Solr with RankingAlgorithm ver1.1

By

Nagendra Nagarajayya
transaxtions llc (solr-ra.tgels.com)

Table of Contents

Solr with RankingAlgorithm ver 1.1

By

Nagendra Nagarajayya
transaxtions llc (solr-ra.tgels.com)
updated 04/10/2010

1. Introduction

Solr (a lightning fast, open source search platform) can now work with a new search library using the RankingAlgorithm instead of Lucene, the default search library. Solr with RankingAlgorithm ranking seems to be comparable to Google site search (see Perl index comparison results) for certain queries and is much better than Lucene.

RankingAlgorithm enables Solr to rank product searches very accurately and also enables Near Real Time Search. It has two modes, a document mode that ranks documents by relevance and a product mode that ranks precisely based on occurrence. The RankingAlgorithm has been integrated into Solr in such a way that either Lucene or the RankingAlgorithm can be used to do the search. The RankingAlgorithm scoring or working does not break any of the existing functionality. So shrad, faceting, highlighting, etc. still work as before. RankingAlogirhtm only uses Lucene Apis for retrieving documents and terms from the index and uses its own scoring to rank documents.

2. Using Solr with RankingAlgorithm

There is no change in the way you access Solr. All searches work the same as before.

So a Solr search such as:

http://localhost:8773/solr/?q=california gold rush&fl=score

should still work as before. The only difference is that Solr instead of using Lucene library for search, uses the RankingAlogrithm library to search and rank the document. The returned scores are different from Lucene and reflects the relevancy of a document. For eg. Searching an index made up of the first 1 million documents in the wikipedia for the terms, “california gold rush” brings back:

```
California Gold Rush
Gold rush
Gold Country
List of people associated with the California Gold Rush
History of California to 1899
California Mining and Mineral Museum
Pike's Peak Gold Rush
```

The RankingAlgorithm scores in two different modes, Document mode and Product mode. In Document mode, the scoring is for relevance and in Product mode, scoring is for occurrence. Document mode is suitable for general purpose searches such as Wikipedia docs, HTML, Word/PDF or similar docs. The Document mode is the default. Product mode is for searches found on retail stores, online store/shopping/comparison/auction websites etc, including short text sites like tweeter. Product mode can be enabled by adding “mode=product” to the search query.   For eg. If the search is for “wii console”:

http://localhost:8773/solr/?q=wii console&fl=score&mode***=product***

Product mode takes a term occurrence into account and scores accordingly. Products titles starting with “wii console” are ranked first, and the others rank lower as the occurrence of “wii console” shifts in the title or gets reversed, see below:

```
Wii Console and Wii Fit Plus with Balance Board Bundle (Nintendo Wii)
Wii Console System with Wii Sports Resort Game with TWO MotionPlus Attachments
Nintendo Wii Console w/ Bonus Wii Sports Resort Bundle, Black
Pelican Accessories Wii Console Stand - Nintendo Wii
Grafitti Skin for Nintendo Wii Console
NEW AC Adapter Cable Cord Power Supply For NINTENDO WII Gaming Console
Wii Remote Charging Console Stand
Nintendo Wii Skin - System Console Skin and two Wii Remote Skins - Blue Vortex
CET Domain 10301901 Console Stand Station for Nintendo Wii
```

Product mode has multiple scan modes, where the scan is a fast scan, medium scan or a full scan. Scan is looking up the terms in the index and the process used. The default is fast scan, while medium and full scan can be enabled by adding the below parameters:

scan=fast or scan=medium or scan=full

http://localhost:8773/solr/?q=wii console&fl=score&***scan=medium***&mode=product
or
http://localhost:8773/solr/?q=wii console&fl=score&***scan=full***&mode=product

The full scan is the most accurate but is slower than the medium scan. The medium scan is slower than the fast scan.

2.1 Near Real Time Search

With Solr-RA documents can be added in near real time without a commit and without closing the Index Searchers and clearing the cache. As there is no commit, the indexing is quite fast while enabling searches concurrently. A lock-free concurrent time managed access is used to eliminate locking between the IndexWriter and the IndexSearchers. A 262 TPS index write on a dual core intel system with 2GB heap has been observed with searches in parallel.

Steps to enable RT

```
Add
        <realtime>true</realtime>
        <library>rankingalgorithm</library>

to solrconfig.xml
```

2.3 Using Lucene to search

You can still use Lucene as the search library by adding the parameters, lucene=true. This turns off the RankingAlgorithm and uses the default Lucene library for the search.

http://localhost:8773/solr/?q=wii console&fl=score&***lucene=true***

2.4 Configuring options in solrconfig.xml

```
      <realtime>true</realtime> <!-- true to enable near real time or false -->
      <library>rankingalgorithm</library> <!--rankingalgorithm or lucene -->
```

```
<rankingalgorithm>
   <mode>document</mode> <!-- document or product mode -->
    <scan>fast</scan> <!-- fast, medium, full works in product mode -->
</rankingalgorithm>
```

3. Installing Solr with the RankingAlgorithm

You can install Solr with RA in two different ways. You can download Solr1.4.1 with RA.zip a bundle of Apache Solr 1.4.1 and Ranking Algorithm (a big download) or just download the solr-ra.war.zip which is a web archive file and copy it into an existing or new Solr 1.4.1 installation. Below the are steps for both:

3.1 Download Solr1.4.1 with RA.zip (bundle)

Installation is the same as Solr. Download Solr1.4.1 with RA.zip (Solr version 1.4.1 with the RankingAlgorithm) from solr-ra.tgels.com. Unzip or untar the file, change to examples directory and start Solr as before, java -Xmx 2gb -jar start.jar.

Step1:

Download Solr1.4.1 with RA.zip from http://solr-ra.tgels.com

Step2:

Unzip it to a directory

Step3:

cd unzip directory/apache-solr-1.4.1/examples

Step4:

bash -x start_solr.sh

or

java -Xmx 2gb -jar start.jar.

Step 5:

Configuring options in solrconfig.xml:

```
      <realtime>true</realtime> <!-- true to enable near real time or false -->
      <library>rankingalgorithm</library> <!--rankingalgorithm or lucene -->
      <rankingalgorithm>
         <mode>document</mode> <!-- document or product mode -->
          <scan>fast</scan> <!-- fast, medium, full works in product mode -->
      </rankingalgorithm>
```

3.2 Download Solr1.4.1 with RA.war (war file)

Instead of downloading the Solr 1.4.1 with RA (a huge file), you can download just the solr_ra.war file. You will still need to download the Solr 1.4.1 from the Solr website as below. Unzip that first, and then change to the examples directory and follow the steps as below.

Step1:

Download Solr 1.4.1 from the Apache Solr website, as in here:
http://www.apache.org/dyn/closer.cgi/lucene/solr/

Step2:

Install Solr 1.4.1 by unzip it to a directory.

Step3:

Download the solr_ra war file from solr-ra.tgels.com, as in here:
http://solr-ra.tgels.com/solr-ra.jsp
(click on download war file link at the bottom of the page)

Step4:

cp solr_ra.war.zip file to the examples directory under unzip directory/apache-solr-1.4.1/examples.

Step5:

unzip solr_ra.war.zip

Step 6:

cp solr.war webapps

Step 7:

bash -x start_solr.sh

or

java -Xmx 2gb -jar start.jar.

Step 8:

Configuring options in solrconfig.xml

```
<realtime>true</realtime> <!-- true to enable near real time or false -->
<library>rankingalgorithm</library> <!--rankingalgorithm or lucene -->
<rankingalgorithm>
   <mode>document</mode> <!-- document or product mode -->
    <scan>fast</scan> <!-- fast, medium, full works in product mode -->
</rankingalgorithm>
```

3.3 Installing on Glassfish/Tomcat/JBoss/WebLogic

If you want to deploy Solr on a different container than the default Jetty container, then deploy as before (ie. Deploy examples/webapps/solr.war on Tomcat or Glassfish or Weblogic or Jboss).

4. Using the RankingAlgorithm library

Download the RankingAlgorithm jar file from here:

http://solr-ra.tgels.com/rankingalgorithm.jsp

(Click on download link)

Add the rankingalgorithm.jar file to your classpath.

Using RankingAlgorithm to search is extremely simple since it make uses of the Lucene index to access the index. If you already have a Lucene Index, then you can use that as the first argument, see example code below or at http://solr-ra.tgels.com/downloads/code/Example.java:

```
                RankingQuery rq = new RankingQuery();
                IndexSearcher is = new ~~IndexSearcher~~(index);
                StandardAnalyzer analyzer = new ~~StandardAnalyzer~~();
                QueryParser parser = new ~~QueryParser~~(field, analyzer);
                Query query = parser.parse(searchterms);
                RankingHits rh = rq.search(query, is);
                System.out.println("num hits=" + rh.getNumHits() + "--no docs=" +
.maxDoc());
                for (int i=0; i<rh.getNumHits() && i<10; i++) {
                      System.out.println("i=" + i + "--" + rh.score(i) + "--docid=" +
.docid(i) + "--doc=" + is.doc(rh.docid(i)).get(title) );
                }
```

Make sure Lucene is also in the classpath since the RankingLibrary uses it to access the index but uses its own ranking and scoring to rank the documents in the index. That is it. Very simple to use but gets you very accurate and relevant results.

The default is the document mode. To enable product mode:

```
RankingQuery rq = new RankingQuery();
rq.setMode(RankingQuery.PRODUCT_MODE);
```

5. Conclusion

Solr with RankingAlgorithm offers a new search library along with Near Real Time Search. RankingAlgorithm ranking seems to be comparable to Google site search (see Perl index comparison results) and much better than Lucene.

In document mode RankingAlgorithm ranks documents relevantly while ranking very accurately and precisely in the product mode. Document mode is very well suited for searching html, wikipedia, pdf/word type documents, while product works very well with short text as in retail websites, product comparison websites, short text messaging like twitter, etc. RankingAlgorithm with document and product mode is very well suited for the enterprise as well as the retail, ecommerce and websites.

The near real time search in Solr-RA works well and allows concurrent search with indexing in parallel without closing the IndexSearchers or clearing the cache providing the ability to offer searches in near real time. The indexing performance observed on a 2 core intel system with Fedora Linux 12 is about 262 tps (new document adds). This could be improved to a very high number (from 14 secs for indexing about 3900 documents to about 2 secs) if IndexWriter.getReader() performance is improved; at the moment, it takes about 70-90 ms to get a IndexReader.

6. References

1. Solr with RA, http://solr-ra.tgels.com/solr-ra.jsp
2. RankingAlgorithm, http://rankginalgorithm.tgels.com/rankingalgorithm.jsp
3. Apache Solr, http://projects.apache.org/projects/solr.html
4. Apache Lucene, http://projects.apache.org/projects/lucene_java.html