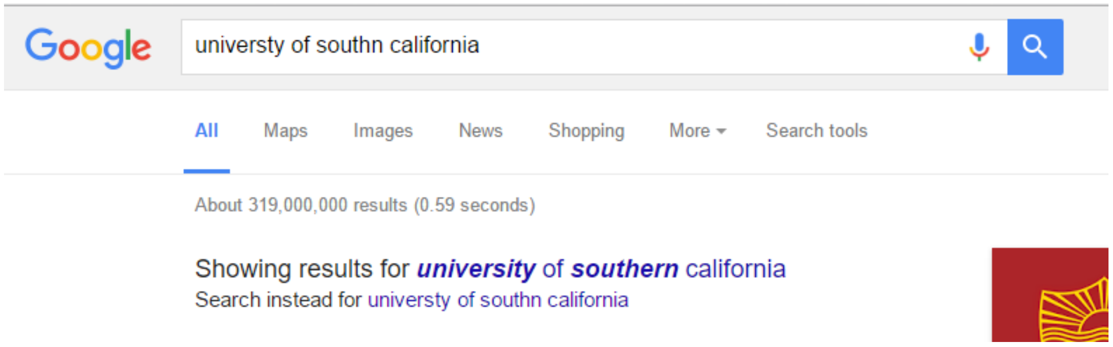
# Enhancing Your Search Engine
# With
# Spellcheck and Suggest in Solr

This tutorial explains how to enable spellchecker and autosuggest in Solr. ***HOWEVER, THE ACTUAL EXERCISE DOES NOT INVOLVE spellchecker in Solr.*** Rather you are required to use a separate spell checking program to build your own spell correct feature. ***The actual exercise is in a separate document***.

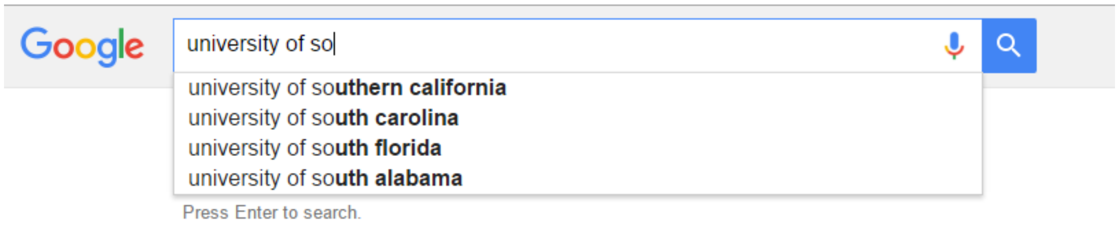
Current search engines automatically identify spelling mistakes in user's queries and return relevant results for the best matching query. The screenshot below shows how "Google" displays results for "university of southern california", even when the query had spelling mistakes.



About 319,000,000 results (0.59 seconds)

Showing results for *university* of *southern* california
Search instead for universty of southn california

Another powerful feature that is present in current search engines is autocomplete or suggest. You might have noticed, while you are typing queries into Google, there are several suggestions that come up in the dropdown below the input box.



university of so**uthern california**
university of so**uth carolina**
university of so**uth florida**
university of so**uth alabama**
Press Enter to search.

## Spellcheck Component

In Solr, the Spellcheck component is designed to provide inline query suggestions based on other, similar terms. The basis for these spelling suggestions can be terms in a field in Solr or an externally created text file. In this tutorial, we will use the terms from a field in Solr. This means that the query phrase input by the user will be matched with a term that has been previously indexed in Solr.

In the tutorial explaining indexing of files in Solr, we had created a field "_text_" as follows:

```
<field name="id" type="string" indexed="true" stored="true" required="true" multiValued="false" />
<field name="_version_" type="long" indexed="true" stored="true"/>
<field name="_root_" type="string" indexed="true" stored="false"/>
<field name="_text_" type="text_en_splitting" indexed="true" stored="false" multiValued="true"/>
```

This field contains all the text from other fields in Solr, as we have specified the copyField source as “*”. We will be using this field as the source for our spell check. The first step to enable this Spellcheck component is to specify the source of the terms in the **solrconfig.xml** file, which is present in the **conf** folder of the core.

```
<searchComponent class="solr.SpellCheckComponent" name="spellcheck">
  <lst name="spellchecker">
    <str name="classname">solr.IndexBasedSpellChecker</str>
    <str name="spellcheckIndexDir">./spellchecker</str>
    <str name="field">_text_</str>
    <str name="buildOnCommit">true</str>
  </lst>
</searchComponent>
```

The first element defines the searchComponent to use the solr.SpellCheckComponent. The classname is the specific implementation of the SpellCheckComponent, in this case solr.IndexBasedSpellChecker. Defining the classname is optional; if not defined, it will default to IndexBasedSpellChecker.

The spellcheckIndexDir defines the location of the directory that holds the spellcheck index, while the field defines the source field (defined in schema.xml) for spell check terms. When choosing a field for the spellcheck index, it's best to avoid a heavily processed field to get more accurate results. If the field has many word variations from processing synonyms and/or stemming, the dictionary will be created with those variations in addition to more valid spelling data.

Finally, *buildOnCommit* defines whether to build the spell check index at every commit (that is, every time new documents are added to the index). It is optional, and can be omitted if you would rather set it to false. In the figure above, we have specified “_text_” to be the field that is the source of our spelling suggestions.

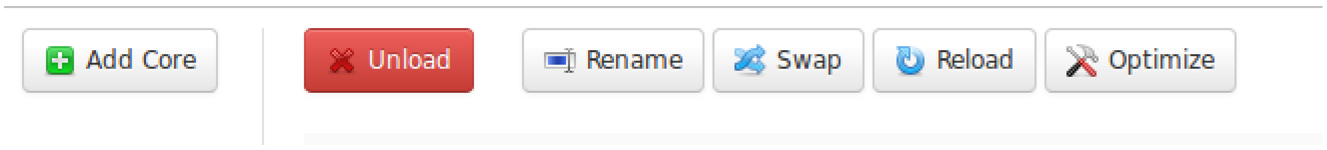
The second step is to add this spellcheck component to the requestHandler “select”. That is, for every query, if you want to perform a spellcheck. We have specified “spellcheck” to be “true” by default, by including it within the “defaults” list.

```
  -->
<requestHandler class="solr.SearchHandler" name="/select">
  <!-- 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>
  </lst>
  <arr name="last-components">
    <str>spellcheck</str>
  </arr>
</requestHandler>
```

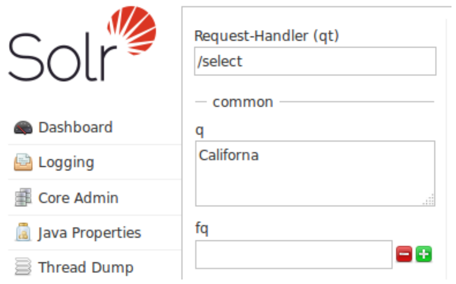
The spellcheck parameters in the above figure are explained below:

| | |
|---|---|
| spellcheck | Turns on or off SpellCheck suggestions for the request. If true, then spelling suggestions will be generated. |
| spellcheck.collate | Causes Solr to build a new query based on the best suggestion for each term in the submitted query. |

Once this addition is made, we need to reload the core. Go to the Solr UI -> Core Admin -> Select the core that you have created -> Click on reload button



Now, let us try to query the core with a spelling mistake. Here, I have misspelt “California” as “Californa”.



Request-Handler (qt)
/select

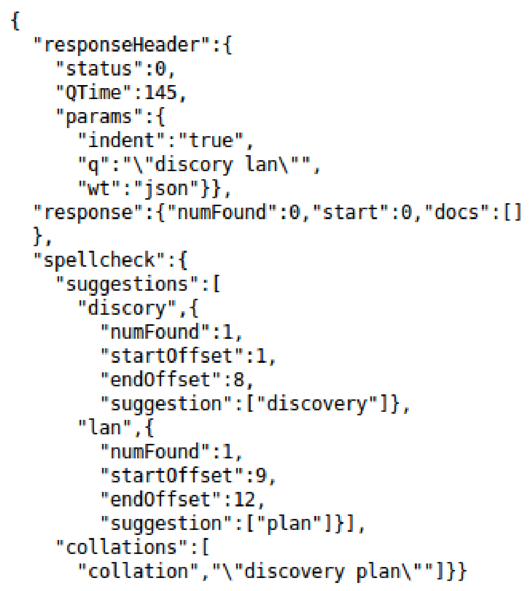
q
Californa

fq

This is the response obtained in json format, with the correct spelling suggested.

```
{
  "responseHeader":{
    "status":0,
    "QTime":17,
    "params":{
      "indent":"true",
      "q":"californa",
      "wt":"json"}},
  "response":{"numFound":0,"start":0,"docs":[]
  },
  "spellcheck":{
    "suggestions":[
      "californa",{
        "numFound":1,
        "startOffset":0,
        "endOffset":9,
        "suggestion":["california"]}],
    "collations":[
      "collation","california"]}}
```

Since, the spelling of the query has a mistake the number of results is 0, as indicated by “numFound”. In the spellcheck key of the json, we see that for our query “californa”, the suggestion is “california”.  Finally, solr has provided the “collations” field, which is a collated result. If the query is multi-phrased, then Solr will provide spelling suggestions for all the terms, and the collations will contain the combined phrase. For example, I am querying for “discovery plan”, but instead type “discory lan”. Solr provided spelling suggestions for both “discory” and “lan” in the “suggestions” field, and collates the best match for each of the phrases in the “collations” field. The spellcheck result is as shown:



```
{
  "responseHeader":{
    "status":0,
    "QTime":145,
    "params":{
      "indent":"true",
      "q":"\"discory lan\"",
      "wt":"json"}},
  "response":{"numFound":0,"start":0,"docs":[]
  },
  "spellcheck":{
    "suggestions":[
      "discory",{
        "numFound":1,
        "startOffset":1,
        "endOffset":8,
        "suggestion":["discovery"]},
      "lan",{
        "numFound":1,
        "startOffset":9,
        "endOffset":12,
        "suggestion":["plan"]}],
    "collations":[
      "collation","\"discovery plan\""]}}
```

More details on this feature can be found in Solr Wiki.

## Suggest Component

The SuggestComponent in Solr provides users with automatic suggestions for query terms. You can use this to implement a powerful auto-suggest feature in your search application.

Although it is possible to use the Spell Checking functionality to power autosuggest behavior, Solr has a dedicated SuggestComponent designed for this functionality.

The first step is to add a search component to solrconfig.xml and tell it to use the SuggestComponent. Here is some sample code that could be used.

```
<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>
```

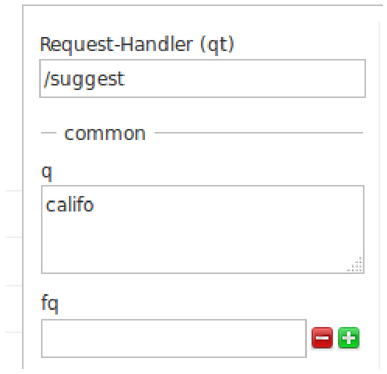
Here, the name of the “SuggestComponent” is “suggest”, the field used to obtain the terms for suggestion is the previously defined “_text_” field. The lookupImpl parameter defines the algorithms used to look up terms in the suggest index. There are several possible implementations to choose from, and some require additional parameters to be configured. This is a suggester in which similarity between the query and suggest terms is measured using Levenshtein algorithm. The field type to use for the query-time and build-time term suggestion analysis is defined by the “suggestAnalyzerFieldType”, here it is a string fieldtype.

After adding the search component, a request handler must be added to solrconfig.xml. This request handler works the same as any other request handler, and allows you to configure default parameters for serving suggestion requests. The request handler definition must incorporate the "suggest" search component defined previously.

```xml
<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>
```

In the screenshot above, we have defined a request handler for “suggest”. The default values for the number of suggestions is set to 5, by defining the value for element “suggest.count”, the default dictionary to be used is defined by “suggest.dictionary”.

Once the request handler is defined in solrconfig.xml, we need to reload the core as explained in the first section. Now let us try to see the suggest functionality in the Solr UI. Go to the Query view, and change the requesthandler from “/select” to “/suggest”. In this example, I have typed just “califo” in the query, and executed the query.



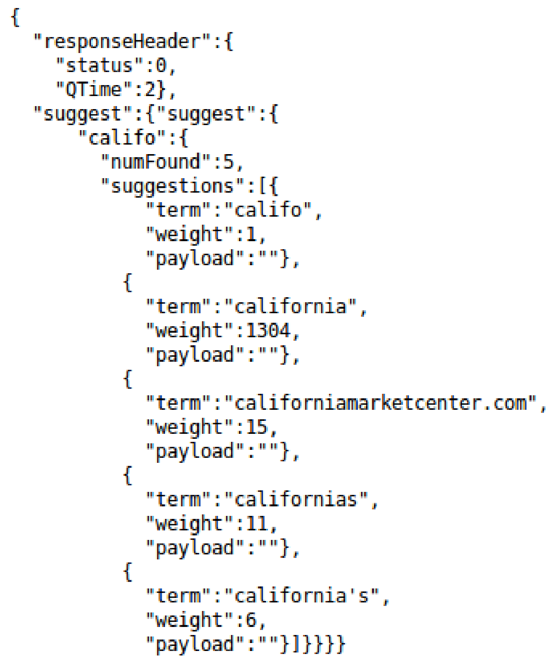
Request-Handler (qt)
/suggest

common

q
califo

fq

The following is the response of the Solr query in json format. Here we can see that, for the term “califo”, 5 suggestions have been returned each in the order of the weight. The weight specifies its match with the original term.



```
{
  "responseHeader":{
    "status":0,
    "QTime":2},
  "suggest":{"suggest":{
      "califo":{
        "numFound":5,
        "suggestions":[{
            "term":"califo",
            "weight":1,
            "payload":""},
          {
            "term":"california",
            "weight":1304,
            "payload":""},
          {
            "term":"californiamarketcenter.com",
            "weight":15,
            "payload":""},
          {
            "term":"californias",
            "weight":11,
            "payload":""},
          {
            "term":"california's",
            "weight":6,
            "payload":""}]}}}}
```

You can leverage this suggest feature, by making the “suggest” request before actually making the “select” request to Solr, when the user is typing in the query. “Select” request can be sent only when the user presses “Enter”.

More details on this feature can be found in Solr Wiki.