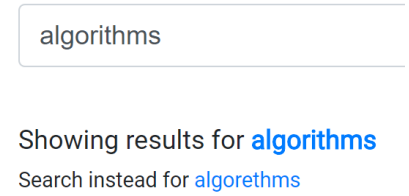
3. *goofle*



4. *infermation ratrieval*



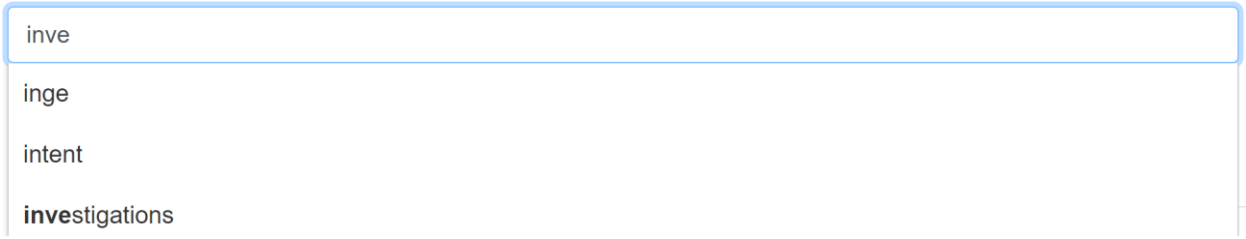
5. *algorethms*



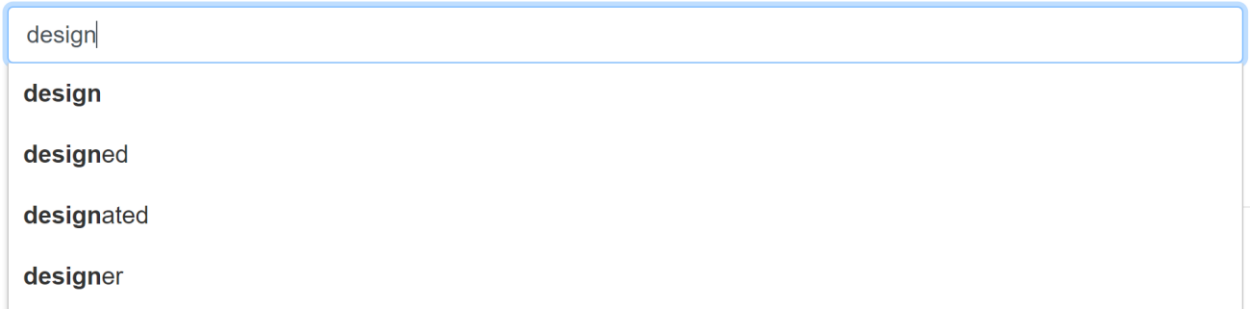
Autocomplete

Using Solr's FuzzyLookupFactory in a custom suggest component, Autocomplete populates a dropdown generated by the jQuery-autocomplete library (by DevBridge). It takes care not to repeat the spelling suggestion.

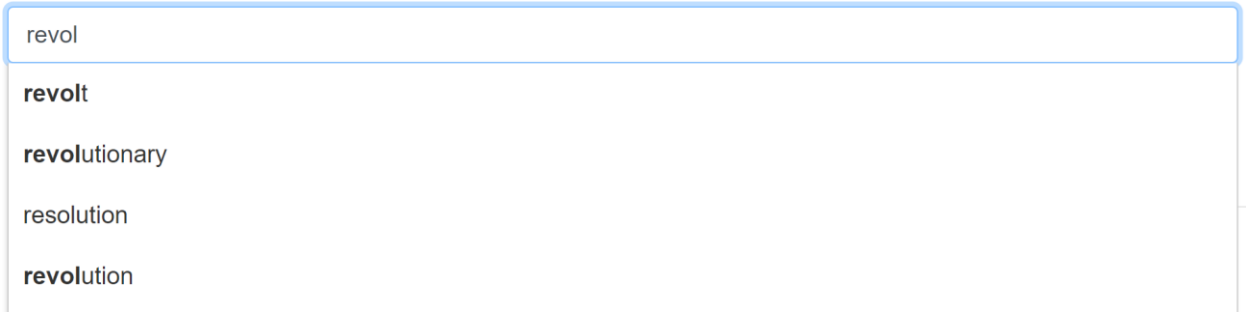
1. *Inve-*



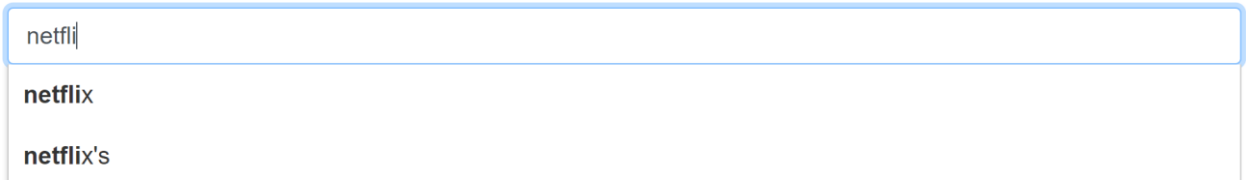
2. *Design-*



3. *Revol-*



4. *Netfli-*



5. *Histor-*