Homework 5:

Adding Spell Checking, AutoComplete and Snippets to Your Search Engine

Objectives

- Experience using a third party spell program
- Developing efficient methods for accomplishing autocomplete
- Implementing a simple snippet feature for search results

In the previous document (SpellAndAutocompleteInSolr.pdf) you saw how to enhance the Solr program with spelling correction and an autocomplete (suggest) function. **In this exercise you are asked to use an external spelling correction program in conjunction with Solr and to enhance the autocomplete functionality of Solr**. In addition, the search results you return should include, in addition to a link to the resulting web page, a snippet that includes one or more of the query keywords. In the case of spelling correction you may use an existing third-party program adapted to your downloaded files. In the case of autocomplete you will need to enhance your client program that communicates with Solr to deliver autocomplete suggestions to the web interface you created in an earlier homework. In the case of snippets, you will need to locate a string in the web page that includes the query terms and output that sentence along with the link results.

Description of the Exercise

Spelling Correction: in the class lecture you saw a complete spelling correction program developed by Peter Norvig. The program was written in Python. For this exercise you are welcome to use whatever third party spelling program you wish, or you may even write your own. Since most of you wrote your previous homework client using PhP, you may want to adopt a version of Norvig's spelling program written in PhP and run it on your server. You can download the PhP version of Norvig's spelling corrector from here:

http://www.phpclasses.org/package/4859-PHP-Suggest-corrected-spelling-text-in-pure-PHP.html#download

(you will have to register at the site before being able to download the software, registration is free)

If you prefer to use Norvig's program in a different language, a wide variety of implementations can be found at the bottom of this page, http://norvig.com/spell-correct.html

You should make sure to enhance your spelling correction program with a set of terms that are specific to the news website that you are responsible for. You should make sure that common terms such as *politics, election, etc*, and the terms used in the queries of homework #4 are handled. Norvig's spell correction program uses a text file(''big.txt") to get set of words to calculate edit distance. For this, you can use any parser (our suggestion - Apache Tika) and

create your own big.txt instead of the one in website. Instructions on using apache Tika for this purpose can be found here.

Autocomplete: for the autocomplete portion of the exercise, you will have to modify your client program so it accepts single character insertions to the text box, and returns a list of completions/suggestions.

There are several ways to implement the autocomplete functionality while using Solr. One possible way is to use the FuzzyLookupFactory feature of Solr/Lucene. The FuzzyLookupFactory creates suggestions for misspelled words in fields. It assumes that what you're sending as the suggest.query parameter is the beginning of the suggestion. It will match terms in your index starting with the provided characters. So if the query is "ca" it will return all the words starting with "ca", e.g. "california" and "carolina" etc. For the first character that is entered, 5 – 10 autocomplete suggestions should appear. For the second character that is entered, 3 – 7 autocomplete suggestions should appear.

For this to work you need to enable the suggest component as described in the tutorial but add some options.

Note: with respect to specific issues about how spelling corrections are displayed or how autocomplete corrections are displayed you should imitate the way Google handles both. For example, while typing in the search box, the top suggestions should automatically appear and be updated as the user keeps typing. The spellcheck suggestion should appear at the top of the retrieved results. If the word typed is correct no suggestion should appear at the top.

Snippets: To produce a snippet for each of the returned search results you can do the following: for each search result you should open the web page that is referred to. You should then look for a string match of the query terms with the web page. Return the first sentence that provides a match. If no match is found, then no snippet is returned.

Demonstrating Your Solution

There will be an in-class demonstration where all students will demonstration their implementation. The demonstration will be on the last day of class. Graders will run a script designed to test your implementation. It will include a set of queries so we can see your autocomplete functionality. It will contain a set of misspellings so we can examine how well your spell correction program performs. And we will be looking at the snippets your program produces for the results returned from the queries.

Submission Instructions

There should be a report describing what you have done. This report should include:

1. Steps you followed to complete this assignment. Include the details of what tools and techniques you used to implement spelling correction and autocomplete.

2. Analysis of the results: In this you should provide FIVE examples of misspelled terms that are correctly handled by your spelling correction program. You should also provide FIVE examples of auto-completion.

3. Using the submit command you should provide a single .zip (CSCI572_HW5.zip) file which contain the following files

- your report (no more than 5 pages)
- the external spelling correction program that was used
- all source code that you wrote, most especially the code implementing the autocomplete functionality.

You are required to submit your results electronically to the csci572 account on SCF. Though there will be an actual in-class demonstration, you must also submit the above files electronically, by entering the following command from your Unix prompt:

submit -user csci572 -tag hw5 CSCI572_HW5.zip