Auto-Complete Suggestion Basing on Solr

515030910553 Sun Rui

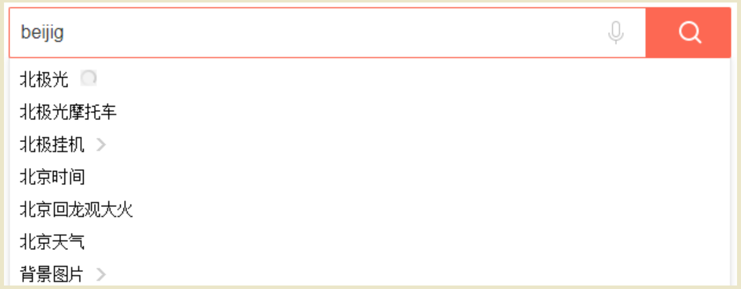
Backgroun
ds

Auto-Complete Search

Solr Search Engine

Auto-Complete Search

- Auto-complete, or word completion, is a feature in which an application predicts the rest of a word a user is typing. It speeds up human-computer interactions when it correctly predicts the word a user intends to enter.
- In search engines, auto-complete user interface features provide users with suggested queries or results as they type their query in the search box. This is also commonly called auto-suggest or incremental search.



beijig

北极光
北极光摩托车
北极挂机
北京时间
北京回龙观大火
北京天气
背景图片

Solr Search Engine

- Solr (pronounced "solar") is an open source enterprise search platform, written in Java, from the Apache Lucene project. Its major features include full-text search, hit highlighting, faceted search, real-time indexing, dynamic clustering, database integration, NoSQL features and rich document handling.
- Apache Solr is a powerful tool with tremendous search capability. In order to search a document, it performs following operations in sequence:
  - Indexing
  - Querying
  - Mapping
  - Ranking

Solr Search Engine

Indexing: First of all, it converts the documents into a machine-readable format which is called Indexing.

Querying:Understanding the terms of a query asked by the user. These terms can be images, keywords, and much more.

Mapping: Solr maps the user query to the documents stored in the database to find the appropriate result.

Ranking the outcome: As soon as the engine searches the indexed documents, it ranks the outputs as per their relevance.

Suggestion

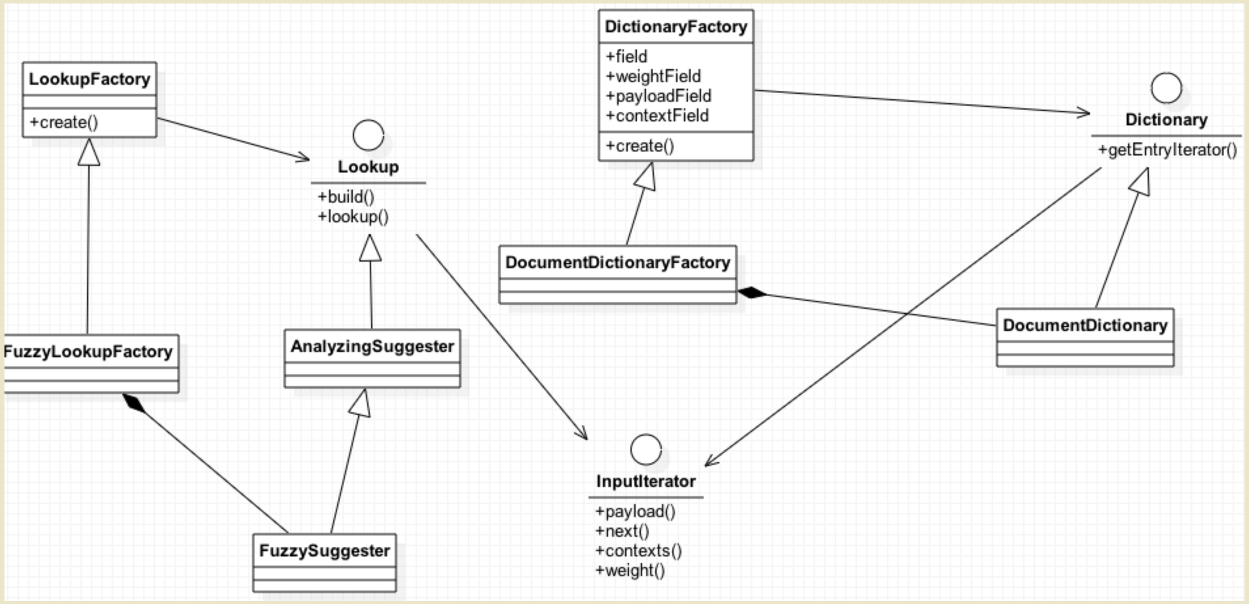
Auto complete basing on Solr

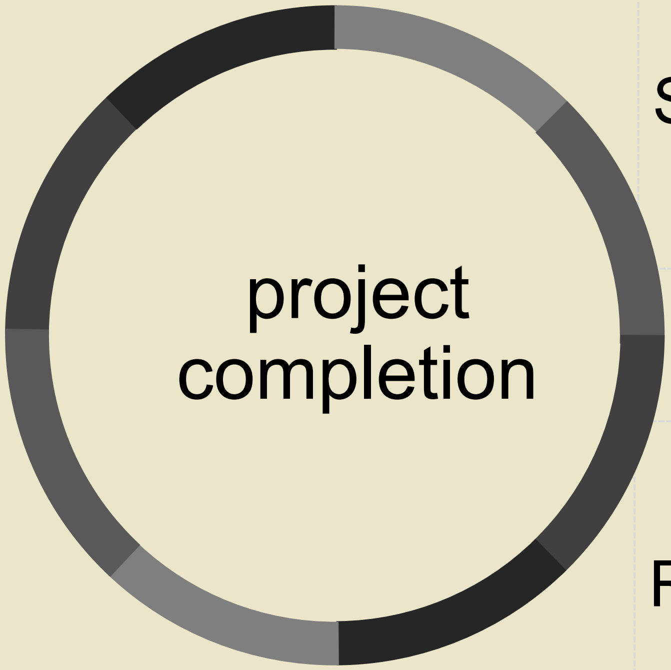
Auto complete suggestion on Solr

- This project is meant to realize the suggester function basing on the apache solr search engine.The suggester completion may come from a dictionary that is based upon the main index or upon any other arbitrary dictionary, data or file. The result will be only top-N suggestions, either ranked alphabetically or according to their usefulness for an average user or weight defined by the system.
- The Suggester Component in Solr provides users with automatic suggestions for query terms.This approach utilizes Lucene’ s Suggester implementation and supports all of the lookup implementations available in Lucene.(The structure is on the next page)
- In my solr auto-complete project, I realize this function basing the spell check component and redefine the suggest component and request handler. The search can be based on database or file-based data.

Auto complete suggestion on Solr

- The main features of Suggester structure are.:
  - Lookup implementation pluggability
  - Term dictionary pluggability, giving you the flexibility to choose the dictionary implementation
  - Distributed support



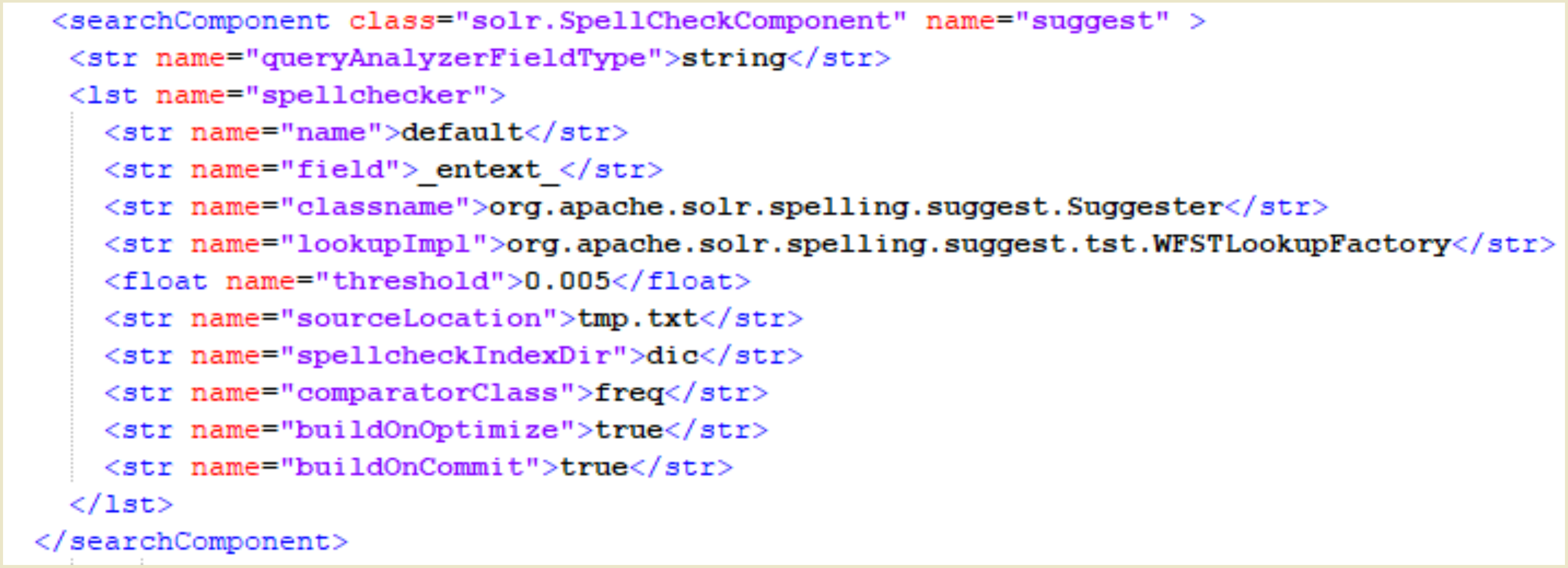


Suggest Component

Request Handler

Suggest Component

- In the suggest component, we reuses much of the SpellCheckComponent infrastructure.
- The code are as below:



```
  <searchComponent class="solr.SpellCheckComponent" name="suggest" >
   <str name="queryAnalyzerFieldType">string</str>
   <lst name="spellchecker">
     <str name="name">default</str>
     <str name="field">_entext_</str>
     <str name="classname">org.apache.solr.spelling.suggest.Suggester</str>
     <str name="lookupImpl">org.apache.solr.spelling.suggest.tst.WFSTLookupFactory</str>
     <float name="threshold">0.005</float>
     <str name="sourceLocation">tmp.txt</str>
     <str name="spellcheckIndexDir">dic</str>
     <str name="comparatorClass">freq</str>
     <str name="buildOnOptimize">true</str>
     <str name="buildOnCommit">true</str>
   </lst>
 </searchComponent>
```

Suggest Component-on dadtabase

parameter	Function
queryAnalyzerFieldType	The fieldType type in managed-schema. If this option is added, the spelling checker will call this fieldType tokenizer. If it is not added, Solr will take the word splitter of field defined below or just created a segmentation based on space because SpellCheckComponent requires a word breaker to run. In our project, we now hope that Analyzer does not make any changes to the query, so select string.
name	Same as the field defined in request handler.
lookupImpl	The look-up of matching suggestions in a dictionary is implemented by subclasses of the Lookup class.

*There are four main implementations that are included in solr:

- JaspellLookup - tree-based representation based on Jaspell
- TSTLookup - ternary tree based representation, capable of immediate data structure updates
- FSTLookup - finite state automaton based representation
- WFSTLookup - weighted automaton representation: an alternative to FSTLookup for more fine-grained ranking

Direct benchmarks indicate that (W)FSTLookup provides better performance and has a much lower memory cost compared to the other two methods so I choose WFSTLookupFactory.

Suggest Component-on dadtabase

parameter	**Function**
threshold	A value between 0 and 1, representing the minimum fraction of documents where a term should appear. This will limiting some infrequent words. But if the search is a file-based dictionary, this parameter will be useless.
comparatorClass	Return the sort of the result. There are four different types: ➢ empty: If not settled, the default will call this ➢ score: Explicitly choose the default case ➢ freq: Sort by frequency first, then score ➢ a fully qualified class name: Provide a custom comparator that implements Comparator
buildOnOptimize/buildOnCommit	If set to true then the Lookup data structure will be rebuilt after commit/optimize. If false (default) then the Lookup data will be built only when requested. Noticing that only when this parameter is set as true the spellchecker will work.

Suggest Component-on dictionary

- This part is defined in the suggest component as:

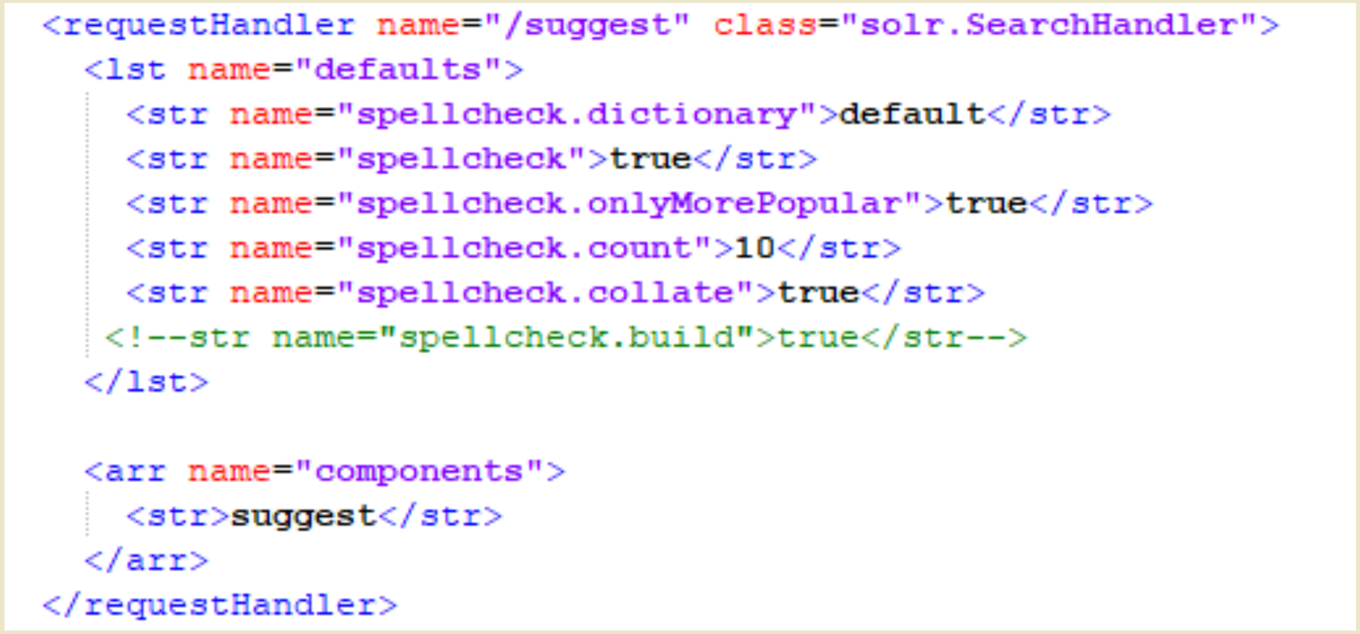
  <str name="sourceLocation">tmp.txt</str>

  <str name="spellcheckIndexDir">dic</str>

- When a file-based dictionary is used then it's expected to be a plain text file in UTF-8 encoding. Each line must be consist of either a string without literal TAB character, or a string and a TAB separated floating-point weight. If weight is missing, it will be assumed as 1.0.
- Weights affect the sorting of matching suggestions when *spellcheck.onlyMorePopular=true* is selected - weights are treated as "popularity" score.
- *spellcheckIndexDir* is the index file directory for recommended check. Solr will create this folder at startup and store the index of the check suggestion under this folder.

Request Handler

- After defining the suggest component, we need to add a new handler that uses SearchHandler with with SearchComponent that we just defined.
- The code are as below:

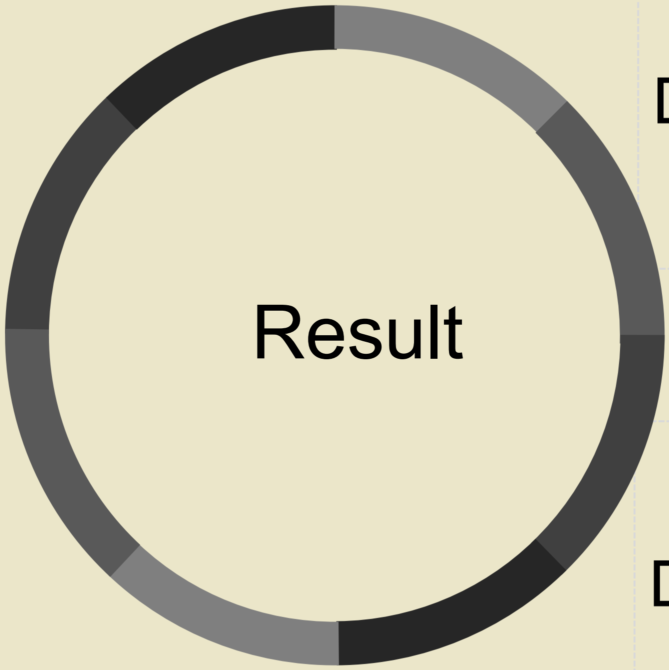


```
<requestHandler name="/suggest" class="solr.SearchHandler">
  <lst name="defaults">
    <str name="spellcheck.dictionary">default</str>
    <str name="spellcheck">true</str>
    <str name="spellcheck.onlyMorePopular">true</str>
    <str name="spellcheck.count">10</str>
    <str name="spellcheck.collate">true</str>
   <!--str name="spellcheck.build">true</str-->
  </lst>

  <arr name="components">
    <str>suggest</str>
  </arr>
</requestHandler>
```

Request Handler

parameter	**Function**
spellcheck	Run the Suggester for queries submitted to this handler.
count	Configure the number of spell check prompt results.
onlyMorePopular	If this parameter is set to true then the suggestions will be sorted by weight ("popularity") - the count parameter will effectively limit this to a top-N list of best suggestions. If this is set to false then suggestions are sorted alphabetically.
collate	Provide a query collated with the first matching suggestion.
build	Use this function to reconstruct the index when the core reloads.

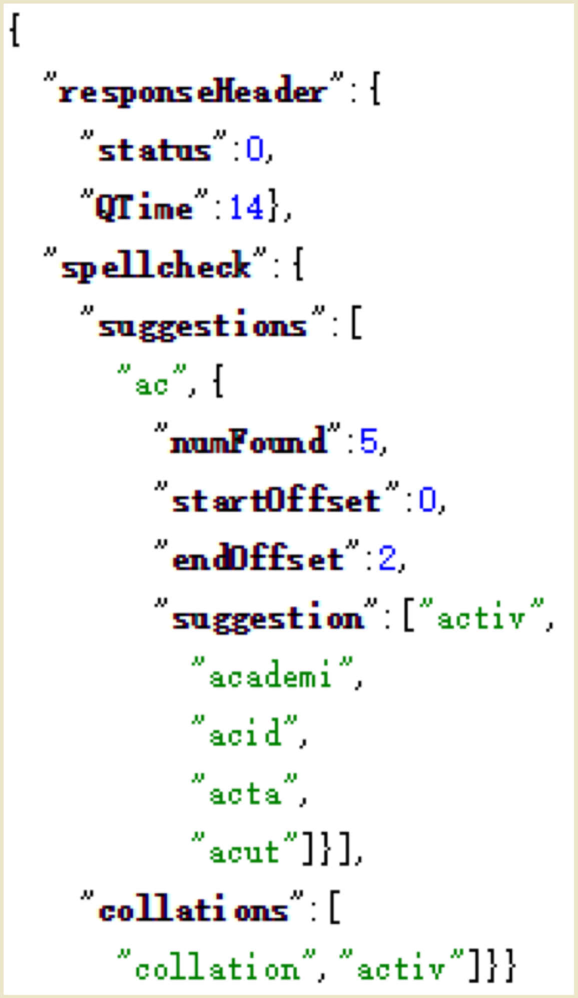


Data search result

Dictionary search Result

Data search result

- If we put "ac" as the input qt basing on , the result is:



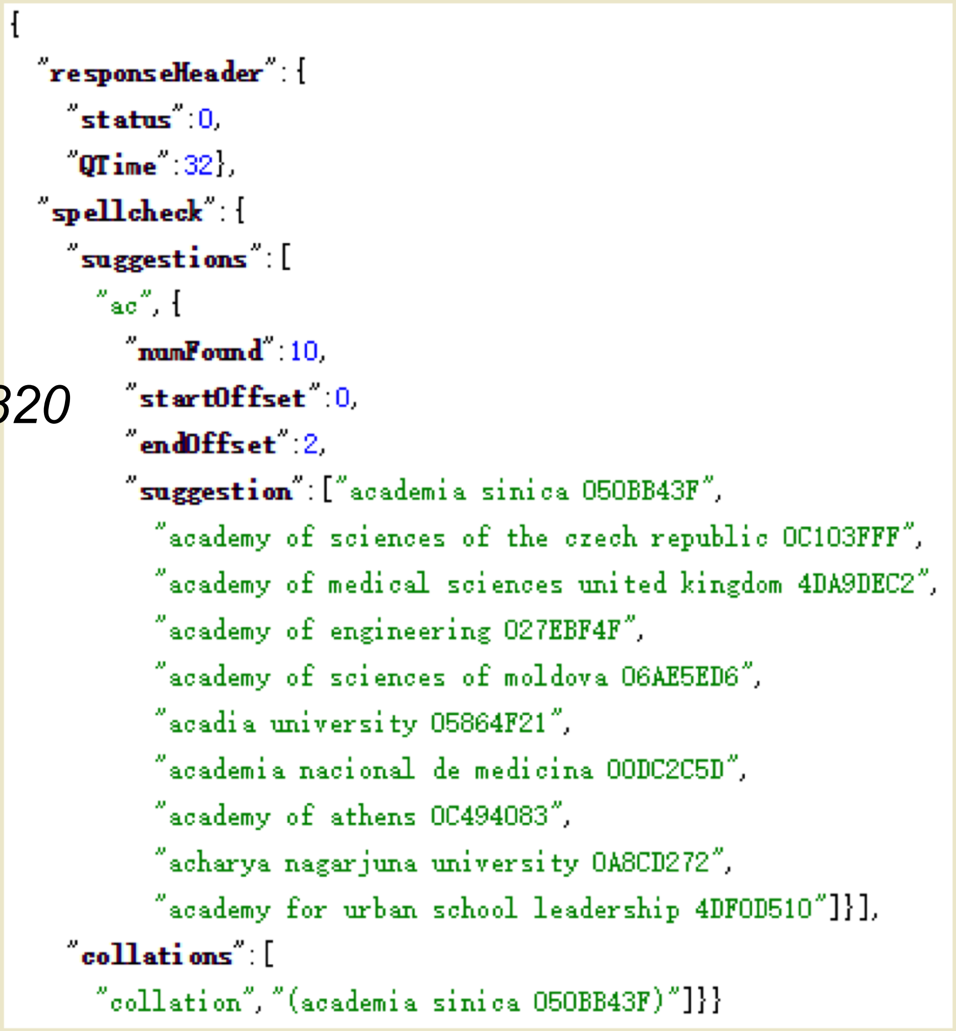
```
{
  "responseHeader":{
    "status":0,
    "QTime":14},
  "spellcheck":{
    "suggestions":[
      "ac",{
        "numFound":5,
        "startOffset":0,
        "endOffset":2,
        "suggestion":["activ",
          "academi",
          "acid",
          "acta",
          "acut"]}],
    "collations":[
      "collation","activ"]}}
```

Data search result

- he give dictionary type is as (paper_name paper_ID TAB weight), for example:

  sangji university 00002523   820

- If we put "ac" as the input qt basing on , the result is:



```
{
  "responseHeader":{
    "status":0,
    "QTime":32},
  "spellcheck":{
    "suggestions":[
      "ac",{
        "numFound":10,
        "startOffset":0,
        "endOffset":2,
        "suggestion":["academia sinica 050BB43F",
          "academy of sciences of the czech republic 0C103FFF",
          "academy of medical sciences united kingdom 4DA9DEC2",
          "academy of engineering 027EBF4F",
          "academy of sciences of moldova 06AE5ED6",
          "acadia university 05864F21",
          "academia nacional de medicina 00DC2C5D",
          "academy of athens 0C494083",
          "acharya nagarjuna university 0A8CD272",
          "academy for urban school leadership 4DF0D510"]}],
    "collations":[
      "collation","(academia sinica 050BB43F)"]}}
```

Conference

1. *https://en.wikipedia.org/wiki/Autocomplete*
2. *https://en.wikipedia.org/wiki/Apache_Solr*
3. *https://wiki.apache.org/solr/Suggester*
4. *https://lucene.apache.org/solr/guide/6_6/solrcloud.html*
5. *http://iamyida.iteye.com/blog/2205114*
6. *http://public.dhe.ibm.com/software/dw/java/j-solr1-pdf.pdf*