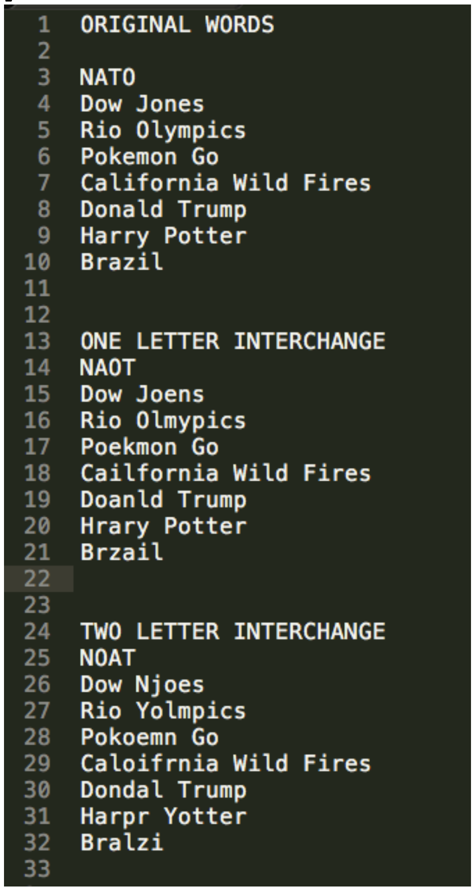
CSCI 572: Information Retrieval and Web Search Engines

Assignment No. 4
Enhancing Your Search Engine

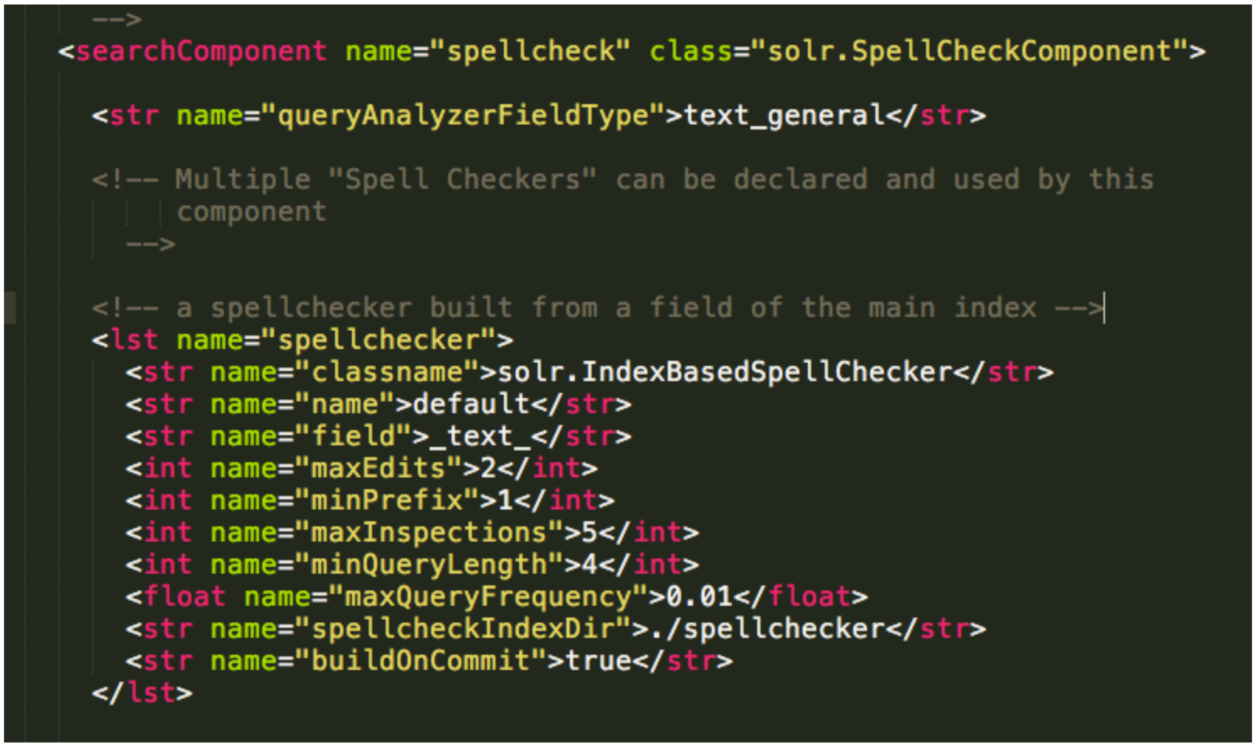
YOUTUBE URL: https://www.youtube.com/watch?v=n8c0_hR5zV8
[NOTE: For this video, I'm attaching a screenshot. This screenshot consists of the misspelled words as described in the grading guidelines on
http://www-scf.usc.edu/~csci572/2016Fall/hw4/TestScript.pdf
The tests being
3. TESTING SPELL CORRECTION - One letter interchange For each of the eight queries, enter the entire query but with the third and fourth letters interchanged. - Two letter interchange For each of the eight queries, enter the entire query but take the fourth, fifth, and sixth letters and rotate them so the sixth letter becomes the fourth, the fourth letter becomes the fifth, and the fifth letter becomes the sixth.
This was done to speed up the process of typing the queries and to reduce the overall time of the video to be uploaded.
]



```
ORIGINAL WORDS

NATO
Dow Jones
Rio Olympics
Pokemon Go
California Wild Fires
Donald Trump
Harry Potter
Brazil


ONE LETTER INTERCHANGE
NAOT
Dow Joens
Rio Olmypics
Poekmon Go
Cailfornia Wild Fires
Doanld Trump
Hrary Potter
Brzail


TWO LETTER INTERCHANGE
NOAT
Dow Njoes
Rio Yolmpics
Pokoemn Go
Caloifrnia Wild Fires
Dondal Trump
Harpr Yotter
Bralzi

```

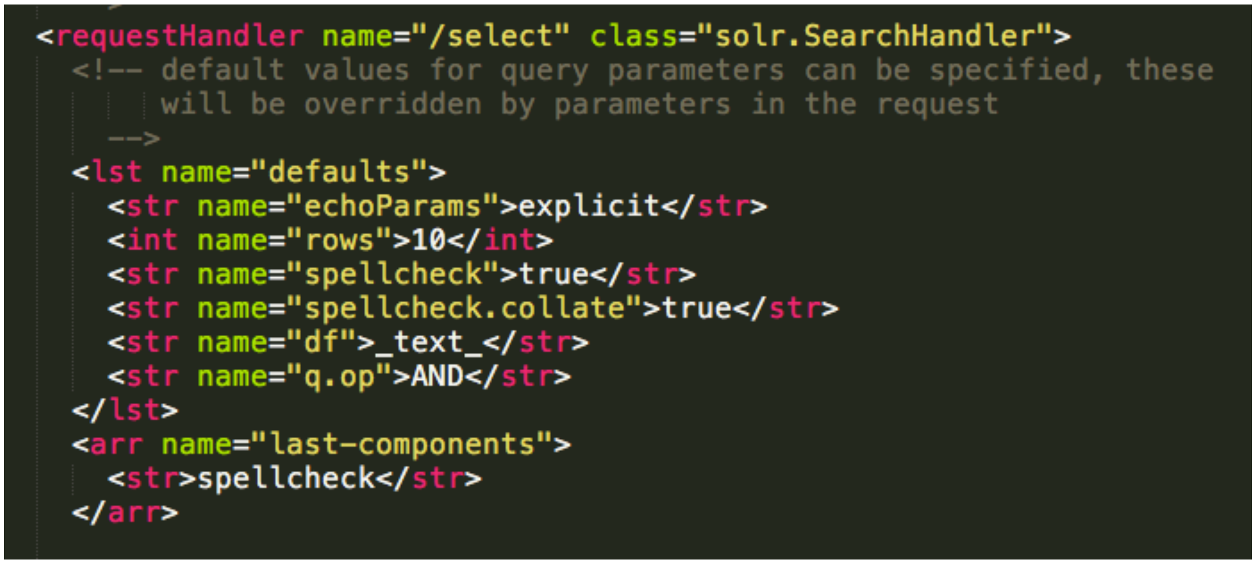
Steps Followed for This Assignment:

- This assignment is a continuation of HW3 (Indexing The Web Using Solr).
- First the instructions on http://www-scf.usc.edu/~csci572/2016Fall/hw4/SpellAndAutocompleteInSolr.pdf were followed to include the spellcheck and suggest components in Solr.
- SPELLCHECK COMPONENT: for this we included the search component in the solrconfig.xml file as follows



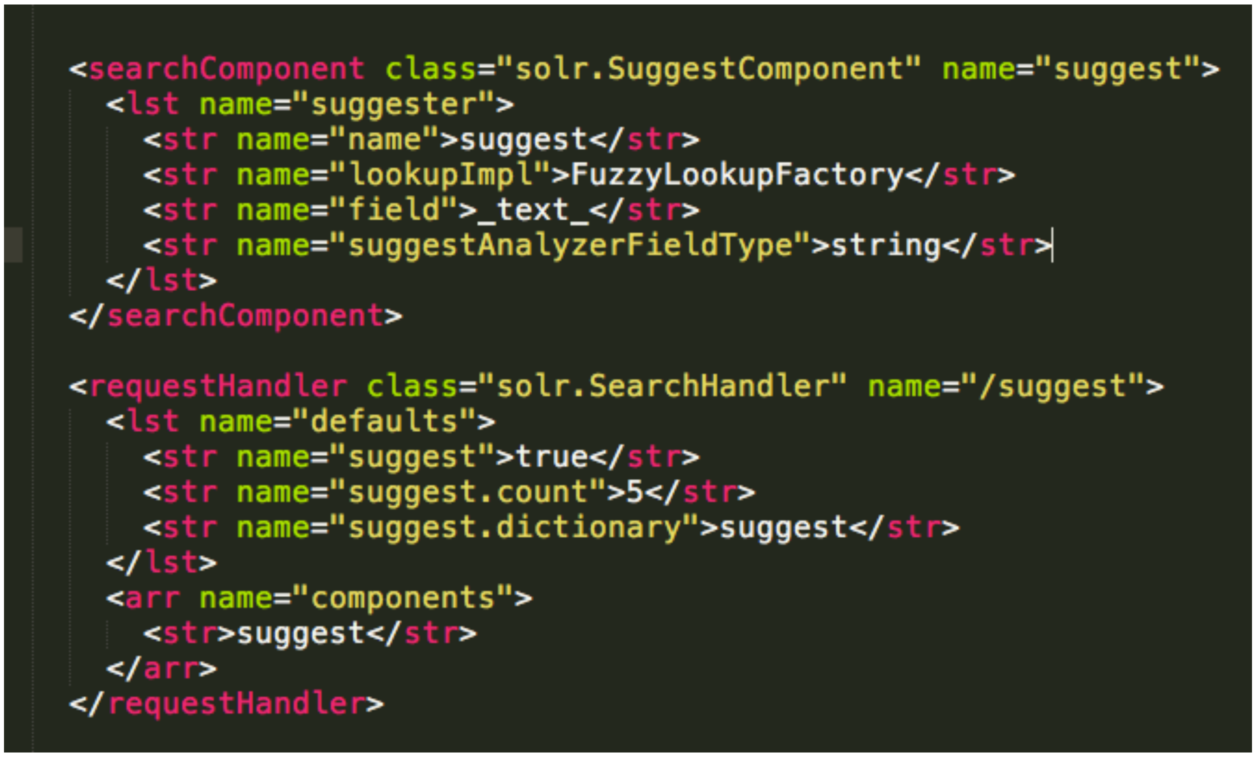
```
  -->
<searchComponent name="spellcheck" class="solr.SpellCheckComponent">

  <str name="queryAnalyzerFieldType">text_general</str>

  <!-- Multiple "Spell Checkers" can be declared and used by this
       component
    -->

  <!-- a spellchecker built from a field of the main index -->
  <lst name="spellchecker">
    <str name="classname">solr.IndexBasedSpellChecker</str>
    <str name="name">default</str>
    <str name="field">_text_</str>
    <int name="maxEdits">2</int>
    <int name="minPrefix">1</int>
    <int name="maxInspections">5</int>
    <int name="minQueryLength">4</int>
    <float name="maxQueryFrequency">0.01</float>
    <str name="spellcheckIndexDir">./spellchecker</str>
    <str name="buildOnCommit">true</str>
  </lst>
```

now to bind this spell checker with the select request Handler we added the following



```
<requestHandler name="/select" class="solr.SearchHandler">
  <!-- default values for query parameters can be specified, these
       will be overridden by parameters in the request
    -->
  <lst name="defaults">
    <str name="echoParams">explicit</str>
    <int name="rows">10</int>
    <str name="spellcheck">true</str>
    <str name="spellcheck.collate">true</str>
    <str name="df">_text_</str>
    <str name="q.op">AND</str>
  </lst>
  <arr name="last-components">
    <str>spellcheck</str>
  </arr>
```

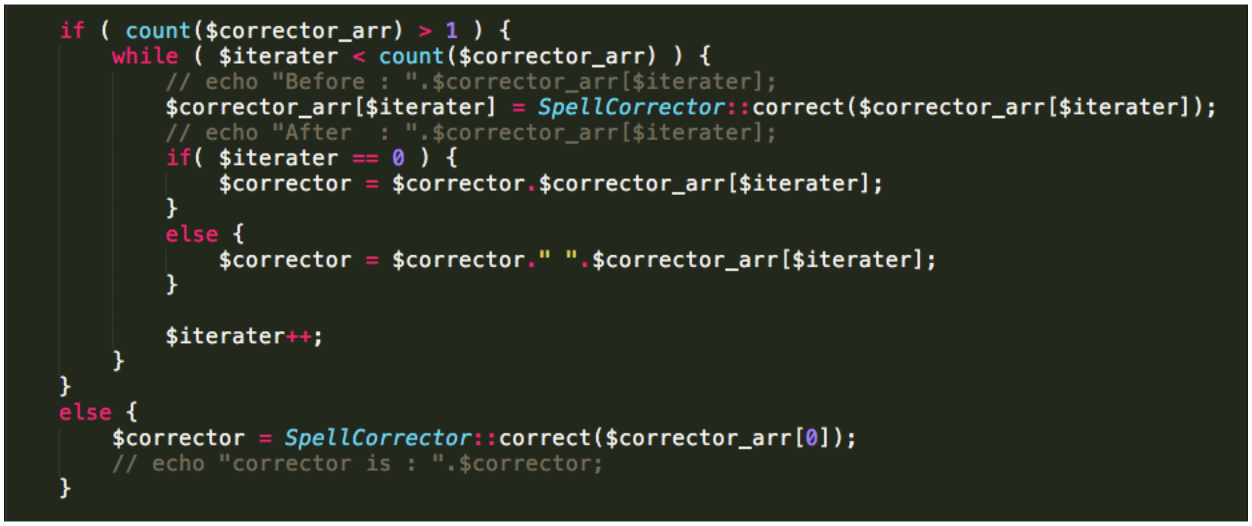
- SUGGEST COMPONENT: just like Spellcheck component a suggest component will have to be included and the request handler. This was done as follows



```
<searchComponent class="solr.SuggestComponent" name="suggest">
  <lst name="suggester">
    <str name="name">suggest</str>
    <str name="lookupImpl">FuzzyLookupFactory</str>
    <str name="field">_text_</str>
    <str name="suggestAnalyzerFieldType">string</str>
  </lst>
</searchComponent>

<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>
```

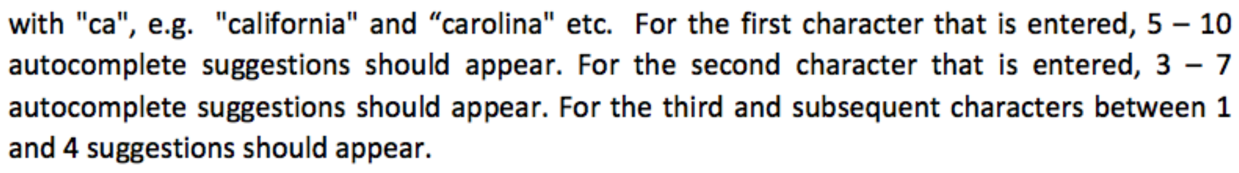
- PHP CODE:
  The spellcheck and auto suggest components now have to be integrated in the php code. For spelling correct the Norvig's Spelling Corrector program was used from http://www.phpclasses.org/package/4859-PHP-Suggest-corrected-spelling-text-in-pure-PHP.html#download
[NOTE: a separate login was also create for PHPclasses to download the php version of Norvig's algorithm]
This php file is saved as SpellCorrector.php

PHP
SpellCorre
ctor.php



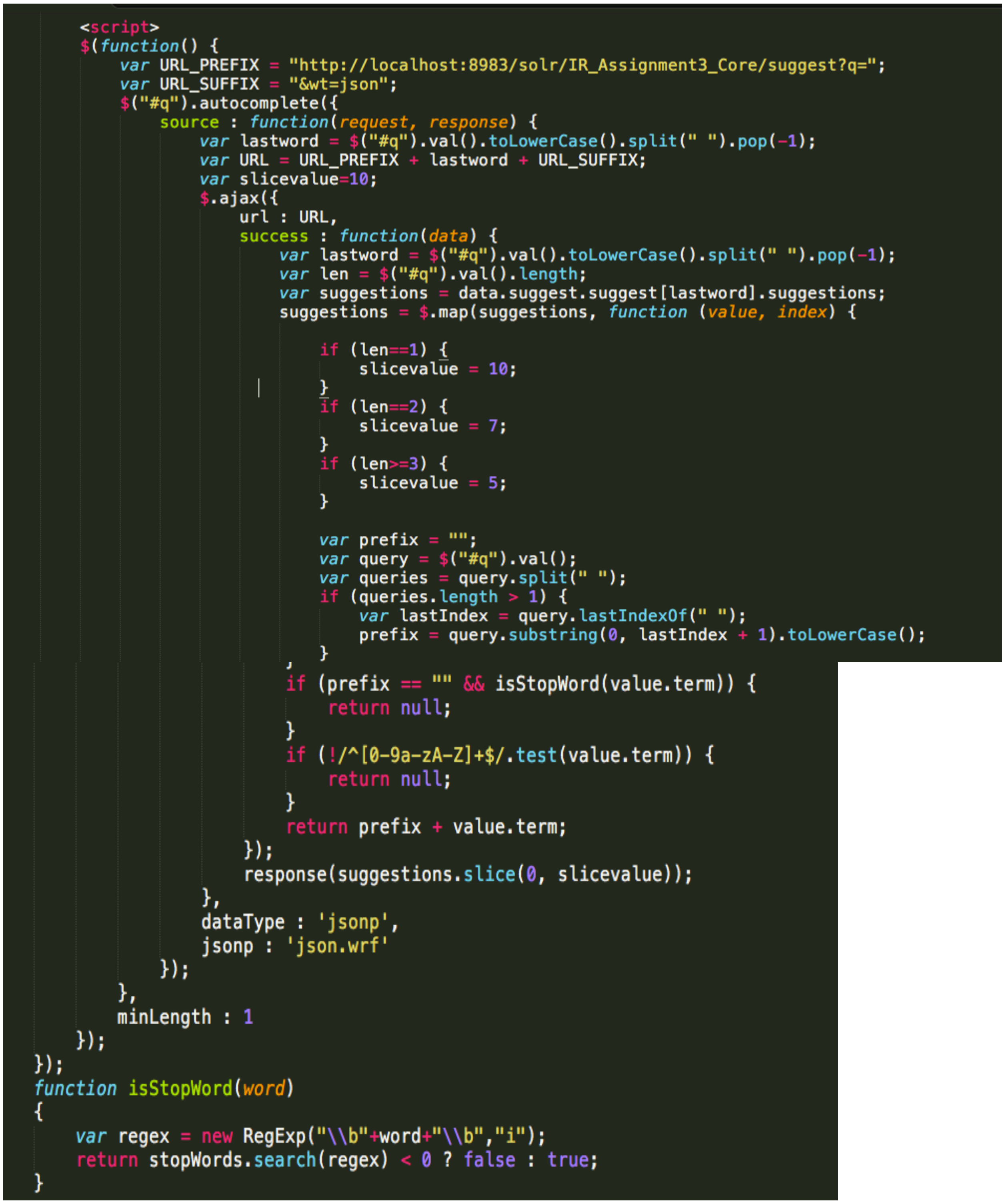
```
if ( count($corrector_arr) > 1 ) {
    while ( $iterater < count($corrector_arr) ) {
        // echo "Before : ".$corrector_arr[$iterater];
        $corrector_arr[$iterater] = SpellCorrector::correct($corrector_arr[$iterater]);
        // echo "After  : ".$corrector_arr[$iterater];
        if( $iterater == 0 ) {
            $corrector = $corrector.$corrector_arr[$iterater];
        }
        else {
            $corrector = $corrector." ".$corrector_arr[$iterater];
        }

        $iterater++;
    }
}
else {
    $corrector = SpellCorrector::correct($corrector_arr[0]);
    // echo "corrector is : ".$corrector;
}
```

to implement the auto complete part in out code, ajax calls are made to the solr server for ever key stroke entered. Along with this the length of the keystrokes are monitored so as to provide appropriate number of suggestions given by –



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 the third and subsequent characters between 1 and 4 suggestions should appear.

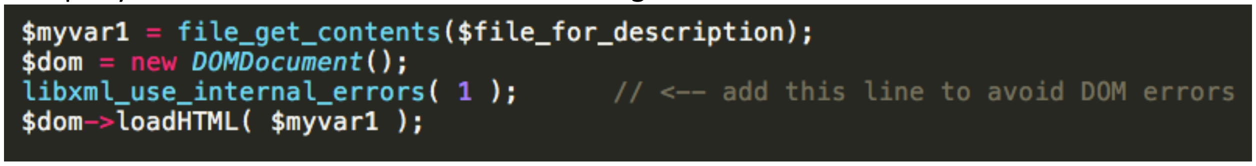
The code looks are follows –



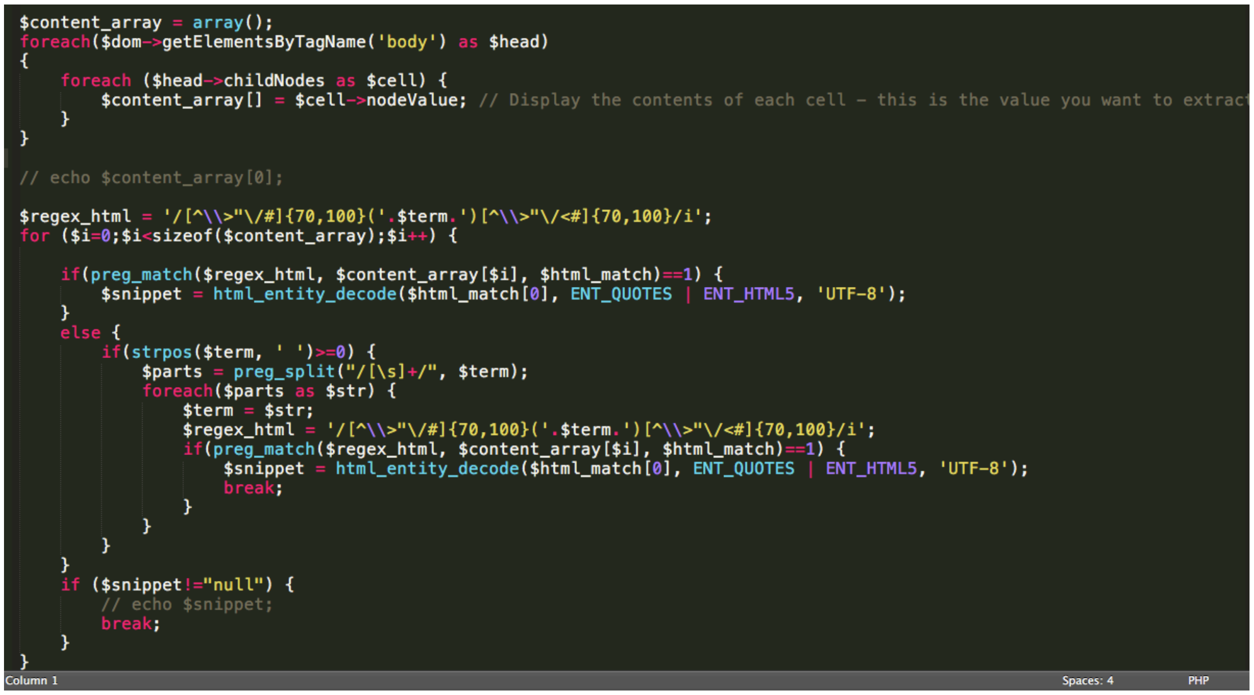
```
<script>
$(function() {
    var URL_PREFIX = "http://localhost:8983/solr/IR_Assignment3_Core/suggest?q=";
    var URL_SUFFIX = "&wt=json";
    $("#q").autocomplete({
        source : function(request, response) {
            var lastword = $("#q").val().toLowerCase().split(" ").pop(-1);
            var URL = URL_PREFIX + lastword + URL_SUFFIX;
            var slicevalue=10;
            $.ajax({
                url : URL,
                success : function(data) {
                    var lastword = $("#q").val().toLowerCase().split(" ").pop(-1);
                    var len = $("#q").val().length;
                    var suggestions = data.suggest.suggest[lastword].suggestions;
                    suggestions = $.map(suggestions, function (value, index) {

                        if (len==1) {
                            slicevalue = 10;
                        }
                        if (len==2) {
                            slicevalue = 7;
                        }
                        if (len>=3) {
                            slicevalue = 5;
                        }

                        var prefix = "";
                        var query = $("#q").val();
                        var queries = query.split(" ");
                        if (queries.length > 1) {
                            var lastIndex = query.lastIndexOf(" ");
                            prefix = query.substring(0, lastIndex + 1).toLowerCase();
                        }
                        if (prefix == "" && isStopWord(value.term)) {
                            return null;
                        }
                        if (!/^[0-9a-zA-Z]+$/.test(value.term)) {
                            return null;
                        }
                        return prefix + value.term;
                    });
                    response(suggestions.slice(0, slicevalue));
                },
                dataType : 'jsonp',
                jsonp : 'json.wrf'
            });
        },
        minLength : 1
    });
});
function isStopWord(word)
{
    var regex = new RegExp("\\b"+word+"\\b","i");
    return stopWords.search(regex) < 0 ? false : true;
}
```

here stopWords is a string variable with all possible stop words. This is taken from – http://www.ranks.nl/stopwords

SNIPPETS: This functionality was achieved by reading the local html file for a particular url and then extracting its contents as per a DOM structure and then applying regex rules for the query term over it. The code looks something like this –



```php
$myvar1 = file_get_contents($file_for_description);
$dom = new DOMDocument();
libxml_use_internal_errors( 1 );       // <-- add this line to avoid DOM errors
$dom->loadHTML( $myvar1 );
```



```php
$content_array = array();
foreach($dom->getElementsByTagName('body') as $head)
{
    foreach ($head->childNodes as $cell) {
        $content_array[] = $cell->nodeValue; // Display the contents of each cell - this is the value you want to extract
    }
}

// echo $content_array[0];

$regex_html = '/[^\\>"\/#]{70,100}('.$term.')[^\\>"\/<#]{70,100}/i';
for ($i=0;$i<sizeof($content_array);$i++) {

    if(preg_match($regex_html, $content_array[$i], $html_match)==1) {
        $snippet = html_entity_decode($html_match[0], ENT_QUOTES | ENT_HTML5, 'UTF-8');
    }
    else {
        if(strpos($term, ' ')>=0) {
            $parts = preg_split("/[\s]+/", $term);
            foreach($parts as $str) {
                $term = $str;
                $regex_html = '/[^\\>"\/#]{70,100}('.$term.')[^\\>"\/<#]{70,100}/i';
                if(preg_match($regex_html, $content_array[$i], $html_match)==1) {
                    $snippet = html_entity_decode($html_match[0], ENT_QUOTES | ENT_HTML5, 'UTF-8');
                    break;
                }
            }
        }
    }
    if ($snippet!="null") {
        // echo $snippet;
        break;
    }
}
```

- ANALYSIS OF THE REULTS:

  MISSPELLED WORDS –
  1. califona -> california
  2. yahop -> yahoo
  3. enrish -> enrich
  4. miltary -> military
  5. animrl -> animal

  AUTO-COMPLETION –
  1. universi -> university
  2. cont -> content
  3. textbo -> textbook
  4. retri -> retrieve
  5. goo -> google