Spell Checking and Auto Complete Feature for the Search Engine developed using Solr

Assignment: to provide spelling correction and auto completion functionalities to the search engine that was developed in the previous assignment.

Steps followed:

- ❖ Tool: Solr 5.3.1
- ❖ School Indexed: USC School of Cinematic Arts
- ❖ Root URL: http://cinema.usc.edu
- ❖ Type of documents Crawled: HTML, PDF, DOC
- ❖ Spelling Correction Algorithm: PHP version of Peter Norvig's Spell Corrector
- ❖ Auto Complete: FuzzyLookupFactory feature of Solr/Lucene

1. Installation and indexing the data

- Solr 5.3.1 was installed according to the instructions provided in the previous assignment
- Apache server was hosted using XAMPP
- A core was created to store all the indexed results

  ```
  – bin/solr create –c myexample
  ```
- Once the core was created, the file ***managed-schema*** was changed to ***schema.xml*** and the necessary changes were made to the file as indicated in the previous assignment
- Indexing of the crawled data was done as follows:

  ```
  bin/post –c myexample <path to the folder>
  ```
- We configure Solr to default querying from the field text by defining the default field in the request handler in solrconfig.xml file.

2. Adding a suggest component

- A search component was added to solrconfig.xml and it uses the Suggest Component. It is done as follows:

  ```
  <searchComponent class='solr.SuggestComponent' name="suggest"> <lst
  name="suggester">
  <str name="name">suggest</str>  <str name="lookupImpl">fuzzyLookupFactory</str>
  <str name="field">_text_</str>
  </lst>
  </searchComponent>
  ```
- The "_text_" field is used to obtain terms for suggestion
- A requestor was added as follows:

  ```
  <requestHandler class='solr.SearchHandler' name="/suggest"> <lst name="defaults">
  <str name="suggest">true</str>
  ```

```
<str name=”suggest.count”>5</str>
<str name=”suggest.dictionary”>suggest</str>
</lst>
<arr name=”components”>
<str>suggest</str>
</arr> </requestHandler>
```

- Solr is configured to return upto 5 auto complete suggestions.

3. Autosuggest Feature

- The user enters the query in the search box and the key strokes are monitored by the keyup component of the jQuery
- When the keyup event occurs, an AJAX request is sent to autofill.php with the query string as the parameter
- Autofill.php is used to send the query to SpellCorrector.php and the AutoComplete feature of Solr
- The returned results are then filtered into JSON format and this formatted result is returned to index.php to be displayed

4. Spell Corrector Algorithm by Peter Norvig

- Given a word, Norvig’s algorithm tries to find the most likely spelling correction for that word. Out of all the possible corrections, we try to find a correction c, that maximizes the probability of c, given a word w. We use Baye’s Theorem to compute the probability (argmax P(c|w))
- Norvig’s algorithm also uses the Edit distance between two words to suggest the correction c
- The spelling corrector model has to be trained specifically for each school that is being indexed
- The big.txt file contains a collection of words. This data is used to train the spelling corrector.
- Python’s beautifulSoup package is used to extract the data from the crawled files.
- Serialized_dictionary.txt is generated when the spelling corrector algorithm is run for the first time. It caches the training data and uses it during the later executions of the algorithm.
- Spelling correction is done for each word the user enters in the query box.

5. Stop Word Removal

- Solr has inbuilt support for stop words removal. Solr by default is configured to remove

stop words
- Stop words can be added to stopwords.txt which is present under conf folder

6. Stemming

- Solr's support for stemming can be enabled by adding the following line to schema.xml <filter class =”solr.SnowballPorterFilterFactory”/>

- **Analysis of the results:**

Spelling Correction:

When a user enters “productios”, the auto correct feature suggests faculties

productios -> production

We might think that ‘productions’ also could be the correct word where we missed entering an ‘n’, but Norvig's Algorithm suggests “production”. This is because, an addition of one letter is needed to change productios to productions whereas as it requires a deletion and an addition (it can also be viewed as one substitution and an addition) to change productios to productions. Also, since the algorithm considers Baye's Theorem, we have to compute the error model P(w|c).

Norvig's Algorithm uses a trivial model which states:

- All known words of edit distance 1 are more probable than known words of edit distance.
- A known word is a word that has been seen in the language model training data.”

Considering these factors, Norvig's Algorithm ranked prodcution higher than productions.

The spelling corrector comes up with the correct spelling as the model is trained specifically from the data crawled from USC School of Cinematic Arts.

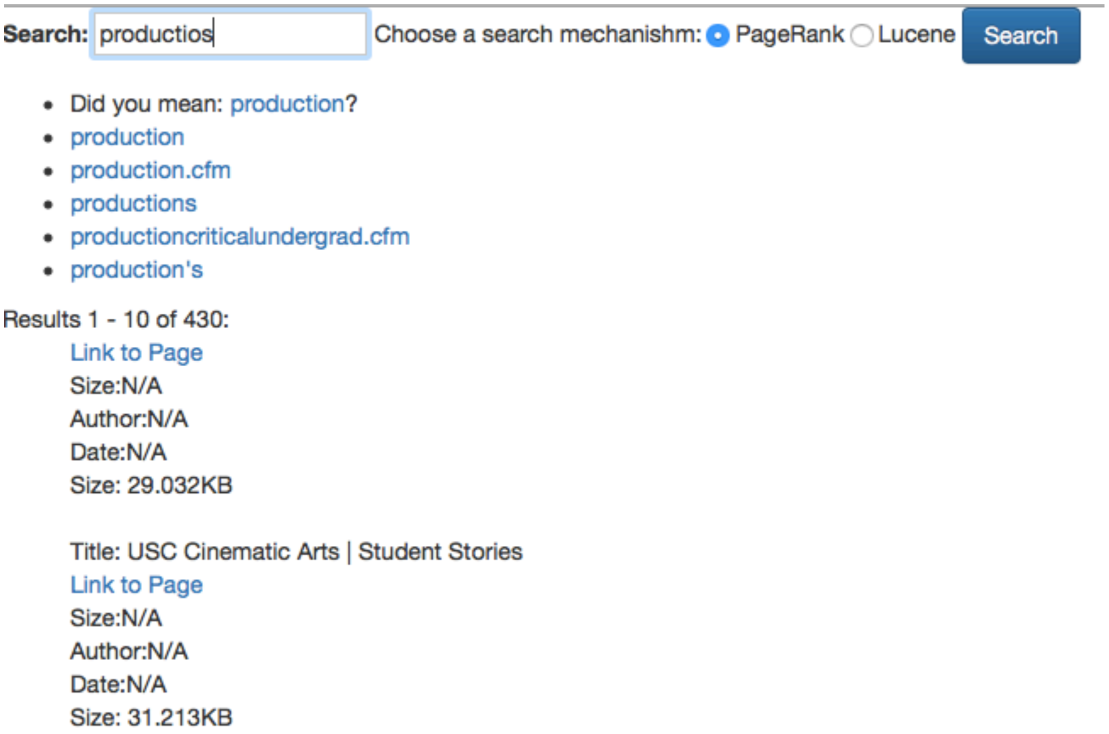
Auto Complete:

production -> production, production.cfm, productions, production's, productioncriticalundergrad.cfm

cinematic -> cinematic, cinematicarts.cfm, cinematics, cinemagic, cinematic

Screenshots

Fig 1. Spelling Correction Example



Search: productios Choose a search mechanishm: ● PageRank ○ Lucene

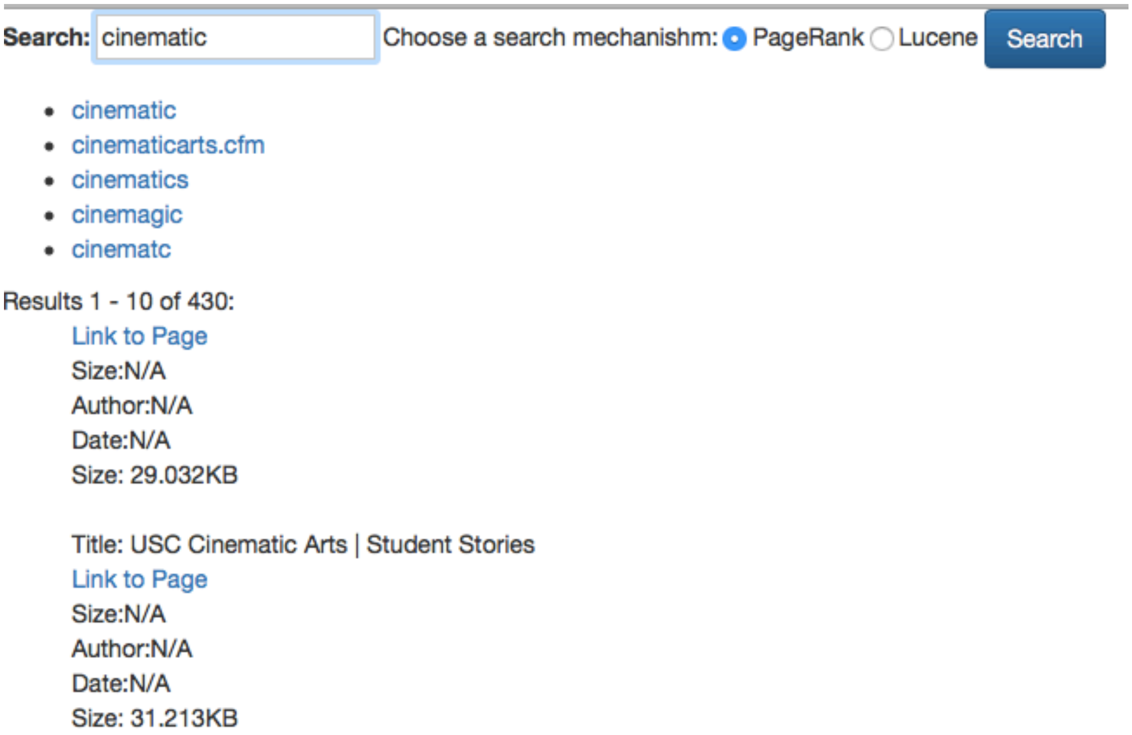
- Did you mean: production?
- production
- production.cfm
- productions
- productioncriticalundergrad.cfm
- production's

Results 1 - 10 of 430:

Link to Page
Size:N/A
Author:N/A
Date:N/A
Size: 29.032KB

Title: USC Cinematic Arts | Student Stories
Link to Page
Size:N/A
Author:N/A
Date:N/A
Size: 31.213KB

Fig 2. Autocomplete Example



Search: cinematic Choose a search mechanishm: ● PageRank ○ Lucene

- cinematic
- cinematicarts.cfm
- cinematics
- cinemagic
- cinematc

Results 1 - 10 of 430:

Link to Page
Size:N/A
Author:N/A
Date:N/A
Size: 29.032KB

Title: USC Cinematic Arts | Student Stories
Link to Page
Size:N/A
Author:N/A
Date:N/A
Size: 31.213KB