solrpres

March 2, 2015

1 A quick tour of Lucene and Solr

Lucene is a widely used information retrieval library in Java.

- implements the TF-IDF model
- simple model for indexing and search
- easy to learn, fast, and scalable
- written by Doug Cutting, who later co-wrote Hadoop
- I've been using it since 2001 (DSpace, Canary, WDL, Chronam, GW Libs)
- free, open source software

1.1 Lucene basics

- everything is a Document (source format independent)
- Documents have Fields
- Fields can be tokenized, indexed, stored, multivalue (or not)
- text is Tokenized, Analyzed on index and query
- queries are parsed, then Tokenized, Analyzed, and results ranked

See a simple example of code using the Lucene API and also more extensive Lucene documentation.

1.2 Solr basics

Solr is a search application (or server) that makes Lucene even easier to implement and scale.

- Lucene under the hood
- simple "index anything" or schema-based configuration
- simple web API to index and search
- scales to many servers
- very widely used
- free, open source software

2 Working with Solr

- index and search using web API - easier code
- define a schema first, or don't
- csv, json, xml results easy to parse in any language
- handles many technical details (caching, faceting, spelling corrections, clustering, hit hightlighting)
- the bee's knees, basically

2.1 Searching solr

It's as easy as sending an HTTP request. Let's use python-requests to handle HTTP.

```
In [1]: import requests
        req = requests.get("http://localhost:8983/solr/gettingstarted/select?wt=json&q=foundation")
        resp = req.json()
        resp.keys()

Out[1]: [u'responseHeader', u'response']
```

The response header is pretty straightforward:

```
In [2]: resp['responseHeader']

Out[2]: {u'QTime': 38,
         u'params': {u'indent': u'true', u'q': u'foundation', u'wt': u'json'},
         u'status': 0}

In [3]: resp['response'].keys()

Out[3]: [u'start', u'maxScore', u'numFound', u'docs']

In [6]: resp['response']['start']

Out[6]: 0

In [7]: resp['response']['maxScore']

Out[7]: 0.37489003

In [8]: resp['response']['numFound']

Out[8]: 3106
```

That's the basic header. We're starting with the first page of results, and it's a zero-based index, which explains why "start" is 0. We'll look more at scores in a moment.

Let's take a look at the results first.

```
In [10]: len(resp['response']['docs'])

Out[10]: 10

In [11]: docs = resp['response']['docs']
         docs[0]

Out[11]: {u'_version_': 1494510828527812608,
          u'author': [u'Isaac Asimov'],
          u'cat': [u'book'],
          u'genre_s': u'scifi',
          u'id': u'0553293354',
          u'inStock': [True],
          u'name': [u'Foundation'],
          u'price': [7.99],
          u'sequence_i': 1,
          u'series_t': [u'Foundation Novels']}

In [12]: docs[1]
```

```
Out[12]: {u'_version_': 1494510737274437632,
          u'cat': [u'software', u'search'],
          u'features': [u'No accents here',
           u'This is an e acute: \xe9',
           u'eaiou with circumflexes: \xea\xe2\xee\xf4\xfb',
           u'eaiou with umlauts: \xeb\xe4\xef\xf6\xfc',
           u'tag with escaped chars: <nicetag/>',
           u'escaped ampersand: Bonnie & Clyde',
           u'Outside the BMP:\U00010308 codepoint=10308, a circle with an x inside. UTF8=f0908c88 UTF16=
          u'id': u'UTF8TEST',
          u'inStock': [True],
          u'manu': [u'Apache Software Foundation'],
          u'name': [u'Test with some UTF-8 encoded characters'],
          u'price': [0.0]}

In [21]: for doc in docs:
             print "%s: %s, $%s" % (doc['id'], doc.get('name', '(no title)'), doc.get('price', '-'))

0553293354: [u'Foundation'], $[7.99]
UTF8TEST: [u'Test with some UTF-8 encoded characters'], $[0.0]
SOLR1000: [u'Solr, the Enterprise Search Server'], $[0.0]
/Users/dchud/Downloads/apps/solr-5.0.0/docs/solr-analytics/deprecated-list.html: (no title), $-
/Users/dchud/Downloads/apps/solr-5.0.0/docs/solr-clustering/org/apache/solr/handler/clustering/carrot2/p
/Users/dchud/Downloads/apps/solr-5.0.0/docs/solr-core/org/apache/solr/logging/log4j/package-use.html: (n
/Users/dchud/Downloads/apps/solr-5.0.0/docs/solr-core/org/apache/solr/spelling/suggest/tst/package-use.h
/Users/dchud/Downloads/apps/solr-5.0.0/docs/solr-dataimporthandler-extras/deprecated-list.html: (no titl
/Users/dchud/Downloads/apps/solr-5.0.0/docs/solr-solrj/org/apache/solr/client/solrj/util/package-use.htm
/Users/dchud/Downloads/apps/solr-5.0.0/docs/solr-test-framework/org/apache/solr/analysis/package-use.htm
```

Let's revisit this query and add relevance rank scores to the results.

```
In [22]: req = requests.get("http://localhost:8983/solr/gettingstarted/select?wt=json&indent=true&q=foun
         resp = req.json()
         docs = resp['response']['docs']
         docs[0]

Out[22]: {u'_version_': 1494510828527812608,
          u'author': [u'Isaac Asimov'],
          u'cat': [u'book'],
          u'genre_s': u'scifi',
          u'id': u'0553293354',
          u'inStock': [True],
          u'name': [u'Foundation'],
          u'price': [7.99],
          u'score': 0.37489003,
          u'sequence_i': 1,
          u'series_t': [u'Foundation Novels']}

In [24]: for doc in docs:
             print "%s - %s" % (doc['score'], doc.get('name', '(no title)'))

0.37489003 - [u'Foundation']
0.13254364 - [u'Test with some UTF-8 encoded characters']
0.115975685 - [u'Solr, the Enterprise Search Server']
0.05857657 - (no title)
0.05857657 - (no title)
```

```
0.05857657 - (no title)
0.05857657 - (no title)
0.05857657 - (no title)
0.05857657 - (no title)
0.05857657 - (no title)
```

3 Under the hood

Now let's take a closer look at how those scores are generated. Solr offers easy tools for examining a live index and debugging information about results.

```
In [25]: # Add parameter debugQuery=true, repeat
         req = requests.get("http://localhost:8983/solr/gettingstarted/select?wt=json&indent=true&q=fou
         resp = req.json()
         resp.keys()

Out[25]: [u'debug', u'responseHeader', u'response']

In [26]: debug = resp['debug']
         debug.keys()

Out[26]: [u'parsedquery',
          u'track',
          u'explain',
          u'querystring',
          u'rawquerystring',
          u'parsedquery_toString',
          u'QParser',
          u'timing']
```

First, we can see that the query is parsed to determine how it should be processed. In the absence of named fields, it assumes a default, here `_text`.

```
In [30]: debug['QParser']

Out[30]: u'LuceneQParser'

In [27]: debug['rawquerystring']

Out[27]: u'foundation'

In [29]: debug['parsedquery']

Out[29]: u'_text:foundation'
```

The `timing` element gives the processing time in milliseconds. Note that a lot of tasks were not performed.

```
In [31]: debug['timing']

Out[31]: {u'prepare': {u'debug': {u'time': 0.0},
           u'expand': {u'time': 0.0},
           u'facet': {u'time': 0.0},
           u'highlight': {u'time': 0.0},
           u'mlt': {u'time': 0.0},
           u'query': {u'time': 4.0},
           u'stats': {u'time': 0.0},
           u'time': 4.0},
```

```
 u'process': {u'debug': {u'time': 36.0},
  u'expand': {u'time': 0.0},
  u'facet': {u'time': 0.0},
  u'highlight': {u'time': 0.0},
  u'mlt': {u'time': 0.0},
  u'query': {u'time': 6.0},
  u'stats': {u'time': 0.0},
  u'time': 42.0},
 u'time': 46.0}
```

3.1 The good stuff: relevance scoring

Now we get to the good bits. The `explain` element contains the precise scoring details for each aspect of a parsed query.

```
In [34]: explain = debug['explain']
         explain.keys()

Out[34]: [u'0553293354',
          u'UTF8TEST',
          u'/Users/dchud/Downloads/apps/solr-5.0.0/docs/solr-core/org/apache/solr/spelling/suggest/tst/p
          u'/Users/dchud/Downloads/apps/solr-5.0.0/docs/solr-solrj/org/apache/solr/client/solrj/util/pa
          u'/Users/dchud/Downloads/apps/solr-5.0.0/docs/solr-core/org/apache/solr/logging/log4j/package-
          u'/Users/dchud/Downloads/apps/solr-5.0.0/docs/solr-dataimporthandler-extras/deprecated-list.h
          u'/Users/dchud/Downloads/apps/solr-5.0.0/docs/solr-test-framework/org/apache/solr/analysis/pa
          u'SOLR1000',
          u'/Users/dchud/Downloads/apps/solr-5.0.0/docs/solr-clustering/org/apache/solr/handler/cluster
          u'/Users/dchud/Downloads/apps/solr-5.0.0/docs/solr-analytics/deprecated-list.html']

In [36]: print explain['UTF8TEST']

0.13254364 = (MATCH) weight(_text:foundation in 16) [DefaultSimilarity], result of:
  0.13254364 = fieldWeight in 16, product of:
    1.0 = tf(freq=1.0), with freq of:
      1.0 = termFreq=1.0
    1.0603491 = idf(docFreq=1494, maxDocs=1588)
    0.125 = fieldNorm(doc=16)

In [37]: print explain['0553293354']

0.37489003 = (MATCH) weight(_text:foundation in 3) [DefaultSimilarity], result of:
  0.37489003 = fieldWeight in 3, product of:
    1.4142135 = tf(freq=2.0), with freq of:
      2.0 = termFreq=2.0
    1.0603491 = idf(docFreq=1494, maxDocs=1588)
    0.25 = fieldNorm(doc=3)

In [45]: req = requests.get("http://localhost:8983/solr/gettingstarted/select?wt=json&indent=true&q=chi
         resp = req.json()
         resp['debug']['parsedquery']

Out[45]: u'_text:chinese name:tokenizer'

In [46]: for doc in resp['response']['docs']:
             print "%s - %s: %s" % (doc['score'], doc['id'], doc.get('name', '(no name)'))
```

```
0.10238882 - /Users/dchud/Downloads/apps/solr-5.0.0/docs/solr-clustering/org/apache/solr/handler/cluste
0.09220693 - /Users/dchud/Downloads/apps/solr-5.0.0/docs/solr-clustering/org/apache/solr/handler/cluste
0.009314307 - /Users/dchud/Downloads/apps/solr-5.0.0/docs/changes/Changes.html: (no name)
```

Because this is a slightly more complex query, we can see a lot more going on in the explanations.

```
In [50]: explain = resp['debug']['explain']
         for id, result in explain.items():
             print "%s%s" % (id, result)
```

```
/Users/dchud/Downloads/apps/solr-5.0.0/docs/changes/Changes.html
0.009314307 = (MATCH) product of:
  0.018628614 = (MATCH) sum of:
    0.018628614 = (MATCH) weight(_text:chinese in 2) [DefaultSimilarity], result of:
      0.018628614 = score(doc=2,freq=4.0), product of:
        0.6558272 = queryWeight, product of:
          7.2716184 = idf(docFreq=2, maxDocs=1588)
          0.09018999 = queryNorm
        0.02840476 = fieldWeight in 2, product of:
          2.0 = tf(freq=4.0), with freq of:
            4.0 = termFreq=4.0
          7.2716184 = idf(docFreq=2, maxDocs=1588)
          0.001953125 = fieldNorm(doc=2)
  0.5 = coord(1/2)

/Users/dchud/Downloads/apps/solr-5.0.0/docs/solr-clustering/org/apache/solr/handler/clustering/carrot2/
0.09220693 = (MATCH) product of:
  0.18441387 = (MATCH) sum of:
    0.18441387 = (MATCH) weight(_text:chinese in 131) [DefaultSimilarity], result of:
      0.18441387 = score(doc=131,freq=2.0), product of:
        0.6558272 = queryWeight, product of:
          7.2716184 = idf(docFreq=2, maxDocs=1588)
          0.09018999 = queryNorm
        0.28119275 = fieldWeight in 131, product of:
          1.4142135 = tf(freq=2.0), with freq of:
            2.0 = termFreq=2.0
          7.2716184 = idf(docFreq=2, maxDocs=1588)
          0.02734375 = fieldNorm(doc=131)
  0.5 = coord(1/2)

/Users/dchud/Downloads/apps/solr-5.0.0/docs/solr-clustering/org/apache/solr/handler/clustering/carrot2/
0.10238882 = (MATCH) product of:
  0.20477764 = (MATCH) sum of:
    0.20477764 = (MATCH) weight(_text:chinese in 122) [DefaultSimilarity], result of:
      0.20477764 = score(doc=122,freq=4.0), product of:
        0.6762287 = queryWeight, product of:
          7.75227 = idf(docFreq=1, maxDocs=1712)
          0.087229766 = queryNorm
        0.30282307 = fieldWeight in 122, product of:
          2.0 = tf(freq=4.0), with freq of:
            4.0 = termFreq=4.0
          7.75227 = idf(docFreq=1, maxDocs=1712)
          0.01953125 = fieldNorm(doc=122)
  0.5 = coord(1/2)
```

```
In [ ]:
```