Note: This document contains documentation for functionality shared by all Sematext ReSearcher components.

How Fixing of Common Misspellings Works

Fixing common misspellings is the first step in the algorithm of all ReSearcher components. If the file is provided and defined with property commonMisspellingsFile, the component will first check if there were any common language misspellings in the original query. If misspellings were found and if the corrected query returns a satisfying number of results (defined with `maxOriginalResults`), it will be returned as a suggestion. If not, the component will construct queries based on original Solr Spellchecker's suggestions.

Definitions of common language misspellings are defined in the file in the following format:

```
abandonned->abandoned
aberation->aberration
abilties->abilities
abilty->ability
```

In case the word from the left side appears somewhere in the original query, it will be replaced with the word from the right side. This functionality is similar to Solr's handling of synonyms, but by using it with ReSearcher components, one can also benefit from:

- getting results of both the original and the corrected query, so users can see what each query would return
- correction highlighting feature (more about it in a separate section) – Sematext's ReSearcher components can mark the differences between the original and the corrected query so the users can easily see what was corrected. For instance, if query “Once upon a time in Amerika” was corrected to “Once upon a time in America”, correction highlighting feature would also generate this string:

“Once upon a time in <b>America</b>”

- Sematext provides the default list of common word misspellings for English language (you can find it in example/solr/conf/common_misspellings_en.txt).

Correction highlighting

All Sematext's ReSearcher components support highlighting of corrections. To enable highlighting functionality, provide at least one of the following query parameters in the request URL:

- `res.hl.removed` – the name of the tag which should mark words removed from the original query. For instance, if parameter was specified as “res.hl.removed=strike”, highlighted version of the query might look like “harry potter <strike>cvd</strike>”
- `res.hl.replaced` – the name of the tag which should mark words that replaced words from the original query. For instance, if parameter was specified as “res.hl.replaced=b”, highlighted version of the query might look like “harry potter <b>dvd</b>”

Both parameters can be specified at the same time. In such cases, highlighted queries might look like this:

“Solr <strike>searc</strike> server and <b>Lucene</b>”

There is another parameter that can be specified as a request parameter:

- `res.hl.ignorequotes` – can be true (default) or false. In case it is set to true, the removed quotes will not be marked in highlighted query. If it is set to false, the removed quotes will be marked the same as other removed words (with the tag defined with `res.hl.removed`).

Absolutely nothing needs to be specified in `solrconfig.xml`.

Suggestions with highlighted corrections are returned in separate XML tag in the response. In case of DYM ReSearcher component, this XML tag would be named `extended_spellchecker_suggestions_highlighted` , and it would be provided along with pure-text version of suggestions, like this:

```
<arr name="extended_spellchecker_suggestions_highlighted">
      <str>&lt;b&gt;foo:bob&lt;/b&gt; AND foo:marley</str>
      <str>&lt;b&gt;foo:bono&lt;/b&gt; AND foo:marley</str>
</arr>
<arr name="extended_spellchecker_suggestions">
      <str>foo:bob AND foo:marley</str>
      <str>foo:bono AND foo:marley</str>
</arr>
```

Usage with SolrJ

Suggestions produced by ReSearcher components can easily be read from Solr's response by using SolrJ API. Here is an example code snippet:

```
    List<String> suggestions = null;
    SolrDocumentList sugDocList = null;
    if (queryResponse.getResponse() != null) {
      // extract DYM ReSearcher's suggestions and documents for best suggestion
      suggestions = (List<String>)
queryResponse.getResponse().get("extended_spellchecker_suggestions");
      sugDocList = (SolrDocumentList)
queryResponse.getResponse().get("spellchecked_response");
    }

    if (queryResponse.getSpellCheckResponse() != null &&
queryResponse.getSpellCheckResponse().getCollatedResult() != null) {
      if (suggestions == null) {
        suggestions = new ArrayList<String>();
      }
      // also, extract collated suggestion returned by Solr's original
      // Spellchecker, in case you want to display it too
      String collatedResult = 
queryResponse.getSpellCheckResponse().getCollatedResult();
      if (!suggestions.contains(collatedResult)) {
        suggestions.add(collatedResult);
      }
    }
```

Note that this piece of code works for DYM ReSearcher component, but it can easily be modified to work with any other ReSearcher component by just changing the names of the fields (for instance, instead of “`extended_spellchecker_suggestions`” use “`relaxer_suggestions`”).